jvl_git Scan Report

Project Name	jvl_git
Scan Start	Wednesday, October 28, 2020 11:43:37 PM
Preset	Checkmarx Default
Scan Time	00h:01m:36s
Lines Of Code Scanned	6912
Files Scanned	69
Report Creation Time	Wednesday, October 28, 2020 11:45:19 PM
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36
Team	CxServer
Checkmarx Version	9.2.0.41015 HF4
Scan Type	Full
Source Origin	GIT
Branch	refs/heads/master
Density	1/10 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

Included: Confirmed, Not Exploitable, To Verify, Urgent, Proposed Not Exploitable

Excluded: None

Assigned to

Included: All

Categories

Included:

Uncategorized	All
Custom	All
PCI DSS v3.2	All
OWASP Top 10 2013	All
FISMA 2014	All
NIST SP 800-53	All
OWASP Top 10 2017	All
OWASP Mobile Top 10 2016	All

Excluded:

Uncategorized	None
Custom	None
PCI DSS v3.2	None
OWASP Top 10 2013	None

FISMA 2014	None
NIST SP 800-53	None
OWASP Top 10 2017	None
OWASP Mobile Top 10 2016	None

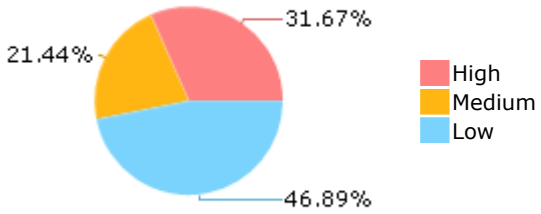
Results Limit

Results limit per query was set to 50

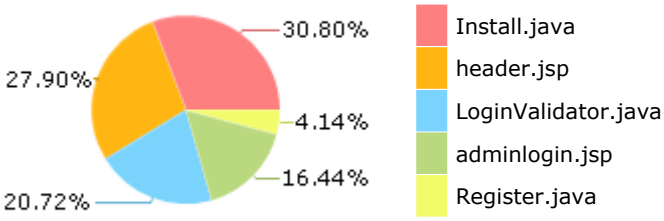
Selected Queries

Selected queries are listed in Result Summary

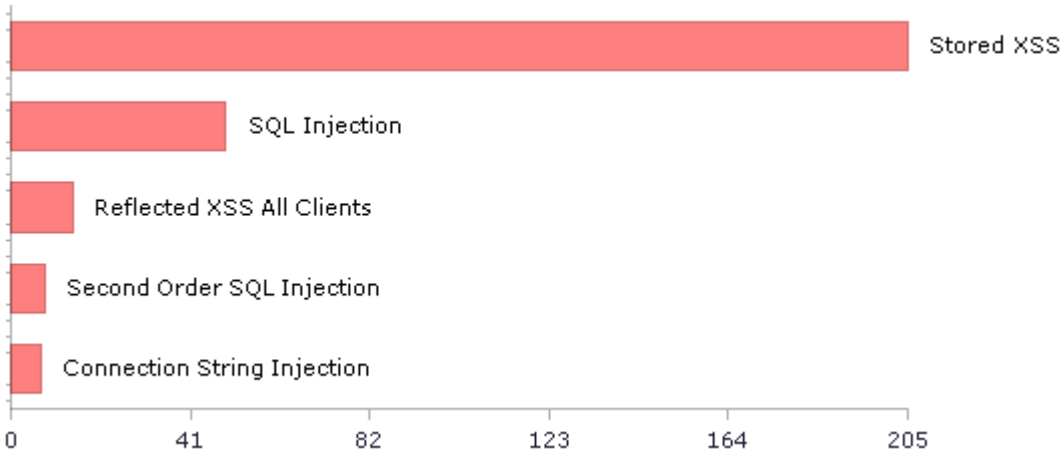
Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 2017

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2017

Category	Threat Agent	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection*	App. Specific	EASY	COMMON	EASY	SEVERE	App. Specific	115	58
A2-Broken Authentication*	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	12	3
A3-Sensitive Data Exposure*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	25	20
A4-XML External Entities (XXE)	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	1	1
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	46	24
A6-Security Misconfiguration *	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	62	39
A7-Cross-Site Scripting (XSS)*	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	230	37
A8-Insecure Deserialization	App. Specific	DIFFICULT	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A9-Using Components with Known Vulnerabilities*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	MODERATE	App. Specific	0	0
A10-Insufficient Logging & Monitoring	App. Specific	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	App. Specific	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Top 10 2013

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2013

Category	Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection*	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	AVERAGE	SEVERE	ALL DATA	102	50
A2-Broken Authentication and Session Management*	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	8	1
A3-Cross-Site Scripting (XSS)*	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	227	34
A4-Insecure Direct Object References*	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	11	9
A5-Security Misconfiguration *	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	42	19
A6-Sensitive Data Exposure*	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	25	20
A7-Missing Function Level Access Control*	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	9	9
A8-Cross-Site Request Forgery (CSRF)*	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	120	23
A9-Using Components with Known Vulnerabilities*	EXTERNAL USERS, AUTOMATED TOOLS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	5	2

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection	109	53
PCI DSS (3.2) - 6.5.2 - Buffer overflows	0	0
PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage*	2	2
PCI DSS (3.2) - 6.5.4 - Insecure communications*	1	1
PCI DSS (3.2) - 6.5.5 - Improper error handling*	119	115
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	237	39
PCI DSS (3.2) - 6.5.8 - Improper access control*	16	11
PCI DSS (3.2) - 6.5.9 - Cross-site request forgery*	111	14
PCI DSS (3.2) - 6.5.10 - Broken authentication and session management	8	1

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - FISMA 2014

Category	Description	Issues Found	Best Fix Locations
Access Control*	Organizations must limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems) and to the types of transactions and functions that authorized users are permitted to exercise.	18	18
Audit And Accountability*	Organizations must: (i) create, protect, and retain information system audit records to the extent needed to enable the monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity; and (ii) ensure that the actions of individual information system users can be uniquely traced to those users so they can be held accountable for their actions.	0	0
Configuration Management*	Organizations must: (i) establish and maintain baseline configurations and inventories of organizational information systems (including hardware, software, firmware, and documentation) throughout the respective system development life cycles; and (ii) establish and enforce security configuration settings for information technology products employed in organizational information systems.	48	25
Identification And Authentication*	Organizations must identify information system users, processes acting on behalf of users, or devices and authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.	76	72
Media Protection*	Organizations must: (i) protect information system media, both paper and digital; (ii) limit access to information on information system media to authorized users; and (iii) sanitize or destroy information system media before disposal or release for reuse.	9	8
System And Communications Protection	Organizations must: (i) monitor, control, and protect organizational communications (i.e., information transmitted or received by organizational information systems) at the external boundaries and key internal boundaries of the information systems; and (ii) employ architectural designs, software development techniques, and systems engineering principles that promote effective information security within organizational information systems.	0	0
System And Information Integrity*	Organizations must: (i) identify, report, and correct information and information system flaws in a timely manner; (ii) provide protection from malicious code at appropriate locations within organizational information systems; and (iii) monitor information system security alerts and advisories and take appropriate actions in response.	376	108

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)	0	0
AC-3 Access Enforcement (P1)*	121	102
AC-4 Information Flow Enforcement (P1)	0	0
AC-6 Least Privilege (P1)	0	0
AU-9 Protection of Audit Information (P1)	0	0
CM-6 Configuration Settings (P2)	0	0
IA-5 Authenticator Management (P1)	0	0
IA-6 Authenticator Feedback (P2)	0	0
IA-8 Identification and Authentication (Non-Organizational Users) (P1)	0	0
SC-12 Cryptographic Key Establishment and Management (P1)	0	0
SC-13 Cryptographic Protection (P1)	4	3
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)	2	2
SC-23 Session Authenticity (P1)*	111	14
SC-28 Protection of Information at Rest (P1)*	1	1
SC-4 Information in Shared Resources (P1)	71	26
SC-5 Denial of Service Protection (P1)*	135	132
SC-8 Transmission Confidentiality and Integrity (P1)	4	4
SI-10 Information Input Validation (P1)*	154	76
SI-11 Error Handling (P2)*	15	11
SI-15 Information Output Filtering (P0)*	222	32
SI-16 Memory Protection (P1)*	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Mobile Top 10 2016

Category	Description	Issues Found	Best Fix Locations
M1-Improper Platform Usage*	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.	0	0
M2-Insecure Data Storage*	This category covers insecure data storage and unintended data leakage.	0	0
M3-Insecure Communication*	This category covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.	0	0
M4-Insecure Authentication*	This category captures notions of authenticating the end user or bad session management. This can include: -Failing to identify the user at all when that should be required -Failure to maintain the user's identity when it is required -Weaknesses in session management	0	0
M5-Insufficient Cryptography*	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasnt done correctly.	0	0
M6-Insecure Authorization*	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.). If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.	0	0
M7-Client Code Quality*	This category is the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.	56	28
M8-Code Tampering*	This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or	0	0

	modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.		
M9-Reverse Engineering*	This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.	0	0
M10-Extraneous Functionality*	Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

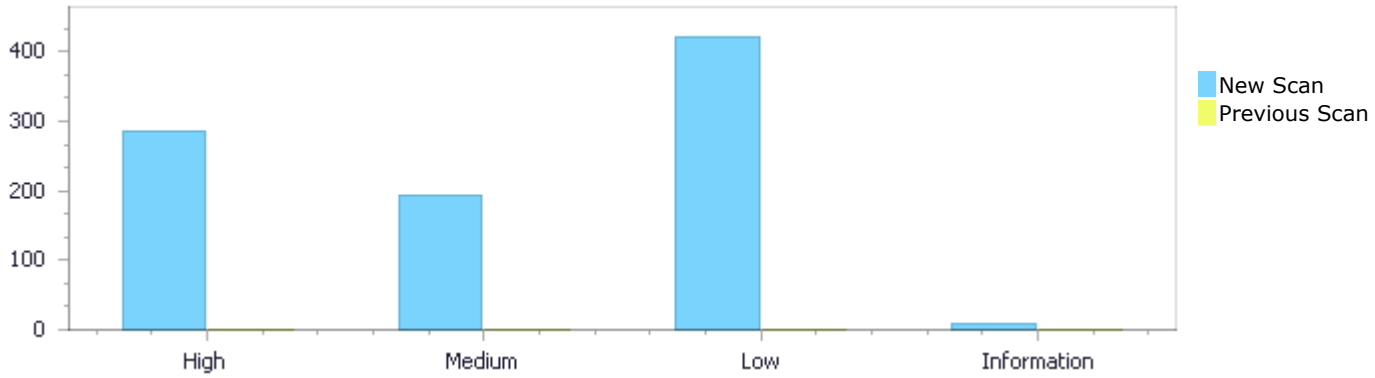
Scan Summary - Custom

Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	284	193	422	8	907
Recurrent Issues	0	0	0	0	0
Total	284	193	422	8	907
Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	1	0	0	0	1
To Verify	284	193	422	8	907
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	285	193	422	8	908

Result Summary

Vulnerability Type	Occurrences	Severity
Stored XSS	205	High
SQL Injection	49	High
Reflected XSS All Clients	14	High
Second Order SQL Injection	8	High
Connection String Injection	7	High

XPath Injection	2	High
XSRF	111	Medium
Trust Boundary Violation	12	Medium
HTTP Response Splitting	10	Medium
Cross Site History Manipulation	9	Medium
External Control of System or Config Setting	8	Medium
HttpOnlyCookies	7	Medium
Input Path Not Canonicalized	7	Medium
Privacy Violation	7	Medium
SSRF	7	Medium
Heap Inspection	6	Medium
Use of a One Way Hash with a Predictable Salt	2	Medium
Absolute Path Traversal	1	Medium
Download of Code Without Integrity Check	1	Medium
HttpOnlyCookies In Config	1	Medium
Improper Restriction of XXE Ref	1	Medium
Missing HSTS Header	1	Medium
Plaintext Storage of a Password	1	Medium
Unchecked Input for Loop Condition	1	Medium
Improper Exception Handling	104	Low
Improper Resource Access Authorization	68	Low
Unsynchronized Access To Shared Data	50	Low
Blind SQL Injections	37	Low
Improper Resource Shutdown or Release	31	Low
Exposure of System Data	27	Low
Incorrect Permission Assignment For Critical Resources	18	Low
Information Exposure Through an Error Message	15	Low
Information Exposure Through Query String	11	Low
Data Leak Between Sessions	8	Low
Race Condition	8	Low
Stored Boundary Violation	7	Low
Relative Path Traversal	6	Low
Open Redirect	5	Low
Reliance on Cookies in a Decision	4	Low
Sensitive Cookie in HTTPS Session Without Secure Attribute	4	Low
Suspected XSS	3	Low
Plaintext Storage in a Cookie	2	Low
Stored Absolute Path Traversal	2	Low
Stored Relative Path Traversal	2	Low
Creation of Temp File in Dir with Incorrect Permissions	1	Low
Information Leak Through Shell Error Message	1	Low
Missing Content Security Policy	1	Low
Missing X Frame Options	1	Low
Potential Clickjacking on Legacy Browsers	1	Low
Reversible One Way Hash	1	Low
Stored HTTP Response Splitting	1	Low
Unrestricted File Upload	1	Low
Use of Broken or Risky Cryptographic Algorithm	1	Low
Use of Non Cryptographic Random	1	Low
Portability Flaw In File Separator	8	Information

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
src/main/webapp/header.jsp	160
src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	132
src/main/webapp/admin/adminlogin.jsp	110
src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	106
src/main/webapp/changeCardDetails.jsp	16
src/main/webapp/vulnerability/forum.jsp	15
src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	14
src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	13
src/main/webapp/vulnerability/csrf/change-info.jsp	12
src/main/webapp/vulnerability/csrf/changepassword.jsp	11

Scan Results Details

Stored XSS

Query Path:
Java\Cx\Java High Risk\Stored XSS Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)
OWASP Top 10 2013: A3-Cross-Site Scripting (XSS)
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-15 Information Output Filtering (P0)
OWASP Top 10 2017: A7-Cross-Site Scripting (XSS)

Description
Stored XSS\Path 1:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=91
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp
Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 2:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=92
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 3:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=93
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 4:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=94
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 5:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=95
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 6:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=96
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 7:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=97
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 8:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=98
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 9:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=99
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 10:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=100
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 11:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=101
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 12:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=102
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 13:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=103
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 14:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=104
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 15:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=105
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 16:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=106
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 17:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=107
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 18:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=108
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 19:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=109
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 20:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=110
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 21:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=111
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 22:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=112
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 23:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=113
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 24:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=114
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 25:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=115
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 26:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=116
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 27:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=117
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 28:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=118
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 29:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=119
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 30:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=120
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 31:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=121
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 32:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=122
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 33:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=123
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 34:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=124
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 35:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=125
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp

Method	<a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 36:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=126
Status	New

The application's <input type="hidden" name="id" value="<% out.print(session.getAttribute("userid"));%>"/> embeds untrusted data in the generated output with print, at line 19 of src/main/webapp/vulnerability/idor/change-email.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/idor/change-email.jsp
Line	52	19
Object	rs	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/vulnerability/idor/change-email.jsp
Method	<input type="hidden" name="id" value="<% out.print(session.getAttribute("userid"));%>"/>

```
....
19.          <input type="hidden" name="id" value="<%
out.print(session.getAttribute("userid"));%>"/>
```

Stored XSS\Path 37:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=127
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                    rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 38:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=128
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 39:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=129
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 40:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=130
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 41:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=131
Status	New

The application's <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details embeds untrusted data in the generated output with print, at line 87 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	87
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method <a href="<%=path%>/myprofile.jsp?id=<% if(session.getAttribute("userid")!=null){ out.print(session.getAttribute("userid"));} %>" title="Make sure you have logged in ">Viewing Details

```
....
87.                                                 <li><a
href="<%=path%>/myprofile.jsp?id=<%
if(session.getAttribute("userid")!=null){
out.print(session.getAttribute("userid"));} %>" title="Make sure you
have logged in ">Viewing Details</a>
```

Stored XSS\Path 42:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=132
Status	New

The application's <td><input type="hidden" name="sender" value="<%=session.getAttribute("user")%>"/></td> embeds untrusted data in the generated output with getAttribute, at line 21 of src/main/webapp/vulnerability/SendMessage.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/SendMessage.jsp
Line	52	21
Object	rs	getAttribute

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/vulnerability/SendMessage.jsp

Method <tr> <td><input type="hidden" name="sender" value="<%=session.getAttribute("user")%>"/></td></tr>

```
....
21.  <tr> <td><input type="hidden" name="sender"
value="<%=session.getAttribute("user")%>"/></td></tr>
```

Stored XSS\Path 43:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=133
Status	New

The application's <input type="hidden" name="user" value="<% if(session.getAttribute("user")!=null){out.print(session.getAttribute("user"));} else { out.print("Anonymous"); } %>" size="50"/>
 embeds untrusted data in the generated output with print, at line 32 of src/main/webapp/vulnerability/forum.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/forum.jsp
Line	52	32
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/vulnerability/forum.jsp

Method <input type="hidden" name="user" value="<% if(session.getAttribute("user")!=null){out.print(session.getAttribute("user"));} else { out.print("Anonymous"); } %>" size="50"/>

```
....
32.         <input type="hidden" name="user" value="<%
if(session.getAttribute("user")!=null){out.print(session.getAttribute("u
ser"));} else { out.print("Anonymous"); } %>" size="50"/><br/>
```

Stored XSS\Path 44:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=134
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 148 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	148
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method out.print("
My Profile
");

```
....
148.                                    out.print("<li><a
href='"+path+"/myprofile.jsp?id="+session.getAttribute("userid")+"'>My
Profile</a></li>");
```

Stored XSS\Path 45:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=135

Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 148 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	148
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method out.print("
My Profile
");

```
....
148.                                    out.print("<li><a
href='"+path+"/myprofile.jsp?id="+session.getAttribute("userid")+"'>My
Profile</a></li>");
```

Stored XSS\Path 46:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=136
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 148 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	148
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method out.print("
My Profile
");

```
....
148.                                        out.print("<li><a
href='"+path+"/myprofile.jsp?id="+session.getAttribute("userid")+"'>My
Profile</a></li>");
```

Stored XSS\Path 47:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=137
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 148 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	148
Object	rs	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/header.jsp

Method out.print("
My Profile
");

```
....
148.                                        out.print("<li><a
href='"+path+"/myprofile.jsp?id="+session.getAttribute("userid")+"'>My
Profile</a></li>");
```

Stored XSS\Path 48:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=138
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 36 of src/main/webapp/vulnerability/idor/change-email.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/idor/change-email.jsp
Line	52	36

Object	rs	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method out.print("

Return to Profile Page >>");

```
....
36.      out.print("<br/><br/><a
href='"+path+"/myprofile.jsp?id="+session.getAttribute("userid")+"'>Retu
rn to Profile Page &gt;&gt;</a>");
```

Stored XSS\Path 49:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=139
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 148 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	148
Object	rs	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp
Method	out.print(" My Profile ");

```
....
148.                                    out.print("<li><a
href='"+path+"/myprofile.jsp?id="+session.getAttribute("userid")+"'>My
Profile</a></li>");
```

Stored XSS\Path 50:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=140
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 148 of src/main/webapp/header.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the processRequest method with rs, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/header.jsp
Line	52	148
Object	rs	print

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name	src/main/webapp/header.jsp
Method	out.print(" My Profile ");

```
....
148.                                        out.print("<li><a
href='"+path+"/myprofile.jsp?id="+session.getAttribute("userid")+"'>My
Profile</a></li>");
```

SQL Injection

Query Path:
Java\Cx\Java High Risk\SQL Injection Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
OWASP Top 10 2013: A1-Injection
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection
OWASP Mobile Top 10 2016: M7-Client Code Quality

Description
SQL Injection\Path 1:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=32
Status	New

The application's processRequest method executes an SQL query with executeQuery, at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""email""; this input is then read by the processRequest method at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Line	42	48
Object	""email""	executeQuery

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
42.                 String email=request.getParameter("email").trim();
....
48.                      rs=stmt.executeQuery("select * from users where
email='"+email+"'");
```

SQL Injection\Path 2:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=33
Status	New

The application's setup method executes an SQL query with executeUpdate, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""dbname""; this input is then read by the processRequest method at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	117
Object	""dbname""	executeUpdate

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
117.                                 stmt.executeUpdate("DROP DATABASE IF
EXISTS "+dbname);
```

SQL Injection\Path 3:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=34
Status	New

The application's setup method executes an SQL query with executeUpdate, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""dbname""; this input is then read by the processRequest method at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	119
Object	""dbname""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.           dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
119.                                  stmt.executeUpdate("CREATE DATABASE
"+dbname);
```

SQL Injection\Path 4:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=35
Status	New

The application's setup method executes an SQL query with executeUpdate, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. The application constructs this SQL query by

embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""adminuser""; this input is then read by the processRequest method at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	60	127
Object	""adminuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.          adminuser= request.getParameter("adminuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

SQL Injection\Path 5:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=36
Status	New

The application's processRequest method executes an SQL query with executeQuery, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	52
Object	""username""	executeQuery

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

SQL Injection\Path 6:

Severity High
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=37
Status New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/changeCardDetails.jsp
Line	43	43
Object	""username""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

SQL Injection\Path 7:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=38
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 31 of src/main/webapp/vulnerability/csrf/change-info.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	43	31
Object	""username""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp

Method	stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.              stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

SQL Injection\Path 8:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=39
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 40 of src/main/webapp/vulnerability/csrf/changepassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	43	40
Object	""username""	executeUpdate

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.          String user=request.getParameter("username").trim();
```

File Name	src/main/webapp/vulnerability/csrf/changepassword.jsp
Method	stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.                 stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

SQL Injection\Path 9:

Severity	High

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=40
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/Messages.jsp
Line	43	14
Object	""username""	executeQuery

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
```

File Name src/main/webapp/vulnerability/Messages.jsp

Method rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.              rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

SQL Injection\Path 10:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=41
Status	New

The application's processRequest method executes an SQL query with executeQuery, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. The application constructs this SQL query

by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""password""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	52
Object	""password""	executeQuery

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

SQL Injection\Path 11:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=42
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""password""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/changeCardDetails.jsp
Line	44	43

Object	""password""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password").trim();
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

SQL Injection\Path 12:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=43
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 31 of src/main/webapp/vulnerability/csrf/change-info.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""password""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	44	31
Object	""password""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
```

File Name	src/main/webapp/vulnerability/csrf/change-info.jsp
Method	stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.               stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

SQL Injection\Path 13:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=44
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 40 of src/main/webapp/vulnerability/csrf/changepassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""password""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	44	40
Object	""password""	executeUpdate

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
```

File Name	src/main/webapp/vulnerability/csrf/changepassword.jsp
Method	stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.                 stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

SQL Injection\Path 14:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=45
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""password""; this input is then read by the processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/Messages.jsp
Line	44	14
Object	""password""	executeQuery

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
```

File Name src/main/webapp/vulnerability/Messages.jsp

Method rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.                 rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

SQL Injection\Path 15:

Severity	High
Result State	To Verify

Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=46
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	43	58
Object	""username""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.           String user=request.getParameter("username");
....
58.                                       stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

SQL Injection\Path 16:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=47
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	43	59
Object	""username""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.           String user=request.getParameter("username");
....
59.                                              stmt.executeUpdate("INSERT
into UserMessages(recipient, sender, subject, msg) values
('"+user+"','admin','Hi','Hi<br/> This is admin of this page. <br/>
Welcome to Our Forum')");
```

SQL Injection\Path 17:

Severity High
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=48
Status New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""password""; this input is then read by the processRequest method at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	44	58
Object	""password""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password");
....
58.                                        stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

SQL Injection\Path 18:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=49
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""email""; this input is then read by the processRequest method at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	45	58
Object	""email""	executeUpdate

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
45.            String email=request.getParameter("email");
....
58.                                        stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

SQL Injection\Path 19:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=50
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""About""; this input is then read by the processRequest method at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	46	58
Object	""About""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
46.            String about=request.getParameter("About");
....
58.                                        stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

SQL Injection\Path 20:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=51
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""secret""; this input is then read by the processRequest method at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	47	58
Object	""secret""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
47.            String secret=request.getParameter("secret");
....
58.                                        stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

SQL Injection\Path 21:

Severity High
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=52
Status New

The application's processRequest method executes an SQL query with executeQuery, at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Line	42	48
Object	""username""	executeQuery

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
42.                 String user=request.getParameter("username").trim();
....
48.                       rs=stmt.executeQuery("select * from users where
username='"+user+"'");
```

SQL Injection\Path 22:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=53
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 19 of src/main/webapp/admin/adminlogin.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the user=request.getParameter method at line 11 of src/main/webapp/admin/adminlogin.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	11	19
Object	""username""	executeQuery

Code Snippet

File Name	src/main/webapp/admin/adminlogin.jsp
Method	String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name	src/main/webapp/admin/adminlogin.jsp
Method	rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                       rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

SQL Injection\Path 23:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=54
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the user=request.getParameter method at line 11 of src/main/webapp/admin/adminlogin.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/changeCardDetails.jsp
Line	11	43
Object	""username""	executeUpdate

Code Snippet

File Name src/main/webapp/admin/adminlogin.jsp

Method String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

SQL Injection\Path 24:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=55
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 31 of src/main/webapp/vulnerability/csrf/change-info.jsp. The application constructs this SQL query by embedding

an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the user=request.getParameter method at line 11 of src/main/webapp/admin/adminlogin.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	11	31
Object	""username""	executeUpdate

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp
Method stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.              stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

SQL Injection\Path 25:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=56
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 40 of src/main/webapp/vulnerability/csrf/changepassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the user=request.getParameter method at line 11 of src/main/webapp/admin/adminlogin.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	11	40
Object	""username""	executeUpdate

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp
Method stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.               stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

SQL Injection\Path 26:

Severity High
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=57
Status New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the user=request.getParameter method at line 11 of src/main/webapp/admin/adminlogin.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/vulnerability/Messages.jsp
Line	11	14
Object	""username""	executeQuery

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp

Method	String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name	src/main/webapp/vulnerability/Messages.jsp
Method	rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.               rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

SQL Injection\Path 27:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=58
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 14 of src/main/webapp/admin/manageusers.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""user""; this input is then read by the user=request.getParameter method at line 13 of src/main/webapp/admin/manageusers.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	13	14
Object	""user""	executeUpdate

Code Snippet	
File Name	src/main/webapp/admin/manageusers.jsp
Method	String user=request.getParameter("user");

```
....
13.       String user=request.getParameter("user");
```

File Name	src/main/webapp/admin/manageusers.jsp
Method	stmt.executeUpdate("Delete from users where username='"+user+"'");

```
....
14.       stmt.executeUpdate("Delete from users where
username='"+user+"'");
```

SQL Injection\Path 28:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=59
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""cardno""; this input is then read by the cardno=request.getParameter method at line 37 of src/main/webapp/changeCardDetails.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	37	43
Object	""cardno""	executeUpdate

Code Snippet

File Name src/main/webapp/changeCardDetails.jsp

Method String cardno=request.getParameter("cardno");

```
....
37.          String cardno=request.getParameter("cardno");
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

SQL Injection\Path 29:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=60
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""cvv""; this input is then read by the cvv=request.getParameter method at line 38 of src/main/webapp/changeCardDetails.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	38	43
Object	""cvv""	executeUpdate

Code Snippet

File Name src/main/webapp/changeCardDetails.jsp

Method String cvv=request.getParameter("cvv");

```
....
38.          String cvv=request.getParameter("cvv");
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

SQL Injection\Path 30:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=61
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""expirydate""; this input is then read by the expirydate=request.getParameter method at line 39 of src/main/webapp/changeCardDetails.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	39	43
Object	""expirydate""	executeUpdate

Code Snippet
File Name src/main/webapp/changeCardDetails.jsp
Method String expirydate=request.getParameter("expirydate");

```
....
39.          String expirydate=request.getParameter("expirydate");
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

SQL Injection\Path 31:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=62
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 42 of src/main/webapp/ForgotPassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the rs=stmt.executeQuery method at line 42 of src/main/webapp/ForgotPassword.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination

File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	42	42
Object	""username""	executeQuery

Code Snippet
File Name src/main/webapp/ForgotPassword.jsp
Method rs=stmt.executeQuery("select * from users where username='"+request.getParameter("username").trim()+"' and secret='"+request.getParameter("secret")+"'");

```
....
42.                     rs=stmt.executeQuery("select * from users where
username='"+request.getParameter("username").trim()+"' and
secret='"+request.getParameter("secret")+"'");
```

SQL Injection\Path 32:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=63
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 42 of src/main/webapp/ForgotPassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""secret""; this input is then read by the rs=stmt.executeQuery method at line 42 of src/main/webapp/ForgotPassword.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	42	42
Object	""secret""	executeQuery

Code Snippet
File Name src/main/webapp/ForgotPassword.jsp
Method rs=stmt.executeQuery("select * from users where username='"+request.getParameter("username").trim()+"' and secret='"+request.getParameter("secret")+"'");

```
....
42.                     rs=stmt.executeQuery("select * from users where
username='"+request.getParameter("username").trim()+"' and
secret='"+request.getParameter("secret")+"'");
```

SQL Injection\Path 33:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=64
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 21 of src/main/webapp/myprofile.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""id""; this input is then read by the id=request.getParameter method at line 16 of src/main/webapp/myprofile.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	16	21
Object	""id""	executeQuery

Code Snippet

File Name src/main/webapp/myprofile.jsp

Method String id=request.getParameter("id");

```
....
16.     String id=request.getParameter("id");
```

File Name src/main/webapp/myprofile.jsp

Method rs=stmt.executeQuery("select * from users where id="+id);

```
....
21.                 rs=stmt.executeQuery("select * from users where
id="+id);
```

SQL Injection\Path 34:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=65
Status	New

The application's rs1=stmt.executeQuery method executes an SQL query with executeQuery, at line 29 of src/main/webapp/myprofile.jsp. The application constructs this SQL query by embedding an untrusted string

into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""id""; this input is then read by the id=request.getParameter method at line 16 of src/main/webapp/myprofile.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	16	29
Object	""id""	executeQuery

Code Snippet
File Name src/main/webapp/myprofile.jsp
Method String id=request.getParameter("id");

```
....
16.     String id=request.getParameter("id");
```

File Name src/main/webapp/myprofile.jsp
Method ResultSet rs1=stmt.executeQuery("select * from cards where id="+id);

```
....
29.                 ResultSet rs1=stmt.executeQuery("select * from
cards where id="+id);
```

SQL Injection\Path 35:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=66
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 31 of src/main/webapp/vulnerability/csrf/change-info.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""info""; this input is then read by the info=request.getParameter method at line 26 of src/main/webapp/vulnerability/csrf/change-info.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination

File	src/main/webapp/vulnerability/csrf/change-info.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	26	31
Object	""info""	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/csrf/change-info.jsp
Method String info=request.getParameter("info");

```
....
26.     String info=request.getParameter("info");
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp
Method stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.             stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

SQL Injection\Path 36:

Severity High
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=67
Status New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 40 of src/main/webapp/vulnerability/csrf/changepassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""password""; this input is then read by the pass=request.getParameter method at line 33 of src/main/webapp/vulnerability/csrf/changepassword.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	33	40
Object	""password""	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/csrf/changepassword.jsp
Method String pass=request.getParameter("password");

```
....
33.          String pass=request.getParameter("password");
```

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp

Method stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.               stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

SQL Injection\Path 37:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=68
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 16 of src/main/webapp/vulnerability/DisplayMessage.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""msgid""; this input is then read by the rs=stmt.executeQuery method at line 16 of src/main/webapp/vulnerability/DisplayMessage.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/DisplayMessage.jsp	src/main/webapp/vulnerability/DisplayMessage.jsp
Line	16	16
Object	""msgid""	executeQuery

Code Snippet

File Name src/main/webapp/vulnerability/DisplayMessage.jsp

Method rs=stmt.executeQuery("select * from UserMessages where msgid="+request.getParameter("msgid"));

```
....
16.               rs=stmt.executeQuery("select * from UserMessages where
msgid="+request.getParameter("msgid"));
```

SQL Injection\Path 38:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=69
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 48 of src/main/webapp/vulnerability/forum.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""content""; this input is then read by the content=request.getParameter method at line 42 of src/main/webapp/vulnerability/forum.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	42	48
Object	""content""	executeUpdate

Code Snippet

File Name src/main/webapp/vulnerability/forum.jsp

Method String content=request.getParameter("content");

```
....
42.                                          String
content=request.getParameter("content");
```

File Name src/main/webapp/vulnerability/forum.jsp

Method stmt.executeUpdate("INSERT into posts(content,title,user) values ('"+content+"','"+title+"','"+user+"')");

```
....
48.
stmt.executeUpdate("INSERT into posts(content,title,user) values
('"+content+"','"+title+"','"+user+"')");
```

SQL Injection\Path 39:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=70
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 48 of src/main/webapp/vulnerability/forum.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""title""; this input is then read by the title=request.getParameter method at line 43 of src/main/webapp/vulnerability/forum.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	43	48
Object	""title""	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method String title=request.getParameter("title");

```
....
43.                                          String
title=request.getParameter("title");
```

File Name src/main/webapp/vulnerability/forum.jsp

Method stmt.executeUpdate("INSERT into posts(content,title,user) values ('"+content+"','"+title+"','"+user+"')");

```
....
48.
stmt.executeUpdate("INSERT into posts(content,title,user) values
('"+content+"','"+title+"','"+user+"')");
```

SQL Injection\Path 40:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=71
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 48 of src/main/webapp/vulnerability/forum.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""user""; this input is then read by the user=request.getParameter method at line 41 of src/main/webapp/vulnerability/forum.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination

File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	41	48
Object	""user""	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method String user=request.getParameter("user");

```
....
41.                                          String
user=request.getParameter("user");
```

File Name src/main/webapp/vulnerability/forum.jsp

Method stmt.executeUpdate("INSERT into posts(content,title,user) values ('"+content+"','"+title+"','"+user+"')");

```
....
48.
stmt.executeUpdate("INSERT into posts(content,title,user) values
('"+content+"','"+title+"','"+user+"')");
```

SQL Injection\Path 41:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=72
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/forumposts.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""postid""; this input is then read by the postid=request.getParameter method at line 9 of src/main/webapp/vulnerability/forumposts.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/forumposts.jsp	src/main/webapp/vulnerability/forumposts.jsp
Line	9	14
Object	""postid""	executeQuery

Code Snippet
File Name src/main/webapp/vulnerability/forumposts.jsp

Method	String postid=request.getParameter("postid");

```
....
9.     String postid=request.getParameter("postid");
```

File Name	src/main/webapp/vulnerability/forumposts.jsp
Method	rs=stmt.executeQuery("select * from posts where postid="+postid);

```
....
14.               rs=stmt.executeQuery("select * from posts where
postid="+postid);
```

SQL Injection\Path 42:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=73
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 32 of src/main/webapp/vulnerability/idor/change-email.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""email""; this input is then read by the email=request.getParameter method at line 27 of src/main/webapp/vulnerability/idor/change-email.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/idor/change-email.jsp	src/main/webapp/vulnerability/idor/change-email.jsp
Line	27	32
Object	""email""	executeUpdate

Code Snippet

File Name	src/main/webapp/vulnerability/idor/change-email.jsp
Method	String email=request.getParameter("email");

```
....
27.     String email=request.getParameter("email");
```

File Name	src/main/webapp/vulnerability/idor/change-email.jsp
Method	stmt.executeUpdate("Update users set email='"+email+"' where id="+id);

```
....
32.             stmt.executeUpdate("Update users set email='"+email+"'
where id="+id);
```

SQL Injection\Path 43:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=74
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 32 of src/main/webapp/vulnerability/idor/change-email.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""id""; this input is then read by the id=request.getParameter method at line 28 of src/main/webapp/vulnerability/idor/change-email.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/idor/change-email.jsp	src/main/webapp/vulnerability/idor/change-email.jsp
Line	28	32
Object	""id""	executeUpdate

Code Snippet

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method String id=request.getParameter("id");

```
....
28.     String id=request.getParameter("id");
```

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method stmt.executeUpdate("Update users set email='"+email+"' where id="+id);

```
....
32.             stmt.executeUpdate("Update users set email='"+email+"'
where id="+id);
```

SQL Injection\Path 44:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=75
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the processRequest method at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/webapp/vulnerability/Messages.jsp
Line	35	14
Object	""username""	executeQuery

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
35.             String user=request.getParameter("username");
```

File Name src/main/webapp/vulnerability/Messages.jsp

Method rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.              rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

SQL Injection\Path 45:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=76
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""password""; this input is then read by the processRequest method at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/webapp/vulnerability/Messages.jsp
Line	36	14
Object	""password""	executeQuery

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             String pass=request.getParameter("password");
```

File Name src/main/webapp/vulnerability/Messages.jsp

Method rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.              rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

SQL Injection\Path 46:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=77
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 24 of src/main/webapp/vulnerability/sqli/download_id.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""fileid""; this input is then read by the fileid=request.getParameter method at line 18 of src/main/webapp/vulnerability/sqli/download_id.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination

File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	18	24
Object	""fileid""	executeQuery

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.              rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

SQL Injection\Path 47:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=78
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 24 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""fileid""; this input is then read by the fileid=request.getParameter method at line 18 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	18	24
Object	""fileid""	executeQuery

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.               rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

SQL Injection\Path 48:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=79
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 13 of src/main/webapp/vulnerability/UserDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""username""; this input is then read by the username=request.getParameter method at line 8 of src/main/webapp/vulnerability/UserDetails.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/UserDetails.jsp	src/main/webapp/vulnerability/UserDetails.jsp
Line	8	13
Object	""username""	executeQuery

Code Snippet

File Name src/main/webapp/vulnerability/UserDetails.jsp

Method String username=request.getParameter("username");

```
....
8.     String username=request.getParameter("username");
```

File Name src/main/webapp/vulnerability/UserDetails.jsp

Method rs=stmt.executeQuery("select * from users where username='"+username+"'");

```
....
13.                rs=stmt.executeQuery("select * from users where
username='"+username+"'");
```

SQL Injection\Path 49:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=80
Status	New

The application's session.createQuery method executes an SQL query with createQuery, at line 11 of src/main/webapp/vulnerability/Injection/orm.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

An attacker would be able to inject arbitrary syntax and data into the SQL query, by crafting a malicious payload and providing it via the input ""id""; this input is then read by the out.print method at line 50 of src/main/webapp/vulnerability/Injection/orm.jsp. This input then flows through the code, into a query and to the database server - without sanitization.

This may enable an SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/Injection/orm.jsp	src/main/webapp/vulnerability/Injection/orm.jsp
Line	50	11
Object	""id""	createQuery

Code Snippet

File Name src/main/webapp/vulnerability/Injection/orm.jsp

Method out.print(queryUsers(ormSession,request.getParameter("id")));

```
....
50.
out.print(queryUsers(ormSession,request.getParameter("id")));
```

File Name src/main/webapp/vulnerability/Injection/orm.jsp

Method Query query = session.createQuery("from Users where id="+id);

```
....
11.          Query query = session.createQuery("from Users where
id="+id);
```

Reflected XSS All Clients

Query Path:

Java\Cx\Java High Risk\Reflected XSS All Clients Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)
OWASP Top 10 2013: A3-Cross-Site Scripting (XSS)
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-15 Information Output Filtering (P0)
OWASP Top 10 2017: A7-Cross-Site Scripting (XSS)

Description

Reflected XSS All Clients\Path 1:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=18
Status	New

The application's processRequest embeds untrusted data in the generated output with print, at line 38 of src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input getInputStream, which is read by the processRequest method at line 38 of src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	44	54
Object	getInputStream	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             InputStream xml=request.getInputStream();
....
54.               out.print(nodes.item(i).getNodeName()+" : " +
nodes.item(i).getFirstChild().getNodeValue().toString());
```

Reflected XSS All Clients\Path 2:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=19
Status	New

The application's processRequest embeds untrusted data in the generated output with print, at line 34 of src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""filename"", which is read by the processRequest method at line 34 of src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	39	55
Object	""filename""	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.             String fileName=request.getParameter("filename");
....
55.                         out.print("Successfully created the file: <a
href='../pages/"+fileName+"'>"+fileName+"</a>");
```

Reflected XSS All Clients\Path 3:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=20
Status	New

The application's <td></td><td class="fail"><% if(request.getParameter("err")!=null){out.print(request.getParameter("err"));} %></td> embeds untrusted data in the generated output with print, at line 58 of src/main/webapp/admin/adminlogin.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""err"", which is read by the <td></td><td class="fail"><% if(request.getParameter("err")!=null){out.print(request.getParameter("err"));} %></td> method at line 58 of src/main/webapp/admin/adminlogin.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination

File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	58	58
Object	""err""	print

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method <tr><td></td><td class="fail"><%
if(request.getParameter("err")!=null){out.print(request.getParameter("err"));}
%></td></tr>

```
....
58.  <tr><td></td><td class="fail"><%
if(request.getParameter("err")!=null){out.print(request.getParameter("er
r"));} %></td></tr>
```

Reflected XSS All Clients\Path 4:

Severity High
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=21
Status New

The application's <td></td><td class="fail"><% if(request.getParameter("err")!=null){out.print(request.getParameter("err"));} %></td> embeds untrusted data in the generated output with print, at line 26 of src/main/webapp/login.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""err"", which is read by the <td></td><td class="fail"><% if(request.getParameter("err")!=null){out.print(request.getParameter("err"));} %></td> method at line 26 of src/main/webapp/login.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/login.jsp	src/main/webapp/login.jsp
Line	26	26
Object	""err""	print

Code Snippet
File Name src/main/webapp/login.jsp
Method <tr><td></td><td class="fail"><%
if(request.getParameter("err")!=null){out.print(request.getParameter("err"));}
%></td></tr>

```
....
26.  <tr><td></td><td class="fail"><%
if(request.getParameter("err")!=null){out.print(request.getParameter("er
r"));} %></td></tr>
```

Reflected XSS All Clients\Path 5:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=22
Status	New

The application's <td></td><td class="fail"><% if(request.getParameter("err")!=null){out.print(request.getParameter("err"));} %></td> embeds untrusted data in the generated output with print, at line 9 of src/main/webapp/vulnerability/Injection/xpath_login.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""err"", which is read by the <td></td><td class="fail"><% if(request.getParameter("err")!=null){out.print(request.getParameter("err"));} %></td> method at line 9 of src/main/webapp/vulnerability/Injection/xpath_login.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/Injection/xpath_login.jsp	src/main/webapp/vulnerability/Injection/xpath_login.jsp
Line	9	9
Object	""err""	print

Code Snippet

File Name src/main/webapp/vulnerability/Injection/xpath_login.jsp

Method <tr><td></td><td class="fail"><% if(request.getParameter("err")!=null){out.print(request.getParameter("err"));} %></td></tr>

```
....
9.  <tr><td></td><td class="fail"><%
if(request.getParameter("err")!=null){out.print(request.getParameter("er
r"));} %></td></tr>
```

Reflected XSS All Clients\Path 6:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=23
Status	New

The application's <c:import url='${param["style"]}' var="xslt"/> embeds untrusted data in the generated output with write, at line 14 of src/main/webapp/vulnerability/Injection/xslt.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""style"", which is read by the <c:import url='${param["style"]}' var="xslt"/> method at line 14 of src/main/webapp/vulnerability/Injection/xslt.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/Injection/xslt.jsp	src/main/webapp/vulnerability/Injection/xslt.jsp
Line	14	14
Object	""style""	write

Code Snippet
File Name src/main/webapp/vulnerability/Injection/xslt.jsp
Method <c:import url='${param["style"]}' var="xslt"/>

```
....
14.  <c:import url='${param["style"]}' var="xslt"/>
```

Reflected XSS All Clients\Path 7:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=24
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 11 of src/main/webapp/vulnerability/SendMessage.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""status"", which is read by the out.print method at line 11 of src/main/webapp/vulnerability/SendMessage.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/SendMessage.jsp	src/main/webapp/vulnerability/SendMessage.jsp
Line	11	11
Object	""status""	print

Code Snippet	
File Name	src/main/webapp/vulnerability/SendMessage.jsp
Method	out.print(request.getParameter("status")); //Displaying any error message

```
....
11.         out.print(request.getParameter("status")); //Displaying any
error message
```

Reflected XSS All Clients\Path 8:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=25
Status	New

The application's <td>Recipient: </td><td><input type="text" name="recipient" value="<% if(request.getParameter("recipient")!=null){ out.print(request.getParameter("recipient")); } %>"/></td> embeds untrusted data in the generated output with print, at line 18 of src/main/webapp/vulnerability/SendMessage.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""recipient"", which is read by the <td>Recipient: </td><td><input type="text" name="recipient" value="<% if(request.getParameter("recipient")!=null){ out.print(request.getParameter("recipient")); } %>"/></td> method at line 18 of src/main/webapp/vulnerability/SendMessage.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/SendMessage.jsp	src/main/webapp/vulnerability/SendMessage.jsp
Line	18	18
Object	""recipient""	print

Code Snippet	
File Name	src/main/webapp/vulnerability/SendMessage.jsp
Method	<tr><td>Recipient: </td><td><input type="text" name="recipient" value="<% if(request.getParameter("recipient")!=null){ out.print(request.getParameter("recipient")); } %>"/></td></tr>

```
....
18.  <tr><td>Recipient: </td><td><input type="text" name="recipient"
value="<% if(request.getParameter("recipient")!=null){
out.print(request.getParameter("recipient")); } %>"/></td></tr>
```

Reflected XSS All Clients\Path 9:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=26
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 23 of src/main/webapp/vulnerability/UserDetails.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""username"", which is read by the username=request.getParameter method at line 8 of src/main/webapp/vulnerability/UserDetails.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/UserDetails.jsp	src/main/webapp/vulnerability/UserDetails.jsp
Line	8	23
Object	""username""	print

Code Snippet

File Name src/main/webapp/vulnerability/UserDetails.jsp

Method String username=request.getParameter("username");

```
....
8.      String username=request.getParameter("username");
```

File Name src/main/webapp/vulnerability/UserDetails.jsp

Method out.print("Send Message to "+username+"");

```
....
23.             out.print("<a
href='SendMessage.jsp?recipient="+username+"'>Send Message to
"+username+"</a>");
```

Reflected XSS All Clients\Path 10:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=27
Status	New

The application's <%=searchedName%> embeds untrusted data in the generated output with searchedName, at line 21 of src/main/webapp/vulnerability/xss/search.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""keyword"", which is read by the request.getParameter method at line 16 of

src/main/webapp/vulnerability/xss/search.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/xss/search.jsp	src/main/webapp/vulnerability/xss/search.jsp
Line	16	21
Object	""keyword""	searchedName

Code Snippet
File Name src/main/webapp/vulnerability/xss/search.jsp
Method String searchedName = request.getParameter("keyword");

```
....
16.                String searchedName = request.getParameter("keyword");
```

File Name src/main/webapp/vulnerability/xss/search.jsp
Method Search Results for <%=searchedName%>

```
....
21.                        Search Results for <%=searchedName%>
```

Reflected XSS All Clients\Path 11:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=28
Status	New

The application's <%=keyword%> embeds untrusted data in the generated output with keyword, at line 15 of src/main/webapp/vulnerability/xss/xss4.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""keyword"", which is read by the <% String keyword = request.getParameter("keyword"); %> method at line 2 of src/main/webapp/vulnerability/xss/xss4.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/xss/xss4.jsp	src/main/webapp/vulnerability/xss/xss4.jsp
Line	2	15
Object	""keyword""	keyword

Code Snippet
File Name src/main/webapp/vulnerability/xss/xss4.jsp
Method <% String keyword = request.getParameter("keyword"); %>

```
....
2.    <% String keyword = request.getParameter("keyword"); %>
```

File Name src/main/webapp/vulnerability/xss/xss4.jsp
Method Search Results for <%=keyword%>

```
....
15.                     Search Results for <%=keyword%>
```

Reflected XSS All Clients\Path 12:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=29
Status	New

The application's <input type="text" name="keyword" value=<% if (keyword != null){ out.print(keyword);} %>> embeds untrusted data in the generated output with print, at line 6 of src/main/webapp/vulnerability/xss/xss4.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input ""keyword"", which is read by the <% String keyword = request.getParameter("keyword"); %> method at line 2 of src/main/webapp/vulnerability/xss/xss4.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/xss/xss4.jsp	src/main/webapp/vulnerability/xss/xss4.jsp
Line	2	6
Object	""keyword""	print

Code Snippet
File Name src/main/webapp/vulnerability/xss/xss4.jsp
Method <% String keyword = request.getParameter("keyword"); %>

```
....
2.    <% String keyword = request.getParameter("keyword"); %>
```

File Name src/main/webapp/vulnerability/xss/xss4.jsp

Method	<input type="text" name="keyword" value=<% if (keyword != null){ out.print(keyword);} %>>

```
....
6.          <input type="text" name="keyword" value=<% if (keyword !=
null){ out.print(keyword);} %>>
```

Reflected XSS All Clients\Path 13:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=30
Status	New

The application's <td>UserName: </td><td><input type="text" name="username" value="<%=username%>" /></td> embeds untrusted data in the generated output with username, at line 22 of src/main/webapp/login.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input getCookies, which is read by the request.getCookies method at line 7 of src/main/webapp/login.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/login.jsp	src/main/webapp/login.jsp
Line	7	22
Object	getCookies	username

Code Snippet	
File Name	src/main/webapp/login.jsp
Method	Cookie[] cookies = request.getCookies();

```
....
7.   Cookie[] cookies = request.getCookies();
```

File Name	src/main/webapp/login.jsp
Method	<tr><td>UserName: </td><td><input type="text" name="username" value="<%=username%>" /></td></tr>

```
....
22.      <tr><td>UserName: </td><td><input type="text" name="username"
value="<%=username%>" /></td></tr>
```

Reflected XSS All Clients\Path 14:

Severity	High
Result State	To Verify

Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=31
Status	New

The application's <td>Password :</td><td><input type="text" name="password" value="<%=password%>"/></td> embeds untrusted data in the generated output with password, at line 23 of src/main/webapp/login.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by simply providing modified data in the user input getCookies, which is read by the request.getCookies method at line 7 of src/main/webapp/login.jsp. This input then flows through the code straight to the output web page, without sanitization.

This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/login.jsp	src/main/webapp/login.jsp
Line	7	23
Object	getCookies	password

Code Snippet

File Name src/main/webapp/login.jsp

Method Cookie[] cookies = request.getCookies();

```
....
7.   Cookie[] cookies = request.getCookies();
```

File Name src/main/webapp/login.jsp

Method <tr><td>Password :</td><td><input type="text" name="password" value="<%=password%>"/></td></tr>

```
....
23.  <tr><td>Password :</td><td><input type="text" name="password"
value="<%=password%>"/></td></tr>
```

Second Order SQL Injection

Query Path:

Java\Cx\Java High Risk\Second Order SQL Injection Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
OWASP Top 10 2013: A1-Injection
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description

Second Order SQL Injection\Path 1:

Severity	High
Result State	To Verify

Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=8
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with BinaryExpr, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

The attacker may be able to write arbitrary data to the database, which is then retrieved by the application with rs in processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This data then flows through the code, until it is used directly in the SQL query without sanitization, and then submitted to the database server for execution.

This may enable a Second-Order SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/Messages.jsp
Line	52	14
Object	rs	BinaryExpr

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/vulnerability/Messages.jsp

Method rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.               rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

Second Order SQL Injection\Path 2:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=9
Status	New

The application's stmt.executeUpdate method executes an SQL query with BinaryExpr, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an

untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

The attacker may be able to write arbitrary data to the database, which is then retrieved by the application with rs in processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This data then flows through the code, until it is used directly in the SQL query without sanitization, and then submitted to the database server for execution.

This may enable a Second-Order SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/changeCardDetails.jsp
Line	52	43
Object	rs	BinaryExpr

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

Second Order SQL Injection\Path 3:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=10
Status	New

The application's stmt.executeUpdate method executes an SQL query with BinaryExpr, at line 31 of src/main/webapp/vulnerability/csrf/change-info.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

The attacker may be able to write arbitrary data to the database, which is then retrieved by the application with rs in processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This data then flows through the code, until it is used directly in the SQL query without sanitization, and then submitted to the database server for execution.

This may enable a Second-Order SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	52	31
Object	rs	BinaryExpr

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp

Method stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.             stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

Second Order SQL Injection\Path 4:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=11
Status	New

The application's stmt.executeUpdate method executes an SQL query with BinaryExpr, at line 40 of src/main/webapp/vulnerability/csrf/changepassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

The attacker may be able to write arbitrary data to the database, which is then retrieved by the application with rs in processRequest method at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This data then flows through the code, until it is used directly in the SQL query without sanitization, and then submitted to the database server for execution.

This may enable a Second-Order SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	52	40

Object	rs	BinaryExpr

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp

Method stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.                 stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

Second Order SQL Injection\Path 5:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=12
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with BinaryExpr, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

The attacker may be able to write arbitrary data to the database, which is then retrieved by the application with rs in rs=stmt.executeQuery method at line 19 of src/main/webapp/admin/adminlogin.jsp. This data then flows through the code, until it is used directly in the SQL query without sanitization, and then submitted to the database server for execution.

This may enable a Second-Order SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/vulnerability/Messages.jsp
Line	19	14
Object	rs	BinaryExpr

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

File Name	src/main/webapp/vulnerability/Messages.jsp
Method	rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.               rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

Second Order SQL Injection\Path 6:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=13
Status	New

The application's stmt.executeUpdate method executes an SQL query with BinaryExpr, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

The attacker may be able to write arbitrary data to the database, which is then retrieved by the application with rs in rs=stmt.executeQuery method at line 19 of src/main/webapp/admin/adminlogin.jsp. This data then flows through the code, until it is used directly in the SQL query without sanitization, and then submitted to the database server for execution.

This may enable a Second-Order SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/changeCardDetails.jsp
Line	19	43
Object	rs	BinaryExpr

Code Snippet

File Name	src/main/webapp/admin/adminlogin.jsp
Method	rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

File Name	src/main/webapp/changeCardDetails.jsp
Method	stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

Second Order SQL Injection\Path 7:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=14
Status	New

The application's stmt.executeUpdate method executes an SQL query with BinaryExpr, at line 31 of src/main/webapp/vulnerability/csrf/change-info.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

The attacker may be able to write arbitrary data to the database, which is then retrieved by the application with rs in rs=stmt.executeQuery method at line 19 of src/main/webapp/admin/adminlogin.jsp. This data then flows through the code, until it is used directly in the SQL query without sanitization, and then submitted to the database server for execution.

This may enable a Second-Order SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	19	31
Object	rs	BinaryExpr

Code Snippet

File Name	src/main/webapp/admin/adminlogin.jsp
Method	rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                  rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

File Name	src/main/webapp/vulnerability/csrf/change-info.jsp
Method	stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.             stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

Second Order SQL Injection\Path 8:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=15
Status	New

The application's stmt.executeUpdate method executes an SQL query with BinaryExpr, at line 40 of src/main/webapp/vulnerability/csrf/changepassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

The attacker may be able to write arbitrary data to the database, which is then retrieved by the application with rs in rs=stmt.executeQuery method at line 19 of src/main/webapp/admin/adminlogin.jsp. This data then flows through the code, until it is used directly in the SQL query without sanitization, and then submitted to the database server for execution.

This may enable a Second-Order SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	19	40
Object	rs	BinaryExpr

Code Snippet

File Name src/main/webapp/admin/adminlogin.jsp

Method rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                 rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp

Method stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.                 stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

Connection String Injection

Query Path:

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
OWASP Top 10 2013: A1-Injection
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description

Connection String Injection\Path 1:

Severity	High
Result State	Not Exploitable
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=1
Status	New

The application's setup method receives untrusted, user-controlled data, and uses this data to connect to a database using getConnection, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This may enable a Connection String Injection attack.

The attacker can inject the connection string via user input, ""dburl"", which is retrieved by the application in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	112
Object	""dburl""	getConnection

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Connection String Injection\Path 2:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=2
Status	New

The application's setup method receives untrusted, user-controlled data, and uses this data to connect to a database using getConnection, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This may enable a Connection String Injection attack.

The attacker can inject the connection string via user input, ""dbpass"", which is retrieved by the application in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	112
Object	""dbpass""	getConnection

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
112.                      Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Connection String Injection\Path 3:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=3
Status	New

The application's setup method receives untrusted, user-controlled data, and uses this data to connect to a database using getConnection, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This may enable a Connection String Injection attack.

The attacker can inject the connection string via user input, ""dbuser"", which is retrieved by the application in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Line	56	112
Object	""dbuser""	getConnection

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Connection String Injection\Path 4:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=4
Status	New

The application's setup method receives untrusted, user-controlled data, and uses this data to connect to a database using getConnection, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This may enable a Connection String Injection attack.

The attacker can inject the connection string via user input, ""dburl"", which is retrieved by the application in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	121
Object	""dburl""	getConnection

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Connection String Injection\Path 5:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=5
Status	New

The application's setup method receives untrusted, user-controlled data, and uses this data to connect to a database using getConnection, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This may enable a Connection String Injection attack.

The attacker can inject the connection string via user input, ""dbname"", which is retrieved by the application in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	121
Object	""dbname""	getConnection

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Connection String Injection\Path 6:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=6

Status	New

The application's setup method receives untrusted, user-controlled data, and uses this data to connect to a database using getConnection, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This may enable a Connection String Injection attack.

The attacker can inject the connection string via user input, ""dbpass"", which is retrieved by the application in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	121
Object	""dbpass""	getConnection

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Connection String Injection\Path 7:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=7
Status	New

The application's setup method receives untrusted, user-controlled data, and uses this data to connect to a database using getConnection, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. This may enable a Connection String Injection attack.

The attacker can inject the connection string via user input, ""dbuser"", which is retrieved by the application in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	121

Object	""dbuser""	getConnection

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

XPath Injection

Query Path:
Java\Cx\Java High Risk\XPath Injection Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
OWASP Top 10 2013: A1-Injection
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description
XPath Injection\Path 1:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=16
Status	New

The application's processRequest method constructs an XPath query, for navigating an XML document. The XPath query is created with compile, at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java, using an untrusted string embedded in the expression.

This may enable an attacker to modify the XPath expression, leading to an XPath Injection attack.

The attacker may be able to inject the modified XPath expression via user input, ""username"", which is retrieved by the application in the processRequest method, at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java. This value then flows through the code to compile, as noted.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/co	src/main/java/org/cysecurity/cspf/jvl/co

	ntroller/XPathQuery.java	ntroller/XPathQuery.java
Line	35	53
Object	""username""	compile

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
35.             String user=request.getParameter("username");
....
53.             String name=xPath.compile(xPression).evaluate(xDoc);
```

XPath Injection\Path 2:

Severity	High
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=17
Status	New

The application's processRequest method constructs an XPath query, for navigating an XML document. The XPath query is created with compile, at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java, using an untrusted string embedded in the expression.

This may enable an attacker to modify the XPath expression, leading to an XPath Injection attack.

The attacker may be able to inject the modified XPath expression via user input, ""password"", which is retrieved by the application in the processRequest method, at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java. This value then flows through the code to compile, as noted.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	36	53
Object	""password""	compile

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             String pass=request.getParameter("password");
....
53.             String name=xPath.compile(xPression).evaluate(xDoc);
```

XSRF

Query Path:
Java\Cx\Java Medium Threat\XSRF Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.9 - Cross-site request forgery
OWASP Top 10 2013: A8-Cross-Site Request Forgery (CSRF)
NIST SP 800-53: SC-23 Session Authenticity (P1)

Description

XSRF\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=368
Status	New

Method info=request.getParameter at line 26 of src/main/webapp/vulnerability/csrf/change-info.jsp gets a parameter from a user request from ""info"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/webapp/vulnerability/csrf/change-info.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	26	31
Object	""info""	executeUpdate

Code Snippet

File Name src/main/webapp/vulnerability/csrf/change-info.jsp

Method String info=request.getParameter("info");

```
....
26.     String info=request.getParameter("info");
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp

Method stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.             stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

XSRF\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=369
Status	New

Method user=request.getParameter at line 11 of src/main/webapp/admin/adminlogin.jsp gets a parameter from a user request from ""username"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	11	31
Object	""username""	executeUpdate

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp
Method stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.             stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

XSRF\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=370
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets a parameter from a user request from ""username"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	43	31
Object	""username""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp

Method stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.             stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

XSRF\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=371
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets a parameter from a user request from ""password"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	44	31
Object	""password""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp

Method stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.             stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

XSRF\Path 5:

Severity	Medium

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=372
Status	New

Method pass=request.getParameter at line 33 of src/main/webapp/vulnerability/csrf/changepassword.jsp gets a parameter from a user request from ""password"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	33	40
Object	""password""	executeUpdate

Code Snippet

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp

Method String pass=request.getParameter("password");

```
....
33.          String pass=request.getParameter("password");
```

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp

Method stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.                stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

XSRF\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=373
Status	New

Method user=request.getParameter at line 11 of src/main/webapp/admin/adminlogin.jsp gets a parameter from a user request from ""username"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	11	40
Object	""username""	executeUpdate

Code Snippet	
File Name	src/main/webapp/admin/adminlogin.jsp
Method	String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name	src/main/webapp/vulnerability/csrf/changepassword.jsp
Method	stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.               stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

XSRF\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=374
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets a parameter from a user request from ""username"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	43	40
Object	""username""	executeUpdate

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
```

File Name	src/main/webapp/vulnerability/csrf/changepassword.jsp
Method	stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.                stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

XSRF\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=375
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets a parameter from a user request from ""password"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	44	40
Object	""password""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password").trim();
```

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp

Method stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.                stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

XSRF\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=376
Status	New

Method email=request.getParameter at line 27 of src/main/webapp/vulnerability/idor/change-email.jsp gets a parameter from a user request from ""email"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/webapp/vulnerability/idor/change-email.jsp	src/main/webapp/vulnerability/idor/change-email.jsp
Line	27	32
Object	""email""	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/idor/change-email.jsp
Method String email=request.getParameter("email");

```
....
27.     String email=request.getParameter("email");
```

File Name src/main/webapp/vulnerability/idor/change-email.jsp
Method stmt.executeUpdate("Update users set email='"+email+"' where id="+id);

```
....
32.             stmt.executeUpdate("Update users set email='"+email+"'
where id="+id);
```

XSRF\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=377
Status	New

Method id=request.getParameter at line 28 of src/main/webapp/vulnerability/idor/change-email.jsp gets a parameter from a user request from ""id"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/webapp/vulnerability/idor/change-email.jsp	src/main/webapp/vulnerability/idor/change-email.jsp
Line	28	32
Object	""id""	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/idor/change-email.jsp
Method String id=request.getParameter("id");

```
....
28.     String id=request.getParameter("id");
```

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method stmt.executeUpdate("Update users set email='"+email+"' where id="+id);

```
....
32.             stmt.executeUpdate("Update users set email='"+email+"'
where id="+id);
```

XSRF\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=378
Status	New

Method user=request.getParameter at line 13 of src/main/webapp/admin/manageusers.jsp gets a parameter from a user request from ""user"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	13	14
Object	""user""	executeUpdate

Code Snippet

File Name src/main/webapp/admin/manageusers.jsp

Method String user=request.getParameter("user");

```
....
13.      String user=request.getParameter("user");
```

File Name src/main/webapp/admin/manageusers.jsp

Method stmt.executeUpdate("Delete from users where username='"+user+"'");

```
....
14.      stmt.executeUpdate("Delete from users where
username='"+user+"'");
```

XSRF\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=379
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""adminuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	60	127
Object	""adminuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.          adminuser= request.getParameter("adminuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

XSRF\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=380
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	127

Object	""dburl""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

XSRF\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=381
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	127
Object	""dbname""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

XSRF\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=382
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	127
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

XSRF\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=383

Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	127
Object	""dbuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

XSRF\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=384
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	128
Object	""dburl""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
128.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('victim','victim','victim@localhost','I am the victim of this
application','default.jpg','user',1,'max')");
```

XSRF\Path 18:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=385
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	128
Object	""dbname""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
128.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('victim','victim','victim@localhost','I am the victim of this
application','default.jpg','user',1,'max')");
```

XSRF\Path 19:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=386
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	128
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
128.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('victim','victim','victim@localhost','I am the victim of this
application','default.jpg','user',1,'max')");
```

XSRF\Path 20:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=387

Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	128
Object	""dbuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
128.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('victim','victim','victim@localhost','I am the victim of this
application','default.jpg','user',1,'max')");
```

XSRF\Path 21:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=388
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	129
Object	""dburl""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
129.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('attacker','attacker','attacker@localhost','I am the attacker of this
application','default.jpg','user',1,'bella')");
```

XSRF\Path 22:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=389
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	129
Object	""dbname""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
129.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('attacker','attacker','attacker@localhost','I am the attacker of this
application','default.jpg','user',1,'bella')");
```

XSRF\Path 23:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=390
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	129
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
129.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('attacker','attacker','attacker@localhost','I am the attacker of this
application','default.jpg','user',1,'bella')");
```

XSRF\Path 24:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=391

Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	129
Object	""dbuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
129.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('attacker','attacker','attacker@localhost','I am the attacker of this
application','default.jpg','user',1,'bella')");
```

XSRF\Path 25:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=392
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	130
Object	""dburl""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
130.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values ('NEO','trinity','neo@matrix','I
am the NEO','default.jpg','user',1,'sentinel')");
```

XSRF\Path 26:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=393
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	130
Object	""dbname""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
130.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values ('NEO','trinity','neo@matrix','I
am the NEO','default.jpg','user',1,'sentinel')");
```

XSRF\Path 27:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=394
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	130
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
130.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values ('NEO','trinity','neo@matrix','I
am the NEO','default.jpg','user',1,'sentinel')");
```

XSRF\Path 28:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=395
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	130
Object	""dbuser""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
130.                                  stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values ('NEO','trinity','neo@matrix','I
am the NEO','default.jpg','user',1,'sentinel')");
```

XSRF\Path 29:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=396
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	131
Object	""dburl""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
131.                                  stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('trinity','NEO','trinity@matrix','it is
Trinity','default.jpg','user',1,'sentinel')");
```

XSRF\Path 30:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=397
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	131
Object	""dbname""	executeUpdate

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
131.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('trinity','NEO','trinity@matrix','it is
Trinity','default.jpg','user',1,'sentinel')");
```

XSRF\Path 31:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=398
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	131
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
131.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('trinity','NEO','trinity@matrix','it is
Trinity','default.jpg','user',1,'sentinel')");
```

XSRF\Path 32:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=399

Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	131
Object	""dbuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
131.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('trinity','NEO','trinity@matrix','it is
Trinity','default.jpg','user',1,'sentinel')");
```

XSRF\Path 33:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=400
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	132
Object	""dburl""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
132.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('Anderson','java','anderson@1999','I am computer
programmer','default.jpg','user',1,'C++')");
```

XSRF\Path 34:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=401
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	132
Object	""dbname""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
132.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('Anderson','java','anderson@1999','I am computer
programmer','default.jpg','user',1,'C++')");
```

XSRF\Path 35:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=402
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	132
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
132.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('Anderson','java','anderson@1999','I am computer
programmer','default.jpg','user',1,'C++')");
```

XSRF\Path 36:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=403

Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	132
Object	""dbuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
132.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('Anderson','java','anderson@1999','I am computer
programmer','default.jpg','user',1,'C++')");
```

XSRF\Path 37:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=404
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	136
Object	""dburl""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
136.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Feel free to ask any questions about
Java Vulnerable Lab','First Post', 'admin')");
```

XSRF\Path 38:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=405
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	136
Object	""dbname""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
136.                                stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Feel free to ask any questions about
Java Vulnerable Lab','First Post', 'admin')");
```

XSRF\Path 39:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=406
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	136
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
136.                                stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Feel free to ask any questions about
Java Vulnerable Lab','First Post', 'admin')");
```

XSRF\Path 40:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=407
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	136
Object	""dbuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.           dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
136.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Feel free to ask any questions about
Java Vulnerable Lab','First Post', 'admin')");
```

XSRF\Path 41:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=408
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	137
Object	""dburl""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
137.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello Guys, this is victim','Second
Post', 'victim')");
```

XSRF\Path 42:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=409
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	137
Object	""dbname""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
137.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello Guys, this is victim','Second
Post', 'victim')");
```

XSRF\Path 43:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=410
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	137
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.           dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
137.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello Guys, this is victim','Second
Post', 'victim')");
```

XSRF\Path 44:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=411
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	137

Object	""dbuser""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
137.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello Guys, this is victim','Second
Post', 'victim')");
```

XSRF\Path 45:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=412
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	138
Object	""dburl""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
138.                                    stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello This is attacker','Third
Post', 'attacker')");
```

XSRF\Path 46:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=413
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	138
Object	""dbname""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
138.                                    stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello This is attacker','Third
Post', 'attacker')");
```

XSRF\Path 47:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=414
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbpass"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	138
Object	""dbpass""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.           dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
138.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello This is attacker','Third
Post', 'attacker')");
```

XSRF\Path 48:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=415
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbuser"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	138
Object	""dbuser""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
138.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello This is attacker','Third
Post', 'attacker')");
```

XSRF\Path 49:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=416
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dburl"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	139
Object	""dburl""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
139.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Trinity! Help!','Help','neo')");
```

XSRF\Path 50:

Severity	Medium

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=417
Status	New

Method processRequest at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java gets a parameter from a user request from ""dbname"". This parameter value flows through the code and is eventually used to access application state altering functionality. This may enable Cross-Site Request Forgery (XSRF).

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	139
Object	""dbname""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
139.                                  stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Trinity! Help!','Help','neo')");
```

Trust Boundary Violation

Query Path:
Java\Cx\Java Medium Threat\Trust Boundary Violation Version:1

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A5-Broken Access Control

Description

Trust Boundary Violation\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=346
Status	New

Method user=request.getParameter at line 11 of src/main/webapp/admin/adminlogin.jsp gets user input from element ""username"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in session.setAttribute at line 23 of src/main/webapp/admin/adminlogin.jsp. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	11	23
Object	""username""	getString

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name src/main/webapp/admin/adminlogin.jsp
Method session.setAttribute("user", rs.getString("username"));

```
....
23.                                   session.setAttribute("user",
rs.getString("username"));
```

Trust Boundary Violation\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=347
Status	New

Method user=request.getParameter at line 11 of src/main/webapp/admin/adminlogin.jsp gets user input from element ""username"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in session.setAttribute at line 24 of src/main/webapp/admin/adminlogin.jsp. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	11	24
Object	""username""	getString

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name src/main/webapp/admin/adminlogin.jsp

Method session.setAttribute("avatar", rs.getString("avatar"));

```
....
24.                                       session.setAttribute("avatar",
rs.getString("avatar"));
```

Trust Boundary Violation\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=348
Status	New

Method user=request.getParameter at line 11 of src/main/webapp/admin/adminlogin.jsp gets user input from element ""username"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in session.setAttribute at line 25 of src/main/webapp/admin/adminlogin.jsp. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	11	25
Object	""username""	getString

Code Snippet

File Name src/main/webapp/admin/adminlogin.jsp

Method String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name src/main/webapp/admin/adminlogin.jsp

Method session.setAttribute("privilege", rs.getString("privilege"));

```
....
25.
session.setAttribute("privilege", rs.getString("privilege"));
```

Trust Boundary Violation\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=349
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element ""password"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	57
Object	""password""	getString

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
....
57.                                         session.setAttribute("user",
rs.getString("username"));
```

Trust Boundary Violation\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=350
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element ""username"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	57
Object	""username""	getString

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
57.                                         session.setAttribute("user",
rs.getString("username"));
```

Trust Boundary Violation\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=351
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element ""password"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	58
Object	""password""	getString

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password").trim();
....
58.                                         session.setAttribute("avatar",
rs.getString("avatar"));
```

Trust Boundary Violation\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=352
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element ""username"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of

src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	58
Object	""username""	getString

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
58.                                         session.setAttribute("avatar",
rs.getString("avatar"));
```

Trust Boundary Violation\Path 8:

Severity Medium
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=353
Status New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element ""password"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	56
Object	""password""	getString

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password").trim();
....
56.                                         session.setAttribute("userid",
rs.getString("id"));
```

Trust Boundary Violation\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=354
Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java gets user input from element ""password"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	36	63
Object	""password""	name

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.              String pass=request.getParameter("password");
....
63.                    session.setAttribute("user", name);
```

Trust Boundary Violation\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=355
Status	New

Method user=request.getParameter at line 11 of src/main/webapp/admin/adminlogin.jsp gets user input from element ""username"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in session.setAttribute at line 22 of src/main/webapp/admin/adminlogin.jsp. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	11	22
Object	""username""	getString

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp

Method	String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name	src/main/webapp/admin/adminlogin.jsp
Method	session.setAttribute("userid", rs.getString("id"));

```
....
22.                                       session.setAttribute("userid",
rs.getString("id"));
```

Trust Boundary Violation\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=356
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element ""username"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	56
Object	""username""	getString

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
56.                                       session.setAttribute("userid",
rs.getString("id"));
```

Trust Boundary Violation\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=357
Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java gets user input from element ""username"". This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	35	63
Object	""username""	name

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
35.              String user=request.getParameter("username");
....
63.                    session.setAttribute("user", name);
```

HTTP Response Splitting

Query Path:
Java\Cx\Java Medium Threat\HTTP Response Splitting Version:0

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description
HTTP Response Splitting\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=311
Status	New

Method request.getParameter at line 11 of src/main/webapp/vulnerability/idor/download.jsp gets user input from the ""file"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in response.setHeader at line 19 of src/main/webapp/vulnerability/idor/download.jsp. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	11	19

Object	""file""	setHeader

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method filePath = request.getParameter("file");

```
....
11.             filePath = request.getParameter("file");
```

File Name src/main/webapp/vulnerability/idor/download.jsp

Method response.setHeader("Content-Disposition", "attachment; filename=\"" + fileName + "\"");

```
....
19.             response.setHeader("Content-Disposition", "attachment;
filename=\"" + fileName + "\"");
```

HTTP Response Splitting\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=312
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id_union.jsp gets user input from the ""fileid"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in response.setHeader at line 38 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	18	38
Object	""fileid""	setHeader

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method response.setHeader("Content-Disposition", "attachment; filename=\"" + fileName + "\"");

```
....
38.                  response.setHeader("Content-Disposition",
"attachment; filename=\"" + fileName + "\"");
```

HTTP Response Splitting\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=313
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from the ""username"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	65
Object	""username""	addCookie

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.          String user=request.getParameter("username").trim();
....
65.
response.addCookie(username);
```

HTTP Response Splitting\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=314
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from the ""password"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	66
Object	""password""	addCookie

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
....
66.
response.addCookie(password);
```

HTTP Response Splitting\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=315
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from the ""username"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	66
Object	""username""	addCookie

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.          String user=request.getParameter("username").trim();
....
66.
response.addCookie(password);
```

HTTP Response Splitting\Path 6:

Severity	Medium
Result State	To Verify

Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=316
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from the ""password"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	68
Object	""password""	sendRedirect

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
....
68.
response.sendRedirect(response.encodeURL("ForwardMe?location=/index.jsp"
));
```

HTTP Response Splitting\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=317
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from the ""username"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	68
Object	""username""	sendRedirect

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
68.
response.sendRedirect(response.encodeURL("ForwardMe?location=/index.jsp"
));
```

HTTP Response Splitting\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=318
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from the ""password"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	79
Object	""password""	sendRedirect

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
....
79.
response.sendRedirect("login.jsp?err=something went wrong");
```

HTTP Response Splitting\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=319
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from the ""username"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in processRequest at line

39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	79
Object	""username""	sendRedirect

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
79.
response.sendRedirect("login.jsp?err=something went wrong");
```

HTTP Response Splitting\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=320
Status	New

Method processRequest at line 31 of src/main/java/org/cysecurity/cspf/jvl/controller/Open.java gets user input from the ""url"" element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in processRequest at line 31 of src/main/java/org/cysecurity/cspf/jvl/controller/Open.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Line	36	39
Object	""url""	sendRedirect

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             String url=request.getParameter("url");
....
39.                response.sendRedirect(url);
```

Cross Site History Manipulation

Query Path:
Java\Cx\Java Medium Threat\Cross Site History Manipulation Version:1

Categories

OWASP Top 10 2013: A8-Cross-Site Request Forgery (CSRF)

Description

Cross Site History Manipulation\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=82
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	53	53
Object	if	if

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
53.                                     if(rs != null && rs.next()){
```

Cross Site History Manipulation\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=83
Status	New

Method processRequest at line 31 of src/main/java/org/cysecurity/cspf/jvl/controller/Open.java may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Line	37	37

Object	if	if

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
37.             if(url!=null)
```

Cross Site History Manipulation\Path 3:

Severity Medium
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=84
Status New

Method processRequest at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	54	54
Object	if	if

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.                     if(con!=null && !con.isClosed())
```

Cross Site History Manipulation\Path 4:

Severity Medium
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=85
Status New

Method processRequest at line 34 of src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java
Line	46	46
Object	if	if

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
46.                     if(con!=null && !con.isClosed() &&
request.getParameter("send")!=null)
```

Cross Site History Manipulation\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=86
Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	55	55
Object	if	if

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
55.             if(name.isEmpty())
```

Cross Site History Manipulation\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=87
Status	New

Method if at line 20 of src/main/webapp/admin/adminlogin.jsp may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	20	20
Object	if	if

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method if(rs != null && rs.next()){

```
....
20.                                     if(rs != null && rs.next()){
```

Cross Site History Manipulation\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=88
Status	New

Method if at line 8 of src/main/webapp/admin/adminlogin.jsp may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	8	8
Object	if	if

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method if(request.getParameter("Login")!=null)

```
....
8.  if(request.getParameter("Login")!=null)
```

Cross Site History Manipulation\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=89
Status	New

Method if at line 3 of src/main/webapp/admin/index.jsp may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/webapp/admin/index.jsp	src/main/webapp/admin/index.jsp

Line	3	3
Object	if	if

Code Snippet
File Name src/main/webapp/admin/index.jsp
Method if(session.getAttribute("privilege")!=null &&
session.getAttribute("privilege").equals("admin"))

```
....
3.   if(session.getAttribute("privilege")!=null &&
session.getAttribute("privilege").equals("admin"))
```

Cross Site History Manipulation\Path 9:

Severity Medium
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=90
Status New

Method processRequest at line 32 of src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java may leak server-side conditional values, enabling user tracking from another website. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java	src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java
Line	37	37
Object	if	if

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
37.             if(request.getParameter("location")!=null)
```

External Control of System or Config Setting

Query Path:
Java\Cx\Java Medium Threat\External Control of System or Config Setting Version:1

Categories

OWASP Top 10 2013: A7-Missing Function Level Access Control
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A5-Broken Access Control

Description

External Control of System or Config Setting\Path 1:

Severity Medium

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=297
Status	New

The application sets environment configuration settings setProperty, in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49, providing values received from the user input @Source Element, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	66
Object	""dburl""	setProperty

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.           dburl = request.getParameter("dburl");
....
66.            config.setProperty("dburl",dburl);
```

External Control of System or Config Setting\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=298
Status	New

The application sets environment configuration settings setProperty, in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49, providing values received from the user input @Source Element, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	55	67
Object	""jdbcdriver""	setProperty

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
55.          jdbcdriver = request.getParameter("jdbcdriver");
....
67.           config.setProperty("jdbcdriver",jdbcdriver);
```

External Control of System or Config Setting\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=299
Status	New

The application sets environment configuration settings setProperty, in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49, providing values received from the user input @Source Element, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	68
Object	""dbuser""	setProperty

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
....
68.           config.setProperty("dbuser",dbuser);
```

External Control of System or Config Setting\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=300
Status	New

The application sets environment configuration settings setProperty, in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49, providing values received from the user input @Source Element, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	69

Object	""dbpass""	setProperty

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
....
69.           config.setProperty("dbpass",dbpass);
```

External Control of System or Config Setting\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=301
Status	New

The application sets environment configuration settings setProperty, in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49, providing values received from the user input @Source Element, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	70
Object	""dbname""	setProperty

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
....
70.           config.setProperty("dbname",dbname);
```

External Control of System or Config Setting\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=302
Status	New

The application sets environment configuration settings setProperty, in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49, providing values received from the user input @Source Element, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	59	71
Object	""siteTitle""	setProperty

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.          siteTitle= request.getParameter("siteTitle");
....
71.           config.setProperty("siteTitle",siteTitle);
```

External Control of System or Config Setting\Path 7:

Severity Medium
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=303
Status New

The application sets environment configuration settings setProperty, in src/main/webapp/admin/Configure.jsp at line 21, providing values received from the user input @Source Element, at line 21 of src/main/webapp/admin/Configure.jsp.

	Source	Destination
File	src/main/webapp/admin/Configure.jsp	src/main/webapp/admin/Configure.jsp
Line	21	21
Object	""siteTitle""	setProperty

Code Snippet
File Name src/main/webapp/admin/Configure.jsp
Method props.setProperty("siteTitle",request.getParameter("siteTitle"));

```
....
21.
props.setProperty("siteTitle",request.getParameter("siteTitle"));
```

External Control of System or Config Setting\Path 8:

Severity Medium
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=304
Status New

The application sets environment configuration settings setProperty, in src/main/webapp/vulnerability/baasm/SiteTitle.jsp at line 32, providing values received from the user input @Source Element, at line 32 of src/main/webapp/vulnerability/baasm/SiteTitle.jsp.

	Source	Destination
File	src/main/webapp/vulnerability/baasm/SiteTitle.jsp	src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Line	32	32
Object	""siteTitle""	setProperty

Code Snippet
File Name src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Method props.setProperty("siteTitle",request.getParameter("siteTitle"));

```
....
32.
props.setProperty("siteTitle",request.getParameter("siteTitle"));
```

HttpOnlyCookies

Query Path:
Java\Cx\Java Medium Threat\HttpOnlyCookies Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)
OWASP Top 10 2013: A3-Cross-Site Scripting (XSS)
OWASP Top 10 2017: A7-Cross-Site Scripting (XSS)

Description
HttpOnlyCookies\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=322
Status	New

The web application's processRequest method creates a cookie privilege, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java, and returns it in the response. However, the application is not configured to automatically set the cookie with the "httpOnly" attribute, and the code does not explicitly add this to the cookie.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	59	60
Object	privilege	addCookie

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.                                         Cookie privilege=new
Cookie("privilege","user");
60.                                         response.addCookie(privilege);
```

HttpOnlyCookies\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=323
Status	New

The web application's processRequest method creates a cookie username, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java, and returns it in the response. However, the application is not configured to automatically set the cookie with the "httpOnly" attribute, and the code does not explicitly add this to the cookie.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	63	65
Object	username	addCookie

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
63.                                         Cookie username=new
Cookie("username",user);
....
65.
response.addCookie(username);
```

HttpOnlyCookies\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=324
Status	New

The web application's processRequest method creates a cookie privilege, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java, and returns it in the response. However, the application is not configured to automatically set the cookie with the "httpOnly" attribute, and the code does not explicitly add this to the cookie.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	59	65
Object	privilege	addCookie

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.                                         Cookie privilege=new
Cookie("privilege","user");
....
65.
response.addCookie(username);
```

HttpOnlyCookies\Path 4:

Severity Medium
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=325
Status New

The web application's processRequest method creates a cookie password, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java, and returns it in the response. However, the application is not configured to automatically set the cookie with the "httpOnly" attribute, and the code does not explicitly add this to the cookie.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	64	66
Object	password	addCookie

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
64.                                         Cookie password=new
Cookie("password",pass);
....
66.
response.addCookie(password);
```

HttpOnlyCookies\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=326
Status	New

The web application's processRequest method creates a cookie privilege, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java, and returns it in the response. However, the application is not configured to automatically set the cookie with the "httpOnly" attribute, and the code does not explicitly add this to the cookie.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	59	66
Object	privilege	addCookie

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.                                          Cookie privilege=new
Cookie("privilege","user");
....
66.
response.addCookie(password);
```

HttpOnlyCookies\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=327
Status	New

The web application's processRequest method creates a cookie username, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java, and returns it in the response. However, the application is not configured to automatically set the cookie with the "httpOnly" attribute, and the code does not explicitly add this to the cookie.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	63	66
Object	username	addCookie

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
63.                                          Cookie username=new
Cookie("username",user);
....
66.
response.addCookie(password);
```

HttpOnlyCookies\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=328
Status	New

The web application's Cookie method creates a cookie privilege, at line 27 of src/main/webapp/admin/adminlogin.jsp, and returns it in the response. However, the application is not configured to automatically set the cookie with the "httpOnly" attribute, and the code does not explicitly add this to the cookie.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	27	29
Object	privilege	addCookie

Code Snippet

File Name	src/main/webapp/admin/adminlogin.jsp
Method	Cookie privilege=new Cookie("privilege","admin");

```
....
27.                                     Cookie privilege=new
Cookie("privilege","admin");
```

File Name	src/main/webapp/admin/adminlogin.jsp
Method	response.addCookie(privilege);

```
....
29.                                     response.addCookie(privilege);
```

Input Path Not Canonicalized

Query Path:
Java\Cx\Java Medium Threat\Input Path Not Canonicalized Version:1

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Mobile Top 10 2016: M7-Client Code Quality

Description

Input Path Not Canonicalized\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=330
Status	New

Method processRequest at line 34 of src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java gets dynamic data from the ""filename"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in processRequest at line 34 of src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	39	52
Object	""filename""	FileWriter

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.             String fileName=request.getParameter("filename");
....
52.                         BufferedWriter bw=new BufferedWriter(new
FileWriter(f.getAbsoluteFile()));
```

Input Path Not Canonicalized\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=331
Status	New

Method request.getParameter at line 11 of src/main/webapp/vulnerability/idor/download.jsp gets dynamic data from the ""file"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in = at line 18 of src/main/webapp/vulnerability/idor/download.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	11	18
Object	""file""	File

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method filePath = request.getParameter("file");

```
....
11.             filePath = request.getParameter("file");
```

File Name src/main/webapp/vulnerability/idor/download.jsp

Method String fileName = (new File(filePath)).getName();

```
....
18.             String fileName = (new File(filePath)).getName();
```

Input Path Not Canonicalized\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=332
Status	New

Method request.getParameter at line 11 of src/main/webapp/vulnerability/idor/download.jsp gets dynamic data from the ""file"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in DataInputStream at line 22 of src/main/webapp/vulnerability/idor/download.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	11	22
Object	""file""	FileInputStream

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method filePath = request.getParameter("file");

```
....
11.             filePath = request.getParameter("file");
```

File Name src/main/webapp/vulnerability/idor/download.jsp

Method DataInputStream in = new DataInputStream(new FileInputStream(file));

```
....
22.             DataInputStream in = new DataInputStream(new
FileInputStream(file));
```

Input Path Not Canonicalized\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=333
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id.jsp gets dynamic data from the ""fileid"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in = at line 37 of src/main/webapp/vulnerability/sqli/download_id.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	18	37
Object	""fileid""	File

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method String fileName = (new File(filePath)).getName();

```
....
37.                  String fileName = (new File(filePath)).getName();
```

Input Path Not Canonicalized\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=334
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id.jsp gets dynamic data from the ""fileid"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in DataInputStream at line 41 of src/main/webapp/vulnerability/sqli/download_id.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	18	41

Object	""fileid""	FileInputStream

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method DataInputStream in = new DataInputStream(new FileInputStream(file));

```
....
41.                  DataInputStream in = new DataInputStream(new
FileInputStream(file));
```

Input Path Not Canonicalized\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=335
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id_union.jsp gets dynamic data from the ""fileid"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in = at line 37 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	18	37
Object	""fileid""	File

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method String fileName = (new File(filePath)).getName();

```
....
37.                 String fileName = (new File(filePath)).getName();
```

Input Path Not Canonicalized\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=336
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id_union.jsp gets dynamic data from the ""fileid"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in DataInputStream at line 41 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	18	41
Object	""fileid""	FileInputStream

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method DataInputStream in = new DataInputStream(new FileInputStream(file));

```
....
41.                 DataInputStream in = new DataInputStream(new
FileInputStream(file));
```

Privacy Violation

Query Path:

Java\Cx\Java Medium Threat\Privacy Violation Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Identification And Authentication
NIST SP 800-53: SC-4 Information in Shared Resources (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description

Privacy Violation\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=338
Status	New

Method c.getValue at line 15 of src/main/webapp/login.jsp sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/webapp/login.jsp	src/main/webapp/login.jsp
Line	15	23
Object	password	password

Code Snippet

File Name src/main/webapp/login.jsp

Method password= c.getValue();

```
....
15.             password= c.getValue();
```

File Name src/main/webapp/login.jsp

Method <tr><td>Password :</td><td><input type="text" name="password" value="<%=password%>"/></td></tr>

```
....
23.  <tr><td>Password :</td><td><input type="text" name="password"
value="<%=password%>"/></td></tr>
```

Privacy Violation\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=339
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	64	66
Object	pass	password

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
64.                                                 Cookie password=new
Cookie("password",pass);
....
66.
response.addCookie(password);
```

Privacy Violation\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=340
Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/webapp/index.jsp
Line	50	5
Object	pass	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
50.              String xPression="/users/user[username='"+user+"' and
password='"+pass+"']/name";
```

File Name	src/main/webapp/index.jsp
Method	out.print("Hello "+session.getAttribute("user")+",");

```
....
5.      out.print("Hello "+session.getAttribute("user")+",");
```

Privacy Violation\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=341

Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/webapp/vulnerability/forum.jsp
Line	50	24
Object	pass	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
50.              String xPression="/users/user[username='"+user+"' and
password='"+pass+"']/name";
```

File Name src/main/webapp/vulnerability/forum.jsp

Method out.print("Hello "+session.getAttribute("user")+", Welcome to Our Forum !");

```
....
24.                      out.print("Hello
"+session.getAttribute("user")+", Welcome to Our Forum !");
```

Privacy Violation\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=342
Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/webapp/vulnerability/forum.jsp
Line	50	32
Object	pass	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
50.             String xPression="/users/user[username='"+user+"' and
password='"+pass+"']/name";
```

File Name src/main/webapp/vulnerability/forum.jsp

Method <input type="hidden" name="user" value="<% if(session.getAttribute("user")!=null){out.print(session.getAttribute("user"));} else { out.print("Anonymous"); } %>" size="50"/>

```
....
32.         <input type="hidden" name="user" value="<%
if(session.getAttribute("user")!=null){out.print(session.getAttribute("u
ser"));} else { out.print("Anonymous"); } %>" size="50"/><br/>
```

Privacy Violation\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=343
Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/webapp/vulnerability/SendMessage.jsp
Line	50	21
Object	pass	getAttribute

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
50.             String xPression="/users/user[username='"+user+"' and
password='"+pass+"']/name";
```

File Name src/main/webapp/vulnerability/SendMessage.jsp

Method <tr> <td><input type="hidden" name="sender" value="<%=session.getAttribute("user")%>"/></td></tr>

```
....
21.  <tr> <td><input type="hidden" name="sender"
value="<%=session.getAttribute("user")%>"/></td></tr>
```

Privacy Violation\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=344
Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	50	54
Object	pass	println

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
50.             String xPression="/users/user[username='"+user+"' and
password='"+pass+"']/name";
....
54.             out.println(name);
```

SSRF

Query Path:
Java\Cx\Java Medium Threat\SSRF Version:1

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A5-Broken Access Control

Description

SSRF\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=358
Status	New

The application sends a request to a remote server, for some resource, using dburl in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:103. However, an attacker can control the target of the request, by sending a URL or other data in ""dburl"" at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	112
Object	""dburl""	dburl

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.           dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

SSRF\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=359
Status	New

The application sends a request to a remote server, for some resource, using dbuser in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:103. However, an attacker can control the target of the request, by sending a URL or other data in ""dbuser"" at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	112
Object	""dbuser""	dbuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.           dbuser = request.getParameter("dbuser");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

SSRF\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=360
Status	New

The application sends a request to a remote server, for some resource, using dbpass in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:103. However, an attacker can control the target of the request, by sending a URL or other data in ""dbpass"" at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	112
Object	""dbpass""	dbpass

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

SSRF\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=361

Status	New

The application sends a request to a remote server, for some resource, using BinaryExpr in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:103. However, an attacker can control the target of the request, by sending a URL or other data in ""dburl"" at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	121
Object	""dburl""	BinaryExpr

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

SSRF\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=362
Status	New

The application sends a request to a remote server, for some resource, using BinaryExpr in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:103. However, an attacker can control the target of the request, by sending a URL or other data in ""dbname"" at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	121
Object	""dbname""	BinaryExpr

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
121.                                 con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

SSRF\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=363
Status	New

The application sends a request to a remote server, for some resource, using dbuser in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:103. However, an attacker can control the target of the request, by sending a URL or other data in ""dbuser"" at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	121
Object	""dbuser""	dbuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
121.                                 con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

SSRF\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=364
Status	New

The application sends a request to a remote server, for some resource, using dbpass in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:103. However, an attacker can control the target of the request, by sending a URL or other data in ""dbpass"" at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java:49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	121
Object	""dbpass""	dbpass

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.           dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
121.                                  con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Heap Inspection

Query Path:

Java\Cx\Java Medium Threat\Heap Inspection Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Media Protection
NIST SP 800-53: SC-4 Information in Shared Resources (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description

Heap Inspection\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=305
Status	New

Method password=""; at line 6 of src/main/webapp/login.jsp defines password, which is designated to contain user passwords. However, while plaintext passwords are later assigned to password, this variable is never cleared from memory.

	Source	Destination
File	src/main/webapp/login.jsp	src/main/webapp/login.jsp
Line	6	6
Object	password	password

Code Snippet
File Name src/main/webapp/login.jsp
Method String password="";

```
....
6.   String password="";
```

Heap Inspection\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=306
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java defines password, which is designated to contain user passwords. However, while plaintext passwords are later assigned to password, this variable is never cleared from memory.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	64	64
Object	password	password

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
64.                                          Cookie password=new
Cookie("password",pass);
```

Heap Inspection\Path 3:

Severity	Medium

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=307
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java defines pass, which is designated to contain user passwords. However, while plaintext passwords are later assigned to pass, this variable is never cleared from memory.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	44
Object	pass	pass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
```

Heap Inspection\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=308
Status	New

Method processRequest at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java defines pass, which is designated to contain user passwords. However, while plaintext passwords are later assigned to pass, this variable is never cleared from memory.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	44	44
Object	pass	pass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password");
```

Heap Inspection\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=309
Status	New

Method processRequest at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java defines pass, which is designated to contain user passwords. However, while plaintext passwords are later assigned to pass, this variable is never cleared from memory.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	36	36
Object	pass	pass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             String pass=request.getParameter("password");
```

Heap Inspection\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=310
Status	New

Method pass=request.getParameter at line 33 of src/main/webapp/vulnerability/csrf/changepassword.jsp defines pass, which is designated to contain user passwords. However, while plaintext passwords are later assigned to pass, this variable is never cleared from memory.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	33	33
Object	pass	pass

Code Snippet
File Name src/main/webapp/vulnerability/csrf/changepassword.jsp
Method String pass=request.getParameter("password");

```
....
33.          String pass=request.getParameter("password");
```

Use of a One Way Hash with a Predictable Salt

Query Path:
Java\Cx\Java Medium Threat\Use of a One Way Hash with a Predictable Salt Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Media Protection
NIST SP 800-53: SC-13 Cryptographic Protection (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description

Use of a One Way Hash with a Predictable Salt\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=366
Status	New

The application protects passwords with update in hashMe, of src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java at line 11, using a cryptographic hash ""password"". However, the code does not salt the hash with an unpredictable, random value, allowing an attacker to reverse the hash value.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java
Line	12	17
Object	""password""	update

Code Snippet

File Name src/main/webapp/admin/adminlogin.jsp
Method String pass=HashMe.hashMe(request.getParameter("password")); //Hashed Password

```
....
12.       String pass=HashMe.hashMe(request.getParameter("password"));
//Hashed Password
```

File Name src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java
Method public static String hashMe(String str)

```
....
17.             md.update(str.getBytes());
```

Use of a One Way Hash with a Predictable Salt\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=367
Status	New

The application protects passwords with update in hashMe, of src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java at line 11, using a cryptographic hash ""adminpass"". However, the code does not salt the hash with an unpredictable, random value, allowing an attacker to reverse the hash value.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java
Line	61	17
Object	""adminpass""	update

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
61.           adminpass=
HashMe.hashMe(request.getParameter("adminpass"));
```

File Name src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java

Method public static String hashMe(String str)

```
....
17.               md.update(str.getBytes());
```

Absolute Path Traversal

Query Path:
Java\Cx\Java Medium Threat\Absolute Path Traversal Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.8 - Improper access control
OWASP Top 10 2013: A4-Insecure Direct Object References
OWASP Top 10 2017: A5-Broken Access Control

Description

Absolute Path Traversal\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=81

Status	New

Method request.getParameter at line 11 of src/main/webapp/vulnerability/idor/download.jsp gets dynamic data from the ""file"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in = at line 18 of src/main/webapp/vulnerability/idor/download.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	11	18
Object	""file""	File

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method filePath = request.getParameter("file");

```
....
11.             filePath = request.getParameter("file");
```

File Name src/main/webapp/vulnerability/idor/download.jsp
Method String fileName = (new File(filePath)).getName();

```
....
18.             String fileName = (new File(filePath)).getName();
```

Download of Code Without Integrity Check

Query Path:
Java\Cx\Java Medium Threat\Download of Code Without Integrity Check Version:1

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description
Download of Code Without Integrity Check\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=296
Status	New

The method setup in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 loads and executes external code without running an integrity check, leaving it vulnerable to potential attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/co	src/main/java/org/cysecurity/cspf/jvl/co

	ntroller/Install.java	ntroller/Install.java
Line	111	111
Object	forName	forName

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
111.                    Class.forName(jdbcdriver);
```

HttpOnlyCookies In Config

Query Path:
Java\Cx\Java Medium Threat\HttpOnlyCookies In Config Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)
OWASP Top 10 2013: A3-Cross-Site Scripting (XSS)
OWASP Top 10 2017: A7-Cross-Site Scripting (XSS)

Description
HttpOnlyCookies In Config\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=321
Status	New

The src/main/webapp/WEB-INF/web.xml application configuration file, at line 1, does not define sensitive application cookies with the "httpOnly" flag, which could allow client-side scripts access to the session cookies.

	Source	Destination
File	src/main/webapp/WEB-INF/web.xml	src/main/webapp/WEB-INF/web.xml
Line	1	1
Object	CxXmlConfigClass599923495	CxXmlConfigClass599923495

Code Snippet
File Name src/main/webapp/WEB-INF/web.xml
Method <!DOCTYPE web-app PUBLIC

```
....
1.  <!DOCTYPE web-app PUBLIC
```

Improper Restriction of XXE Ref

Query Path:
Java\Cx\Java Medium Threat\Improper Restriction of XXE Ref Version:1

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A4-XML External Entities (XXE)

Description

Improper Restriction of XXE Ref\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=329
Status	New

The processRequest loads and parses XML using parse, at line 38 of src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java.

This XML was received earlier from user input, getInputStream, at line 38 of src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java. Note that parse is set to automatically load and replace any DTD entity references in the XML, including references to external files.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	44	48
Object	getInputStream	parse

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             InputStream xml=request.getInputStream();
....
48.             Document doc = builder.parse(is);
```

Plaintext Storage of a Password

Query Path:
Java\Cx\Java Medium Threat\Plaintext Storage of a Password Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Identification And Authentication
NIST SP 800-53: SC-28 Protection of Information at Rest (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description

Plaintext Storage of a Password\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=337

Status	New

The password used for authentication comparison in connect was stored in plaintext in src/main/java/org/cysecurity/cspf/jvl/model/DBConnect.java.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/model/DBConnect.java	src/main/java/org/cysecurity/cspf/jvl/model/DBConnect.java
Line	26	35
Object	properties	dbpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/model/DBConnect.java
Method public Connection connect(String path) throws IOException,ClassNotFoundException,SQLException

```
....
26.          properties.load(new FileInputStream(path));
....
35.                          con=
DriverManager.getConnection(dbfullurl,dbuser,dbpass);
```

Missing HSTS Header

Query Path:
Java\Cx\Java Medium Threat\Missing HSTS Header Version:1

Description

Missing HSTS Header\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=345
Status	New

The web-application does not define an HSTS header, leaving it vulnerable to attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	55	55
Object	print	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
55.                          out.print("Successfully created the file: <a
href='../pages/"+fileName+"'>"+fileName+"</a>");
```

Unchecked Input for Loop Condition

Query Path:
Java\Cx\Java Medium Threat\Unchecked Input for Loop Condition Version:1

Description

Unchecked Input for Loop Condition\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=365
Status	New

Method processRequest at line 38 of src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java gets user input from element getInputStream . This element's value flows through the code without being validated, and is eventually used in a loop condition in processRequest at line 38 of src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java. This constitutes an Unchecked Input for Loop Condition.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	44	53
Object	getInputStream	nodes

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             InputStream xml=request.getInputStream();
....
53.             for (int i = 0; i < nodes.getLength(); i++) {
```

Improper Exception Handling

Query Path:
Java\Cx\Java Low Visibility\Improper Exception Handling Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.5 - Improper error handling
NIST SP 800-53: SC-5 Denial of Service Protection (P1)

Description

Improper Exception Handling\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=552
Status	New

The method while at line 24 of src/main/webapp/vulnerability/idor/download.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	24	24
Object	read	read

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method while ((in != null) && ((length = in.read(byteBuffer)) != -1))

```
....
24.                 while ((in != null) && ((length = in.read(byteBuffer))
!= -1))
```

Improper Exception Handling\Path 2:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=553
Status New

The method while at line 20 of src/main/webapp/admin/manageusers.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	20	20
Object	next	next

Code Snippet
File Name src/main/webapp/admin/manageusers.jsp
Method while(rs.next())

```
....
20.    while(rs.next())
```

Improper Exception Handling\Path 3:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=554

Status	New

The method out.print at line 22 of src/main/webapp/admin/manageusers.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	22	22
Object	getString	getString

Code Snippet
File Name src/main/webapp/admin/manageusers.jsp
Method out.print("<input type='radio' name='user' value='"+rs.getString("username")+"'/> "+rs.getString("username")+"
");

```
....
22.       out.print("<input type='radio' name='user'
value='"+rs.getString("username")+"'/>
"+rs.getString("username")+"<br/>");
```

Improper Exception Handling\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=555
Status	New

The method out.print at line 22 of src/main/webapp/admin/manageusers.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	22	22
Object	getString	getString

Code Snippet
File Name src/main/webapp/admin/manageusers.jsp
Method out.print("<input type='radio' name='user' value='"+rs.getString("username")+"'/> "+rs.getString("username")+"
");

```
....
22.       out.print("<input type='radio' name='user'
value='"+rs.getString("username")+"'/>
"+rs.getString("username")+"<br/>");
```

Improper Exception Handling\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=556
Status	New

The method if at line 43 of src/main/webapp/ForgotPassword.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	43	43
Object	next	next

Code Snippet
File Name src/main/webapp/ForgotPassword.jsp
Method if(rs != null && rs.next()){

```
....
43.                     if(rs != null && rs.next()){
```

Improper Exception Handling\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=557
Status	New

The method out.print at line 44 of src/main/webapp/ForgotPassword.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	44	44
Object	getString	getString

Code Snippet
File Name src/main/webapp/ForgotPassword.jsp
Method out.print("Hello "+rs.getString("username")+", <b class='success'> Your Password is: "+rs.getString("password"));

```
....
44.                         out.print("Hello
"+rs.getString("username")+", <b class='success'> Your Password is:
"+rs.getString("password"));
```

Improper Exception Handling\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=558
Status	New

The method out.print at line 44 of src/main/webapp/ForgotPassword.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	44	44
Object	getString	getString

Code Snippet
File Name src/main/webapp/ForgotPassword.jsp
Method out.print("Hello "+rs.getString("username")+", <b class='success'> Your Password is: "+rs.getString("password"));

```
....
44.                         out.print("Hello
"+rs.getString("username")+", <b class='success'> Your Password is:
"+rs.getString("password"));
```

Improper Exception Handling\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=559
Status	New

The method if at line 22 of src/main/webapp/myprofile.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	22	22
Object	next	next

Code Snippet	
File Name	src/main/webapp/myprofile.jsp
Method	if(rs != null && rs.next())

```
....
22.                 if(rs != null && rs.next())
```

Improper Exception Handling\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=560
Status	New

The method out.print at line 24 of src/main/webapp/myprofile.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	24	24
Object	getString	getString

Code Snippet	
File Name	src/main/webapp/myprofile.jsp
Method	out.print("UserName : "+rs.getString("username")+" ");

```
....
24.                     out.print("UserName :
"+rs.getString("username")+"<br>");
```

Improper Exception Handling\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=561
Status	New

The method out.print at line 25 of src/main/webapp/myprofile.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	25	25
Object	getString	getString

Code Snippet	
File Name	src/main/webapp/myprofile.jsp
Method	out.print("Email : "+rs.getString("email")+" ");

```
....
25.                 out.print("Email : "+rs.getString("email")+"<br>");
```

Improper Exception Handling\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=562
Status	New

The method out.print at line 26 of src/main/webapp/myprofile.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	26	26
Object	getString	getString

Code Snippet	
File Name	src/main/webapp/myprofile.jsp
Method	out.print("About : "+rs.getString("about")+" ");

```
....
26.                 out.print("About : "+rs.getString("about")+"<br>");
```

Improper Exception Handling\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=563
Status	New

The method if at line 30 of src/main/webapp/myprofile.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	30	30
Object	next	next

Code Snippet

File Name	src/main/webapp/myprofile.jsp
Method	if(rs1 != null && rs1.next())

```
....
30.                     if(rs1 != null && rs1.next())
```

Improper Exception Handling\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=564
Status	New

The method out.print at line 33 of src/main/webapp/myprofile.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	33	33
Object	getString	getString

Code Snippet	
File Name	src/main/webapp/myprofile.jsp
Method	out.print("Card Number: "+rs1.getString("cardno")+" ");

```
....
33.                     out.print("Card Number:
"+rs1.getString("cardno")+"<br/>");
```

Improper Exception Handling\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=565
Status	New

The method out.print at line 34 of src/main/webapp/myprofile.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	34	34
Object	getString	getString

Code Snippet

File Name	src/main/webapp/myprofile.jsp
Method	out.print("CVV: "+rs1.getString("cvv")+" ");

```
....
34.                     out.print("CVV: "+rs1.getString("cvv")+"<br/>");
```

Improper Exception Handling\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=566
Status	New

The method out.print at line 35 of src/main/webapp/myprofile.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	35	35
Object	getString	getString

Code Snippet	
File Name	src/main/webapp/myprofile.jsp
Method	out.print("Expiry Date: "+rs1.getString("expirydate")+" ");

```
....
35.                     out.print("Expiry Date:
"+rs1.getString("expirydate")+"<br/>");
```

Improper Exception Handling\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=567
Status	New

The method if at line 17 of src/main/webapp/vulnerability/DisplayMessage.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/DisplayMessage.jsp	src/main/webapp/vulnerability/DisplayMessage.jsp
Line	17	17
Object	next	next

Code Snippet	
File Name	src/main/webapp/vulnerability/DisplayMessage.jsp
Method	if(rs.next())

```
....
17.                     if(rs.next())
```

Improper Exception Handling\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=568
Status	New

The method out.print at line 19 of src/main/webapp/vulnerability/DisplayMessage.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/DisplayMessage.jsp	src/main/webapp/vulnerability/DisplayMessage.jsp
Line	19	19
Object	getString	getString

Code Snippet	
File Name	src/main/webapp/vulnerability/DisplayMessage.jsp
Method	out.print("Sender: "+rs.getString("sender"));

```
....
19.                         out.print("<b>Sender:</b>
"+rs.getString("sender"));
```

Improper Exception Handling\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=569
Status	New

The method out.print at line 20 of src/main/webapp/vulnerability/DisplayMessage.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/DisplayMessage.jsp	src/main/webapp/vulnerability/DisplayMessage.jsp
Line	20	20

Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/DisplayMessage.jsp
Method out.print("
Subject:"+rs.getString("subject"));

```
....
20.
out.print("<br/><b>Subject:</b>"+rs.getString("subject"));
```

Improper Exception Handling\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=570
Status	New

The method out.print at line 21 of src/main/webapp/vulnerability/DisplayMessage.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/DisplayMessage.jsp	src/main/webapp/vulnerability/DisplayMessage.jsp
Line	21	21
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/DisplayMessage.jsp
Method out.print("
Message:
"+rs.getString("msg"));

```
....
21.                     out.print("<br/><b>Message:</b>
<br/>"+rs.getString("msg"));
```

Improper Exception Handling\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=571
Status	New

The method while at line 62 of src/main/webapp/vulnerability/forum.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp

Line	62	62
Object	next	next

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method while (rs.next())

```
....
62.                 while (rs.next())
```

Improper Exception Handling\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=572
Status	New

The method out.print at line 65 of src/main/webapp/vulnerability/forum.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	65	65
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method out.print("<td>"+rs.getString("title")+"</td>");

```
....
65.                     out.print("<td><a
href='forumposts.jsp?postid="+rs.getString("postid")+"'>"+rs.getString("
title")+"</a></td>");
```

Improper Exception Handling\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=573
Status	New

The method out.print at line 65 of src/main/webapp/vulnerability/forum.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	65	65
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method out.print("<td>"+rs.getString("title")+"</td>");

```
....
65.                     out.print("<td><a
href='forumposts.jsp?postid="+rs.getString("postid")+"'>"+rs.getString("
title")+"</a></td>");
```

Improper Exception Handling\Path 23:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=574
Status New

The method if at line 67 of src/main/webapp/vulnerability/forum.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	67	67
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method if(!rs.getString("user").equalsIgnoreCase("anonymous"))

```
....
67.
if(!rs.getString("user").equalsIgnoreCase("anonymous"))
```

Improper Exception Handling\Path 24:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=575
Status New

The method out.print at line 69 of src/main/webapp/vulnerability/forum.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	69	69
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method out.print(""+rs.getString("user")+"");

```
....
69.                     out.print("<a href='UserDetails.jsp?username="+rs.getString("user")+"'>"+rs.getString("user")+"</a>");
```

Improper Exception Handling\Path 25:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=576
Status New

The method out.print at line 69 of src/main/webapp/vulnerability/forum.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	69	69
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method out.print(""+rs.getString("user")+"");

```
....
69.                     out.print("<a href='UserDetails.jsp?username="+rs.getString("user")+"'>"+rs.getString("user")+"</a>");
```

Improper Exception Handling\Path 26:

Severity Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=577
Status	New

The method out.print at line 73 of src/main/webapp/vulnerability/forum.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	73	73
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method out.print(rs.getString("user"));

```
....
73.                     out.print(rs.getString("user"));
```

Improper Exception Handling\Path 27:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=578
Status	New

The method if at line 15 of src/main/webapp/vulnerability/forumposts.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forumposts.jsp	src/main/webapp/vulnerability/forumposts.jsp
Line	15	15
Object	next	next

Code Snippet
File Name src/main/webapp/vulnerability/forumposts.jsp
Method if(rs != null && rs.next())

```
....
15.                 if(rs != null && rs.next())
```

Improper Exception Handling\Path 28:

Severity	Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=579
Status	New

The method out.print at line 17 of src/main/webapp/vulnerability/forumposts.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forumposts.jsp	src/main/webapp/vulnerability/forumposts.jsp
Line	17	17
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forumposts.jsp
Method out.print("Title:"+rs.getString("title")+"");

```
....
17.                     out.print("<b style='font-
size:22px'>Title:"+rs.getString("title")+"</b>");
```

Improper Exception Handling\Path 29:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=580
Status	New

The method out.print at line 18 of src/main/webapp/vulnerability/forumposts.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forumposts.jsp	src/main/webapp/vulnerability/forumposts.jsp
Line	18	18
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forumposts.jsp
Method out.print("
- Posted By "+rs.getString("user"));

```
....
18.                     out.print("<br/>-  Posted By
"+rs.getString("user"));
```

Improper Exception Handling\Path 30:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=581
Status	New

The method out.print at line 19 of src/main/webapp/vulnerability/forumposts.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forumposts.jsp	src/main/webapp/vulnerability/forumposts.jsp
Line	19	19
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forumposts.jsp
Method out.print("

Content:
"+rs.getString("content"));

```
....
19.
out.print("<br/><br/>Content:<br/>"+rs.getString("content"));
```

Improper Exception Handling\Path 31:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=582
Status	New

The method while at line 15 of src/main/webapp/vulnerability/forumUsersList.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forumUsersList.jsp	src/main/webapp/vulnerability/forumUsersList.jsp
Line	15	15
Object	next	next

Code Snippet
File Name src/main/webapp/vulnerability/forumUsersList.jsp
Method while(rs.next())

```
....
15.                 while( rs.next())
```

Improper Exception Handling\Path 32:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=583
Status	New

The method out.print at line 18 of src/main/webapp/vulnerability/forumUsersList.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forumUsersList.jsp	src/main/webapp/vulnerability/forumUsersList.jsp
Line	18	18
Object	getString	getString

Code Snippet

File Name src/main/webapp/vulnerability/forumUsersList.jsp

Method out.print(""+rs.getString("username")+"");

```
....
18.                 out.print("<a
href='UserDetails.jsp?username="+rs.getString("username")+"'>"+rs.getStr
ing("username")+"</a>");
```

Improper Exception Handling\Path 33:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=584
Status	New

The method out.print at line 18 of src/main/webapp/vulnerability/forumUsersList.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/forumUsersList.jsp	src/main/webapp/vulnerability/forumUsersList.jsp
Line	18	18

Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/forumUsersList.jsp
Method out.print(""+rs.getString("username")+"");

```
....
18.                     out.print("<a
href='UserDetails.jsp?username="+rs.getString("username")+"'>"+rs.getStr
ing("username")+"</a>");
```

Improper Exception Handling\Path 34:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=585
Status New

The method while at line 17 of src/main/webapp/vulnerability/Messages.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/Messages.jsp	src/main/webapp/vulnerability/Messages.jsp
Line	17	17
Object	next	next

Code Snippet
File Name src/main/webapp/vulnerability/Messages.jsp
Method while (rs.next())

```
....
17.                 while (rs.next())
```

Improper Exception Handling\Path 35:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=586
Status New

The method out.print at line 19 of src/main/webapp/vulnerability/Messages.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/Messages.jsp	src/main/webapp/vulnerability/Messages.jsp
Line	19	19
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/Messages.jsp
Method out.print("
"+rs.getString("subject")+"
");

```
....
19.                  out.print("<li><a
href='DisplayMessage.jsp?msgid="+rs.getString("msgid")+"
'>"+rs.getString("subject")+"</a></li>");
```

Improper Exception Handling\Path 36:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=587
Status New

The method out.print at line 19 of src/main/webapp/vulnerability/Messages.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/Messages.jsp	src/main/webapp/vulnerability/Messages.jsp
Line	19	19
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/Messages.jsp
Method out.print("
"+rs.getString("subject")+"
");

```
....
19.                  out.print("<li><a
href='DisplayMessage.jsp?msgid="+rs.getString("msgid")+"
'>"+rs.getString("subject")+"</a></li>");
```

Improper Exception Handling\Path 37:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=588

Status	New

The method if at line 14 of src/main/webapp/vulnerability/UserDetails.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/UserDetails.jsp	src/main/webapp/vulnerability/UserDetails.jsp
Line	14	14
Object	next	next

Code Snippet
File Name src/main/webapp/vulnerability/UserDetails.jsp
Method if(rs != null && rs.next())

```
....
14.                 if(rs != null && rs.next())
```

Improper Exception Handling\Path 38:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=589
Status	New

The method out.print at line 16 of src/main/webapp/vulnerability/UserDetails.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/UserDetails.jsp	src/main/webapp/vulnerability/UserDetails.jsp
Line	16	16
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/UserDetails.jsp
Method out.print("
About "+rs.getString("username")+":

"+rs.getString("about"));

```
....
16.                     out.print("<br>About "+rs.getString("username")+":
<br>"+rs.getString("about"));
```

Improper Exception Handling\Path 39:

Severity	Low
Result State	To Verify

Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=590
Status	New

The method out.print at line 16 of src/main/webapp/vulnerability/UserDetails.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/UserDetails.jsp	src/main/webapp/vulnerability/UserDetails.jsp
Line	16	16
Object	getString	getString

Code Snippet
File Name src/main/webapp/vulnerability/UserDetails.jsp
Method out.print("
About "+rs.getString("username")+":

"+rs.getString("about"));

```
....
16.                     out.print("<br>About "+rs.getString("username")+":
<br>"+rs.getString("about"));
```

Improper Exception Handling\Path 40:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=591
Status	New

The method outStream.write at line 26 of src/main/webapp/vulnerability/idor/download.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	26	26
Object	write	write

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method outStream.write(byteBuffer,0,length);

```
....
26.                 outStream.write(byteBuffer,0,length);
```

Improper Exception Handling\Path 41:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=592
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 42:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=593
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 43:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=594
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 44:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=595
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 45:

Severity	Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=596
Status	New

The method props.load at line 20 of src/main/webapp/admin/Configure.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/admin/Configure.jsp	src/main/webapp/admin/Configure.jsp
Line	20	20
Object	load	load

Code Snippet
File Name src/main/webapp/admin/Configure.jsp
Method props.load(new FileInputStream(configPath));

```
....
20.          props.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 46:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=597
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 47:

Severity	Low
Result State	To Verify

Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=598
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 48:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=599
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 49:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=600
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Exception Handling\Path 50:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=601
Status	New

The method properties.load at line 9 of src/main/webapp/header.jsp performs an operation that could be expected to throw an exception, and is not properly wrapped in a try-catch block. This constitutes Improper Exception Handling.

	Source	Destination
File	src/main/webapp/header.jsp	src/main/webapp/header.jsp
Line	9	9
Object	load	load

Code Snippet
File Name src/main/webapp/header.jsp
Method properties.load(new FileInputStream(configPath));

```
....
9.      properties.load(new FileInputStream(configPath));
```

Improper Resource Access Authorization

Query Path:
Java\Cx\Java Low Visibility\Improper Resource Access Authorization Version:1

Categories

Description

Improper Resource Access Authorization\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=656
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java in 34 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java
Line	54	54
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.                                 pstmt.executeUpdate();
```

Improper Resource Access Authorization\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=657
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java in 36 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Line	48	48
Object	executeQuery	executeQuery

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
48.                     rs=stmt.executeQuery("select * from users where
email='"+email+"'");
```

Improper Resource Access Authorization\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=658
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	117	117
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
117.                                stmt.executeUpdate("DROP DATABASE IF
EXISTS "+dbname);
```

Improper Resource Access Authorization\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=659
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	119	119
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method	protected boolean setup(String i) throws IOException

```
....
119.                                    stmt.executeUpdate("CREATE DATABASE
"+dbname);
```

Improper Resource Access Authorization\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=660
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	126	126
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
126.                                    stmt.executeUpdate("Create table
users(ID int NOT NULL AUTO_INCREMENT, username varchar(30),email
varchar(60), password varchar(60), about varchar(50),privilege
varchar(20),avatar TEXT,secretquestion int,secret varchar(30),primary
key (id))");
```

Improper Resource Access Authorization\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=661
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	127	127
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

Improper Resource Access Authorization\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=662
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	128	128
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
128.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('victim','victim','victim@localhost','I am the victim of this
application','default.jpg','user',1,'max')");
```

Improper Resource Access Authorization\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=663
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	129	129
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
129.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('attacker','attacker','attacker@localhost','I am the attacker of this
application','default.jpg','user',1,'bella')");
```

Improper Resource Access Authorization\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=664
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	130	130
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
130.                                  stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values ('NEO','trinity','neo@matrix','I
am the NEO','default.jpg','user',1,'sentinel')");
```

Improper Resource Access Authorization\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=665
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	131	131
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
131.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('trinity','NEO','trinity@matrix','it is
Trinity','default.jpg','user',1,'sentinel')");
```

Improper Resource Access Authorization\Path 11:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=666
Status New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	132	132
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
132.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('Anderson','java','anderson@1999','I am computer
programmer','default.jpg','user',1,'C++')");
```

Improper Resource Access Authorization\Path 12:

Severity Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=667
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	135	135
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
135.                                    stmt.executeUpdate("create table
posts(postid int NOT NULL AUTO_INCREMENT, content TEXT,title
varchar(100), user varchar(30), primary key (postid))");
```

Improper Resource Access Authorization\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=668
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	136	136
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
136.                                    stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Feel free to ask any questions about
Java Vulnerable Lab','First Post', 'admin')");
```

Improper Resource Access Authorization\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=669
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	137	137
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
137.                                    stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello Guys, this is victim','Second
Post', 'victim')");
```

Improper Resource Access Authorization\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=670
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	138	138
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
138.                                    stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Hello This is attacker','Third
Post', 'attacker')");
```

Improper Resource Access Authorization\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=671
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	139	139
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
139.                                    stmt.executeUpdate("INSERT into
posts(content,title, user) values ('Trinity! Help!','Help','neo')");
```

Improper Resource Access Authorization\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=672
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	142	142
Object	executeUpdate	executeUpdate

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
142.                                    stmt.executeUpdate("create table
tdata(id int, page varchar(30))");
```

Improper Resource Access Authorization\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=673
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	143	143
Object	executeUpdate	executeUpdate

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
143.                                    stmt.executeUpdate("Insert into
tdata values(1,'ext1.html')");
```

Improper Resource Access Authorization\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=674
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	144	144
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
144.                                    stmt.executeUpdate("Insert into
tdata values(2,'ext2.html')");
```

Improper Resource Access Authorization\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=675
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	147	147
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
147.                                    stmt.executeUpdate("Create table
Messages(msgid int NOT NULL AUTO_INCREMENT,name varchar(30),email
varchar(60), msg varchar(500),primary key (msgid))");
```

Improper Resource Access Authorization\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=676
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	148	148

Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
148.                                    stmt.executeUpdate("INSERT into
Messages(name,email, msg) values ('TestUser','Test@localhost', 'Hi
admin, how are you')");
```

Improper Resource Access Authorization\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=677
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	151	151
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
151.                                    stmt.executeUpdate("Create table
UserMessages(msgid int NOT NULL AUTO_INCREMENT,recipient
varchar(30),sender varchar(30),subject varchar(60), msg
varchar(500),primary key (msgid))");
```

Improper Resource Access Authorization\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=678
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	152	152
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
152.                                    stmt.executeUpdate("INSERT into
UserMessages(recipient, sender, subject, msg) values
('attacker','admin','Hi','Hi<br/> This is admin of this page. <br/>
Welcome to Our Forum')");
```

Improper Resource Access Authorization\Path 24:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=679
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	153	153
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
153.                                    stmt.executeUpdate("INSERT into
UserMessages(recipient, sender, subject, msg) values
('victim','admin','Hi','Hi<br/> This is admin of this page. <br/>
Welcome to Our Forum')");
```

Improper Resource Access Authorization\Path 25:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=680
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	157	157
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
157.                                    stmt.executeUpdate("Create table
cards(id int,cardno varchar(80), cvv varchar(6),expirydate
varchar(15))");
```

Improper Resource Access Authorization\Path 26:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=681
Status New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	158	158
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
158.                                    stmt.executeUpdate("INSERT into
cards(id,cardno, cvv,expirydate) values
('1','4000123456789010','123','12/2014')");
```

Improper Resource Access Authorization\Path 27:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=682

Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	159	159
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
159.                                    stmt.executeUpdate("INSERT into
cards(id,cardno, cvv,expirydate) values ('2','4111111111111111
','321','7/2015')");
```

Improper Resource Access Authorization\Path 28:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=683
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	160	160
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
160.                                    stmt.executeUpdate("INSERT into
cards(id,cardno, cvv,expirydate) values
('3','5111111111111118','111','1/2017')");
```

Improper Resource Access Authorization\Path 29:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=684
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	163	163
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
163.                                    stmt.executeUpdate("Create table
FilesList(fileid int NOT NULL AUTO_INCREMENT,path text,primary key
(fileid))");
```

Improper Resource Access Authorization\Path 30:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=685
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	164	164
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
164.                                    stmt.executeUpdate("INSERT into
FilesList(path) values ('/docs/doc1.pdf')");
```

Improper Resource Access Authorization\Path 31:

Severity	Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=686
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Install.java in 103 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	165	165
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
165.                                    stmt.executeUpdate("INSERT into
FilesList(path) values ('/docs/exampledoc.pdf')");
```

Improper Resource Access Authorization\Path 32:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=687
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in 39 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	52	52
Object	executeQuery	executeQuery

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

Improper Resource Access Authorization\Path 33:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=688
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Register.java in 37 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	58	58
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.                                    stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

Improper Resource Access Authorization\Path 34:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=689
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/Register.java in 37 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	59	59
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.                                          stmt.executeUpdate("INSERT
into UserMessages(recipient, sender, subject, msg) values
('"+user+"','admin','Hi','Hi<br/> This is admin of this page. <br/>
Welcome to Our Forum')");
```

Improper Resource Access Authorization\Path 35:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=690
Status	New

An I\O action occurs at src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java in 36 without authorization checks.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Line	48	48
Object	executeQuery	executeQuery

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
48.                      rs=stmt.executeQuery("select * from users where
username='"+user+"'");
```

Improper Resource Access Authorization\Path 36:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=691
Status	New

An I\O action occurs at src/main/webapp/admin/adminlogin.jsp in 19 without authorization checks.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	19	19
Object	executeQuery	executeQuery

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp

Method	rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

Improper Resource Access Authorization\Path 37:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=692
Status	New

An I\O action occurs at src/main/webapp/admin/manageusers.jsp in 14 without authorization checks.

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	14	14
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/webapp/admin/manageusers.jsp
Method	stmt.executeUpdate("Delete from users where username='"+user+"'");

```
....
14.       stmt.executeUpdate("Delete from users where
username='"+user+"'");
```

Improper Resource Access Authorization\Path 38:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=693
Status	New

An I\O action occurs at src/main/webapp/admin/manageusers.jsp in 19 without authorization checks.

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	19	19
Object	executeQuery	executeQuery

Code Snippet

File Name	src/main/webapp/admin/manageusers.jsp
Method	ResultSet rs=stmt.executeQuery("select * from users where privilege='user'");

```
....
19.   ResultSet rs=stmt.executeQuery("select * from users where
privilege='user'");
```

Improper Resource Access Authorization\Path 39:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=694
Status	New

An I\O action occurs at src/main/webapp/changeCardDetails.jsp in 43 without authorization checks.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	43	43
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/webapp/changeCardDetails.jsp
Method	stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

Improper Resource Access Authorization\Path 40:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=695
Status	New

An I\O action occurs at src/main/webapp/ForgotPassword.jsp in 42 without authorization checks.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	42	42
Object	executeQuery	executeQuery

Code Snippet	
File Name	src/main/webapp/ForgotPassword.jsp

Method	rs=stmt.executeQuery("select * from users where username='"+request.getParameter("username").trim()+"' and secret='"+request.getParameter("secret")+"'");

```
....
42.                     rs=stmt.executeQuery("select * from users where
username='"+request.getParameter("username").trim()+"' and
secret='"+request.getParameter("secret")+"'");
```

Improper Resource Access Authorization\Path 41:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=696
Status	New

An I\O action occurs at src/main/webapp/myprofile.jsp in 21 without authorization checks.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	21	21
Object	executeQuery	executeQuery

Code Snippet	
File Name	src/main/webapp/myprofile.jsp
Method	rs=stmt.executeQuery("select * from users where id="+id);

```
....
21.                 rs=stmt.executeQuery("select * from users where
id="+id);
```

Improper Resource Access Authorization\Path 42:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=697
Status	New

An I\O action occurs at src/main/webapp/myprofile.jsp in 29 without authorization checks.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	29	29
Object	executeQuery	executeQuery

Code Snippet	
File Name	src/main/webapp/myprofile.jsp

Method	ResultSet rs1=stmt.executeQuery("select * from cards where id="+id);

```
....
29.                   ResultSet rs1=stmt.executeQuery("select * from
cards where id="+id);
```

Improper Resource Access Authorization\Path 43:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=698
Status	New

An I\O action occurs at src/main/webapp/vulnerability/csrf/change-info.jsp in 31 without authorization checks.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/change-info.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	31	31
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/webapp/vulnerability/csrf/change-info.jsp
Method	stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.               stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

Improper Resource Access Authorization\Path 44:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=699
Status	New

An I\O action occurs at src/main/webapp/vulnerability/csrf/changepassword.jsp in 40 without authorization checks.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	40	40
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/webapp/vulnerability/csrf/changepassword.jsp

Method	stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.                 stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

Improper Resource Access Authorization\Path 45:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=700
Status	New

An I\O action occurs at src/main/webapp/vulnerability/DisplayMessage.jsp in 16 without authorization checks.

	Source	Destination
File	src/main/webapp/vulnerability/DisplayMessage.jsp	src/main/webapp/vulnerability/DisplayMessage.jsp
Line	16	16
Object	executeQuery	executeQuery

Code Snippet	
File Name	src/main/webapp/vulnerability/DisplayMessage.jsp
Method	rs=stmt.executeQuery("select * from UserMessages where msgid="+request.getParameter("msgid"));

```
....
16.                 rs=stmt.executeQuery("select * from UserMessages where
msgid="+request.getParameter("msgid"));
```

Improper Resource Access Authorization\Path 46:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=701
Status	New

An I\O action occurs at src/main/webapp/vulnerability/forum.jsp in 48 without authorization checks.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	48	48
Object	executeUpdate	executeUpdate

Code Snippet	
File Name	src/main/webapp/vulnerability/forum.jsp

Method	stmt.executeUpdate("INSERT into posts(content,title,user) values ('"+content+"','"+title+"','"+user+"')");

```
....
48.
stmt.executeUpdate("INSERT into posts(content,title,user) values
('"+content+"','"+title+"','"+user+"')");
```

Improper Resource Access Authorization\Path 47:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=702
Status	New

An I\O action occurs at src/main/webapp/vulnerability/forum.jsp in 60 without authorization checks.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	60	60
Object	executeQuery	executeQuery

Code Snippet	
File Name	src/main/webapp/vulnerability/forum.jsp
Method	rs=stmt.executeQuery("select * from posts");

```
....
60.                rs=stmt.executeQuery("select * from posts");
```

Improper Resource Access Authorization\Path 48:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=703
Status	New

An I\O action occurs at src/main/webapp/vulnerability/forumposts.jsp in 14 without authorization checks.

	Source	Destination
File	src/main/webapp/vulnerability/forumposts.jsp	src/main/webapp/vulnerability/forumposts.jsp
Line	14	14
Object	executeQuery	executeQuery

Code Snippet	
File Name	src/main/webapp/vulnerability/forumposts.jsp
Method	rs=stmt.executeQuery("select * from posts where postid="+postid);

```
....
14.                rs=stmt.executeQuery("select * from posts where
postid="+postid);
```

Improper Resource Access Authorization\Path 49:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=704
Status	New

An I\O action occurs at src/main/webapp/vulnerability/forumUsersList.jsp in 12 without authorization checks.

	Source	Destination
File	src/main/webapp/vulnerability/forumUsersList.jsp	src/main/webapp/vulnerability/forumUsersList.jsp
Line	12	12
Object	executeQuery	executeQuery

Code Snippet
File Name src/main/webapp/vulnerability/forumUsersList.jsp
Method rs=stmt.executeQuery("select * from users");

```
....
12.                rs=stmt.executeQuery("select * from users");
```

Improper Resource Access Authorization\Path 50:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=705
Status	New

An I\O action occurs at src/main/webapp/vulnerability/idor/change-email.jsp in 32 without authorization checks.

	Source	Destination
File	src/main/webapp/vulnerability/idor/change-email.jsp	src/main/webapp/vulnerability/idor/change-email.jsp
Line	32	32
Object	executeUpdate	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/idor/change-email.jsp
Method stmt.executeUpdate("Update users set email='"+email+"' where id="+id);

```
....
32.              stmt.executeUpdate("Update users set email='"+email+"'
where id="+id);
```

Unsynchronized Access To Shared Data

Query Path:
Java\Cx\Java Low Visibility\Unsynchronized Access To Shared Data Version:1

Categories

NIST SP 800-53: SC-4 Information in Shared Resources (P1)

Description

Unsynchronized Access To Shared Data\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=843
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dburl in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	54
Object	""dburl""	dburl

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

Unsynchronized Access To Shared Data\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=844
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dburl in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	66
Object	""dburl""	dburl

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.           dburl = request.getParameter("dburl");
....
66.            config.setProperty("dburl",dburl);
```

Unsynchronized Access To Shared Data\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=845
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dburl in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	112
Object	""dburl""	dburl

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.           dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=846
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dburl in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	121
Object	""dburl""	dburl

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=847
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource jdbcdriver

in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	55	55
Object	""jdbcdriver""	jdbcdriver

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
55.          jdbcdriver = request.getParameter("jdbcdriver");
```

Unsynchronized Access To Shared Data\Path 6:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=848
Status New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource jdbcdriver in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	55	67
Object	""jdbcdriver""	jdbcdriver

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
55.          jdbcdriver = request.getParameter("jdbcdriver");
....
67.           config.setProperty("jdbcdriver",jdbcdriver);
```

Unsynchronized Access To Shared Data\Path 7:

Severity Low
Result State To Verify

Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=849
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource jdbcdriver in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	55	111
Object	""jdbcdriver""	jdbcdriver

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
55.          jdbcdriver = request.getParameter("jdbcdriver");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
111.                      Class.forName(jdbcdriver);
```

Unsynchronized Access To Shared Data\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=850
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	56
Object	""dbuser""	dbuser

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

Unsynchronized Access To Shared Data\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=851
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	68
Object	""dbuser""	dbuser

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
....
68.           config.setProperty("dbuser",dbuser);
```

Unsynchronized Access To Shared Data\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=852
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	112
Object	""dbuser""	dbuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                      Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=853
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	121
Object	""dbuser""	dbuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=854
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	57
Object	""dbpass""	dbpass

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

Unsynchronized Access To Shared Data\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=855
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	69
Object	""dbpass""	dbpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
....
69.           config.setProperty("dbpass",dbpass);
```

Unsynchronized Access To Shared Data\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=856
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	112
Object	""dbpass""	dbpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                      Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=857
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	121
Object	""dbpass""	dbpass

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.           dbpass = request.getParameter("dbpass");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
121.                                  con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=858
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	58
Object	""dbname""	dbname

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.           dbname = request.getParameter("dbname");
```

Unsynchronized Access To Shared Data\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=859
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	70
Object	""dbname""	dbname

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.           dbname = request.getParameter("dbname");
....
70.            config.setProperty("dbname",dbname);
```

Unsynchronized Access To Shared Data\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=860
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	117
Object	""dbname""	dbname

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
117.                                  stmt.executeUpdate("DROP DATABASE IF
EXISTS "+dbname);
```

Unsynchronized Access To Shared Data\Path 19:

Severity Low

Result State To Verify

Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=861

Status New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	119
Object	""dbname""	dbname

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
119.                                  stmt.executeUpdate("CREATE DATABASE
"+dbname);
```

Unsynchronized Access To Shared Data\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=862
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	121
Object	""dbname""	dbname

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
121.                                  con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=863
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource siteTitle in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	59	59
Object	""siteTitle""	siteTitle

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.          siteTitle= request.getParameter("siteTitle");
```

Unsynchronized Access To Shared Data\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=864
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource siteTitle in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	59	71
Object	""siteTitle""	siteTitle

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.          siteTitle= request.getParameter("siteTitle");
....
71.           config.setProperty("siteTitle",siteTitle);
```

Unsynchronized Access To Shared Data\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=865
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource adminuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	60	60
Object	""adminuser""	adminuser

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.          adminuser= request.getParameter("adminuser");
```

Unsynchronized Access To Shared Data\Path 24:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=866
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource adminuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	60	127

Object	""adminuser""	adminuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.          adminuser= request.getParameter("adminuser");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

Unsynchronized Access To Shared Data\Path 25:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=867
Status New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource AssignExpr in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	54
Object	AssignExpr	AssignExpr

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.          dburl = request.getParameter("dburl");
```

Unsynchronized Access To Shared Data\Path 26:

Severity Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=868
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource AssignExpr in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	55	55
Object	AssignExpr	AssignExpr

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
55.           jdbcdriver = request.getParameter("jdbcdriver");
```

Unsynchronized Access To Shared Data\Path 27:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=869
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource AssignExpr in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	56
Object	AssignExpr	AssignExpr

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

Unsynchronized Access To Shared Data\Path 28:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=870
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource AssignExpr in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	57	57
Object	AssignExpr	AssignExpr

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

Unsynchronized Access To Shared Data\Path 29:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=871
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource AssignExpr in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	58
Object	AssignExpr	AssignExpr

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

Unsynchronized Access To Shared Data\Path 30:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=872
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource AssignExpr in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	59	59
Object	AssignExpr	AssignExpr

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.          siteTitle= request.getParameter("siteTitle");
```

Unsynchronized Access To Shared Data\Path 31:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=873
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource AssignExpr in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	60	60
Object	AssignExpr	AssignExpr

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.           adminuser= request.getParameter("adminuser");
```

Unsynchronized Access To Shared Data\Path 32:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=874
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource AssignExpr in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	61	61
Object	AssignExpr	AssignExpr

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
61.           adminpass=
HashMe.hashMe(request.getParameter("adminpass"));
```

Unsynchronized Access To Shared Data\Path 33:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=875
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dburl in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	66	66
Object	dburl	dburl

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
66.             config.setProperty("dburl",dburl);
```

Unsynchronized Access To Shared Data\Path 34:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=876
Status New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource jdbcdriver in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	67	67
Object	jdbcdriver	jdbcdriver

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
67.             config.setProperty("jdbcdriver",jdbcdriver);
```

Unsynchronized Access To Shared Data\Path 35:

Severity Low
Result State To Verify

Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=877
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	68	68
Object	dbuser	dbuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
68.           config.setProperty("dbuser",dbuser);
```

Unsynchronized Access To Shared Data\Path 36:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=878
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	69	69
Object	dbpass	dbpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
69.           config.setProperty("dbpass",dbpass);
```

Unsynchronized Access To Shared Data\Path 37:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=879
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	70	70
Object	dbname	dbname

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
70.           config.setProperty("dbname",dbname);
```

Unsynchronized Access To Shared Data\Path 38:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=880
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource siteTitle in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	71	71
Object	siteTitle	siteTitle

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
71.           config.setProperty("siteTitle",siteTitle);
```

Unsynchronized Access To Shared Data\Path 39:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=881
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource jdbcdriver in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	111	111
Object	jdbcdriver	jdbcdriver

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
111.                       Class.forName(jdbcdriver);
```

Unsynchronized Access To Shared Data\Path 40:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=882
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dburl in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/co	src/main/java/org/cysecurity/cspf/jvl/co

	ntroller/Install.java	ntroller/Install.java
Line	112	112
Object	dburl	dburl

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 41:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=883
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dbuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	112	112
Object	dbuser	dbuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 42:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=884
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dbpass in the file

src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	112	112
Object	dbpass	dbpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 43:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=885
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	117	117
Object	dbname	dbname

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
117.                                stmt.executeUpdate("DROP DATABASE IF
EXISTS "+dbname);
```

Unsynchronized Access To Shared Data\Path 44:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=886
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	119	119
Object	dbname	dbname

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
119.                                stmt.executeUpdate("CREATE DATABASE
"+dbname);
```

Unsynchronized Access To Shared Data\Path 45:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=887
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dbuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	121	121
Object	dbuser	dbuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
121.                               con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 46:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=888
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dbpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	121	121
Object	dbpass	dbpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 47:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=889
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dburl in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	121	121
Object	dburl	dburl

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 48:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=890
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	121	121
Object	dbname	dbname

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
121.                                con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
```

Unsynchronized Access To Shared Data\Path 49:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=891
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource adminpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	127	127

Object	adminpass	adminpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

Unsynchronized Access To Shared Data\Path 50:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=892
Status	New

The concurrent process setup found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103 influences the shared resource adminuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 103. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	127	127
Object	adminuser	adminuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

Blind SQL Injections

Query Path:
Java\Cx\Java Low Visibility\Blind SQL Injections Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
OWASP Top 10 2013: A1-Injection
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)

Description

Blind SQL Injections\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=479
Status	New

The application's processRequest method executes an SQL query with executeQuery, at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""email"". This is read in the processRequest method, at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Line	42	48
Object	""email""	executeQuery

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
42.                 String email=request.getParameter("email").trim();
....
48.                      rs=stmt.executeQuery("select * from users where
email='"+email+"'");
```

Blind SQL Injections\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=480
Status	New

The application's setup method executes an SQL query with executeUpdate, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""dbname"". This is read in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	117
Object	""dbname""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.           dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method protected boolean setup(String i) throws IOException

```
....
117.                                  stmt.executeUpdate("DROP DATABASE IF
EXISTS "+dbname);
```

Blind SQL Injections\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=481
Status	New

The application's setup method executes an SQL query with executeUpdate, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""dbname"". This is read in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	119
Object	""dbname""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
119.                                  stmt.executeUpdate("CREATE DATABASE
"+dbname);
```

Blind SQL Injections\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=482
Status	New

The application's setup method executes an SQL query with executeUpdate, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""adminuser"". This is read in the processRequest method, at line 49 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	60	127
Object	""adminuser""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.          adminuser= request.getParameter("adminuser");
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected boolean setup(String i) throws IOException

```
....
127.                                    stmt.executeUpdate("INSERT into
users(username, password, email,About,avatar,
privilege,secretquestion,secret) values
('"+adminuser+"','"+adminpass+"','admin@localhost','I am the admin of
this application','default.jpg','admin',1,'rocky')");
```

Blind SQL Injections\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=483
Status	New

The application's processRequest method executes an SQL query with executeQuery, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""username"". This is read in the processRequest method, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	52
Object	""username""	executeQuery

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

Blind SQL Injections\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=484
Status	New

The application's processRequest method executes an SQL query with executeQuery, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""password"". This is read in the processRequest method, at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	52
Object	""password""	executeQuery

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password").trim();
....
52.                                         rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
```

Blind SQL Injections\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=485
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""username"". This is read in the processRequest method, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	43	59
Object	""username""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.           String user=request.getParameter("username");
....
59.                                              stmt.executeUpdate("INSERT
into UserMessages(recipient, sender, subject, msg) values
('"+user+"','admin','Hi','Hi<br/> This is admin of this page. <br/>
Welcome to Our Forum')");
```

Blind SQL Injections\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=486
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""username"". This is read in the processRequest method, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	43	58
Object	""username""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.           String user=request.getParameter("username");
....
58.                                       stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

Blind SQL Injections\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=487
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""password"". This is read in the processRequest method, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	44	58
Object	""password""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password");
....
58.                                        stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

Blind SQL Injections\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=488
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""email"". This is read in the processRequest method, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	45	58
Object	""email""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
45.            String email=request.getParameter("email");
....
58.                                        stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

Blind SQL Injections\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-

	LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=489
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""About"". This is read in the processRequest method, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	46	58
Object	""About""	executeUpdate

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
46.            String about=request.getParameter("About");
....
58.                                        stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

Blind SQL Injections\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=490
Status	New

The application's processRequest method executes an SQL query with executeUpdate, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""secret"". This is read in the processRequest method, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	47	58
Object	""secret""	executeUpdate

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
47.            String secret=request.getParameter("secret");
....
58.                                    stmt.executeUpdate("INSERT into
users(username, password, email,
About,avatar,privilege,secretquestion,secret) values
('"+user+"','"+pass+"','"+email+"','"+about+"','default.jpg','user',1,'"
+secret+"')");
```

Blind SQL Injections\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=491
Status	New

The application's processRequest method executes an SQL query with executeQuery, at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""username"". This is read in the processRequest method, at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Line	42	48
Object	""username""	executeQuery

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
42.                 String user=request.getParameter("username").trim();
....
48.                       rs=stmt.executeQuery("select * from users where
username='"+user+"'");
```

Blind SQL Injections\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=492
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 19 of src/main/webapp/admin/adminlogin.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""username"". This is read in the user=request.getParameter method, at line 11 of src/main/webapp/admin/adminlogin.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	11	19
Object	""username""	executeQuery

Code Snippet

File Name src/main/webapp/admin/adminlogin.jsp

Method String user=request.getParameter("username");

```
....
11.       String user=request.getParameter("username");
```

File Name src/main/webapp/admin/adminlogin.jsp

Method rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

Blind SQL Injections\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=493
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 14 of src/main/webapp/admin/manageusers.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""user"". This is read in the user=request.getParameter method, at line 13 of src/main/webapp/admin/manageusers.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	13	14
Object	""user""	executeUpdate

Code Snippet

File Name src/main/webapp/admin/manageusers.jsp

Method String user=request.getParameter("user");

```
....
13.       String user=request.getParameter("user");
```

File Name src/main/webapp/admin/manageusers.jsp

Method stmt.executeUpdate("Delete from users where username='"+user+"'");

```
....
14.       stmt.executeUpdate("Delete from users where
username='"+user+"'");
```

Blind SQL Injections\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=494
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an

untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""cardno"". This is read in the cardno=request.getParameter method, at line 37 of src/main/webapp/changeCardDetails.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	37	43
Object	""cardno""	executeUpdate

Code Snippet

File Name src/main/webapp/changeCardDetails.jsp

Method String cardno=request.getParameter("cardno");

```
....
37.          String cardno=request.getParameter("cardno");
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

Blind SQL Injections\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=495
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""cvv"". This is read in the cvv=request.getParameter method, at line 38 of src/main/webapp/changeCardDetails.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	38	43
Object	""cvv""	executeUpdate

Code Snippet
File Name src/main/webapp/changeCardDetails.jsp
Method String cvv=request.getParameter("cvv");

```
....
38.          String cvv=request.getParameter("cvv");
```

File Name src/main/webapp/changeCardDetails.jsp

Method stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

Blind SQL Injections\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=496
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 43 of src/main/webapp/changeCardDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""expirydate"". This is read in the expirydate=request.getParameter method, at line 39 of src/main/webapp/changeCardDetails.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	39	43
Object	""expirydate""	executeUpdate

Code Snippet
File Name src/main/webapp/changeCardDetails.jsp

Method	String expirydate=request.getParameter("expirydate");

```
....
39.          String expirydate=request.getParameter("expirydate");
```

File Name	src/main/webapp/changeCardDetails.jsp
Method	stmt.executeUpdate("INSERT into cards(id,cardno, cvv,expirydate) values ('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");

```
....
43.           stmt.executeUpdate("INSERT into cards(id,cardno,
cvv,expirydate) values
('"+id+"','"+cardno+"','"+cvv+"','"+expirydate+"')");
```

Blind SQL Injections\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=497
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 42 of src/main/webapp/ForgotPassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""username"". This is read in the rs=stmt.executeQuery method, at line 42 of src/main/webapp/ForgotPassword.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	42	42
Object	""username""	executeQuery

Code Snippet

File Name	src/main/webapp/ForgotPassword.jsp
Method	rs=stmt.executeQuery("select * from users where username='"+request.getParameter("username").trim()+"' and secret='"+request.getParameter("secret")+"'");

```
....
42.                    rs=stmt.executeQuery("select * from users where
username='"+request.getParameter("username").trim()+"' and
secret='"+request.getParameter("secret")+"'");
```

Blind SQL Injections\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=498
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 42 of src/main/webapp/ForgotPassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""secret"". This is read in the rs=stmt.executeQuery method, at line 42 of src/main/webapp/ForgotPassword.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	42	42
Object	""secret""	executeQuery

Code Snippet

File Name src/main/webapp/ForgotPassword.jsp

Method rs=stmt.executeQuery("select * from users where username='"+request.getParameter("username").trim()+"' and secret='"+request.getParameter("secret")+"'");

```
....
42.                     rs=stmt.executeQuery("select * from users where
username='"+request.getParameter("username").trim()+"' and
secret='"+request.getParameter("secret")+"'");
```

Blind SQL Injections\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=499
Status	New

The application's rs1=stmt.executeQuery method executes an SQL query with executeQuery, at line 29 of src/main/webapp/myprofile.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""id"". This is read in the id=request.getParameter method, at line 16 of src/main/webapp/myprofile.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	16	29
Object	""id""	executeQuery

Code Snippet
File Name src/main/webapp/myprofile.jsp
Method String id=request.getParameter("id");

```
....
16.     String id=request.getParameter("id");
```

File Name src/main/webapp/myprofile.jsp
Method ResultSet rs1=stmt.executeQuery("select * from cards where id="+id);

```
....
29.                 ResultSet rs1=stmt.executeQuery("select * from
cards where id="+id);
```

Blind SQL Injections\Path 22:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=500
Status New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 21 of src/main/webapp/myprofile.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""id"". This is read in the id=request.getParameter method, at line 16 of src/main/webapp/myprofile.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	16	21
Object	""id""	executeQuery

Code Snippet
File Name src/main/webapp/myprofile.jsp

Method	String id=request.getParameter("id");

```
....
16.     String id=request.getParameter("id");
```

File Name	src/main/webapp/myprofile.jsp
Method	rs=stmt.executeQuery("select * from users where id="+id);

```
....
21.                 rs=stmt.executeQuery("select * from users where
id="+id);
```

Blind SQL Injections\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=501
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 31 of src/main/webapp/vulnerability/csrf/change-info.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""info"". This is read in the info=request.getParameter method, at line 26 of src/main/webapp/vulnerability/csrf/change-info.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/change-info.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	26	31
Object	""info""	executeUpdate

Code Snippet	
File Name	src/main/webapp/vulnerability/csrf/change-info.jsp
Method	String info=request.getParameter("info");

```
....
26.     String info=request.getParameter("info");
```

File Name	src/main/webapp/vulnerability/csrf/change-info.jsp
Method	stmt.executeUpdate("Update users set about='"+info+"' where id="+id);

```
....
31.              stmt.executeUpdate("Update users set about='"+info+"'
where id="+id);
```

Blind SQL Injections\Path 24:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=502
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 40 of src/main/webapp/vulnerability/csrf/changepassword.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""password"". This is read in the pass=request.getParameter method, at line 33 of src/main/webapp/vulnerability/csrf/changepassword.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	33	40
Object	""password""	executeUpdate

Code Snippet

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp

Method String pass=request.getParameter("password");

```
....
33.          String pass=request.getParameter("password");
```

File Name src/main/webapp/vulnerability/csrf/changepassword.jsp

Method stmt.executeUpdate("Update users set password='"+pass+"' where id="+id);

```
....
40.               stmt.executeUpdate("Update users set
password='"+pass+"' where id="+id);
```

Blind SQL Injections\Path 25:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=503
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 16 of src/main/webapp/vulnerability/DisplayMessage.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""msgid"". This is read in the rs=stmt.executeQuery method, at line 16 of src/main/webapp/vulnerability/DisplayMessage.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/DisplayMessage.jsp	src/main/webapp/vulnerability/DisplayMessage.jsp
Line	16	16
Object	""msgid""	executeQuery

Code Snippet
File Name src/main/webapp/vulnerability/DisplayMessage.jsp
Method rs=stmt.executeQuery("select * from UserMessages where msgid="+request.getParameter("msgid"));

```
....
16.                rs=stmt.executeQuery("select * from UserMessages where
msgid="+request.getParameter("msgid"));
```

Blind SQL Injections\Path 26:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=504
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 48 of src/main/webapp/vulnerability/forum.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""content"". This is read in the content=request.getParameter method, at line 42 of src/main/webapp/vulnerability/forum.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp

Line	42	48
Object	""content""	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method String content=request.getParameter("content");

```
....
42.                                          String
content=request.getParameter("content");
```

File Name src/main/webapp/vulnerability/forum.jsp

Method stmt.executeUpdate("INSERT into posts(content,title,user) values ('"+content+"','"+title+"','"+user+"')");

```
....
48.
stmt.executeUpdate("INSERT into posts(content,title,user) values
('"+content+"','"+title+"','"+user+"')");
```

Blind SQL Injections\Path 27:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=505
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 48 of src/main/webapp/vulnerability/forum.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""title"". This is read in the title=request.getParameter method, at line 43 of src/main/webapp/vulnerability/forum.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	43	48
Object	""title""	executeUpdate

Code Snippet
File Name src/main/webapp/vulnerability/forum.jsp
Method String title=request.getParameter("title");

```
....
43.                                         String
title=request.getParameter("title");
```

File Name src/main/webapp/vulnerability/forum.jsp

Method stmt.executeUpdate("INSERT into posts(content,title,user) values ('"+content+"','"+title+"','"+user+"')");

```
....
48.
stmt.executeUpdate("INSERT into posts(content,title,user) values
('"+content+"','"+title+"','"+user+"')");
```

Blind SQL Injections\Path 28:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=506
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 48 of src/main/webapp/vulnerability/forum.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""user"". This is read in the user=request.getParameter method, at line 41 of src/main/webapp/vulnerability/forum.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	41	48
Object	""user""	executeUpdate

Code Snippet

File Name src/main/webapp/vulnerability/forum.jsp

Method String user=request.getParameter("user");

```
....
41.                                         String
user=request.getParameter("user");
```

File Name src/main/webapp/vulnerability/forum.jsp

Method	stmt.executeUpdate("INSERT into posts(content,title,user) values ('"+content+"','"+title+"','"+user+"')");

```
....
48.
stmt.executeUpdate("INSERT into posts(content,title,user) values
('"+content+"','"+title+"','"+user+"')");
```

Blind SQL Injections\Path 29:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=507
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/forumposts.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""postid"". This is read in the postid=request.getParameter method, at line 9 of src/main/webapp/vulnerability/forumposts.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/forumposts.jsp	src/main/webapp/vulnerability/forumposts.jsp
Line	9	14
Object	""postid""	executeQuery

Code Snippet	
File Name	src/main/webapp/vulnerability/forumposts.jsp
Method	String postid=request.getParameter("postid");

```
....
9.      String postid=request.getParameter("postid");
```

File Name	src/main/webapp/vulnerability/forumposts.jsp
Method	rs=stmt.executeQuery("select * from posts where postid="+postid);

```
....
14.               rs=stmt.executeQuery("select * from posts where
postid="+postid);
```

Blind SQL Injections\Path 30:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=508
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 32 of src/main/webapp/vulnerability/idor/change-email.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""email"". This is read in the email=request.getParameter method, at line 27 of src/main/webapp/vulnerability/idor/change-email.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/idor/change-email.jsp	src/main/webapp/vulnerability/idor/change-email.jsp
Line	27	32
Object	""email""	executeUpdate

Code Snippet

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method String email=request.getParameter("email");

```
....
27.     String email=request.getParameter("email");
```

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method stmt.executeUpdate("Update users set email='"+email+"' where id="+id);

```
....
32.             stmt.executeUpdate("Update users set email='"+email+"'
where id="+id);
```

Blind SQL Injections\Path 31:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=509
Status	New

The application's stmt.executeUpdate method executes an SQL query with executeUpdate, at line 32 of src/main/webapp/vulnerability/idor/change-email.jsp. The application constructs this SQL query by embedding

an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""id"". This is read in the id=request.getParameter method, at line 28 of src/main/webapp/vulnerability/idor/change-email.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/idor/change-email.jsp	src/main/webapp/vulnerability/idor/change-email.jsp
Line	28	32
Object	""id""	executeUpdate

Code Snippet

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method String id=request.getParameter("id");

```
....
28.     String id=request.getParameter("id");
```

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method stmt.executeUpdate("Update users set email='"+email+"' where id="+id);

```
....
32.             stmt.executeUpdate("Update users set email='"+email+"'
where id="+id);
```

Blind SQL Injections\Path 32:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=510
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""username"". This is read in the processRequest method, at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/webapp/vulnerability/Messages.jsp
Line	35	14
Object	""username""	executeQuery

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
35.             String user=request.getParameter("username");
```

File Name src/main/webapp/vulnerability/Messages.jsp

Method rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.              rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

Blind SQL Injections\Path 33:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=511
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 14 of src/main/webapp/vulnerability/Messages.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""password"". This is read in the processRequest method, at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/webapp/vulnerability/Messages.jsp
Line	36	14
Object	""password""	executeQuery

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             String pass=request.getParameter("password");
```

File Name src/main/webapp/vulnerability/Messages.jsp

Method rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.              rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

Blind SQL Injections\Path 34:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=512
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 24 of src/main/webapp/vulnerability/sqli/download_id.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""fileid"". This is read in the fileid=request.getParameter method, at line 18 of src/main/webapp/vulnerability/sqli/download_id.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	18	24
Object	""fileid""	executeQuery

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name	src/main/webapp/vulnerability/sqli/download_id.jsp
Method	rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.                 rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

Blind SQL Injections\Path 35:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=513
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 24 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""fileid"". This is read in the fileid=request.getParameter method, at line 18 of src/main/webapp/vulnerability/sqli/download_id_union.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	18	24
Object	""fileid""	executeQuery

Code Snippet

File Name	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method	String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method	rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.                 rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

Blind SQL Injections\Path 36:

Severity	Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=514
Status	New

The application's rs=stmt.executeQuery method executes an SQL query with executeQuery, at line 13 of src/main/webapp/vulnerability/UserDetails.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""username"". This is read in the username=request.getParameter method, at line 8 of src/main/webapp/vulnerability/UserDetails.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/UserDetails.jsp	src/main/webapp/vulnerability/UserDetails.jsp
Line	8	13
Object	""username""	executeQuery

Code Snippet

File Name src/main/webapp/vulnerability/UserDetails.jsp

Method String username=request.getParameter("username");

```
....
8.     String username=request.getParameter("username");
```

File Name src/main/webapp/vulnerability/UserDetails.jsp

Method rs=stmt.executeQuery("select * from users where username='"+username+"'");

```
....
13.              rs=stmt.executeQuery("select * from users where
username='"+username+"'");
```

Blind SQL Injections\Path 37:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=515
Status	New

The application's session.createQuery method executes an SQL query with createQuery, at line 11 of src/main/webapp/vulnerability/Injection/orm.jsp. The application constructs this SQL query by embedding an untrusted string into the query without proper sanitization. The concatenated string is submitted to the database, where it is parsed and executed accordingly.

If an exception is thrown by the database code, it is caught and handled. An attacker could still use inferential or boolean exploitation techniques to retrieve the data, by altering the user input ""id"". This is read in the out.print method, at line 50 of src/main/webapp/vulnerability/Injection/orm.jsp, and used without sanitization in the SQL query that is sent to the database server.

This may enable a Blind SQL Injection attack.

	Source	Destination
File	src/main/webapp/vulnerability/Injection/orm.jsp	src/main/webapp/vulnerability/Injection/orm.jsp
Line	50	11
Object	""id""	createQuery

Code Snippet
File Name src/main/webapp/vulnerability/Injection/orm.jsp
Method out.print(queryUsers(ormSession,request.getParameter("id")));

```
....
50.
out.print(queryUsers(ormSession,request.getParameter("id")));
```

File Name src/main/webapp/vulnerability/Injection/orm.jsp
Method Query query = session.createQuery("from Users where id="+id);

```
....
11.          Query query = session.createQuery("from Users where
id="+id);
```

Improper Resource Shutdown or Release

Query Path:
Java\Cx\Java Low Visibility\Improper Resource Shutdown or Release Version:1

Categories

NIST SP 800-53: SC-5 Denial of Service Protection (P1)

Description
Improper Resource Shutdown or Release\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=724
Status	New

The application's processRequest method in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java defines and initializes the FileOutputStream object at 49. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	72	74
Object	FileOutputStream	close

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
72.           FileOutputStream fileout = new
FileOutputStream(configPath);
....
74.           fileout.close();
```

Improper Resource Shutdown or Release\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=725
Status	New

The application's FileOutputStream method in src/main/webapp/admin/Configure.jsp defines and initializes the FileOutputStream object at 22. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/admin/Configure.jsp	src/main/webapp/admin/Configure.jsp
Line	22	24
Object	FileOutputStream	close

Code Snippet
File Name src/main/webapp/admin/Configure.jsp
Method FileOutputStream fileout = new FileOutputStream(configPath);

```
....
22.          FileOutputStream fileout = new
FileOutputStream(configPath);
```

File Name src/main/webapp/admin/Configure.jsp
Method fileout.close();

```
....
24.          fileout.close();
```

Improper Resource Shutdown or Release\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=726
Status	New

The application's FileOutputStream method in src/main/webapp/vulnerability/baasm/SiteTitle.jsp defines and initializes the FileOutputStream object at 33. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/baasm/SiteTitle.jsp	src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Line	33	35
Object	FileOutputStream	close

Code Snippet

File Name src/main/webapp/vulnerability/baasm/SiteTitle.jsp

Method FileOutputStream fileout = new FileOutputStream(configPath);

```
....
33.          FileOutputStream fileout = new
FileOutputStream(configPath);
```

File Name src/main/webapp/vulnerability/baasm/SiteTitle.jsp

Method fileout.close();

```
....
35.          fileout.close();
```

Improper Resource Shutdown or Release\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=727
Status	New

The application's DataInputStream method in src/main/webapp/vulnerability/idor/download.jsp defines and initializes the DataInputStream object at 22. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp

Line	22	29
Object	DataInputStream	close

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method DataInputStream in = new DataInputStream(new FileInputStream(file));

```
....
22.             DataInputStream in = new DataInputStream(new
FileInputStream(file));
```

File Name src/main/webapp/vulnerability/idor/download.jsp

Method in.close();

```
....
29.             in.close();
```

Improper Resource Shutdown or Release\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=728
Status	New

The application's DataInputStream method in src/main/webapp/vulnerability/sqli/download_id.jsp defines and initializes the DataInputStream object at 41. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	41	48
Object	DataInputStream	close

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method DataInputStream in = new DataInputStream(new FileInputStream(file));

```
....
41.                 DataInputStream in = new DataInputStream(new
FileInputStream(file));
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method in.close();

```
....
48.                 in.close();
```

Improper Resource Shutdown or Release\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=729
Status	New

The application's DataInputStream method in src/main/webapp/vulnerability/sqli/download_id_union.jsp defines and initializes the DataInputStream object at 41. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	41	48
Object	DataInputStream	close

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method DataInputStream in = new DataInputStream(new FileInputStream(file));

```
....
41.                 DataInputStream in = new DataInputStream(new
FileInputStream(file));
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method in.close();

```
....
48.                 in.close();
```

Improper Resource Shutdown or Release\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=730
Status	New

The application's setup method in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java defines and initializes the getConnection object at 103. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	112	120
Object	getConnection	close

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
112.                        Connection con=
DriverManager.getConnection(dburl,dbuser,dbpass);
....
120.                                  con.close();
```

Improper Resource Shutdown or Release\Path 8:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=731
Status New

The application's setup method in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java defines and initializes the getConnection object at 103. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	121	123
Object	getConnection	isClosed

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
121.                                  con=
DriverManager.getConnection(dburl+dbname,dbuser,dbpass);
....
123.                                    if(!con.isClosed())
```

Improper Resource Shutdown or Release\Path 9:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=732
Status	New

The application's connect method in src/main/java/org/cysecurity/cspf/jvl/model/DBConnect.java defines and initializes the getConnection object at 23. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/model/DBConnect.java	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	35	48
Object	getConnection	close

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/model/DBConnect.java

Method public Connection connect(String path) throws IOException,ClassNotFoundException,SQLException

```
....
35.                         con=
DriverManager.getConnection(dbfullurl,dbuser,dbpass);
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method in.close();

```
....
48.                     in.close();
```

Improper Resource Shutdown or Release\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=733
Status	New

The application's processRequest method in src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java defines and initializes the connect object at 36. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Line	41	47
Object	connect	createStatement

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.                 Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
....
47.                       Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=734
Status	New

The application's processRequest method in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java defines and initializes the connect object at 39. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	47	51
Object	connect	createStatement

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
47.                   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
....
51.                                       Statement stmt =
con.createStatement();
```

Improper Resource Shutdown or Release\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=735
Status	New

The application's processRequest method in src/main/java/org/cysecurity/cspf/jvl/controller/Register.java defines and initializes the connect object at 37. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	42	57
Object	connect	createStatement

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
42.              Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
....
57.                                    Statement stmt =
con.createStatement();
```

Improper Resource Shutdown or Release\Path 13:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=736
Status New

The application's processRequest method in src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java defines and initializes the connect object at 34. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java
Line	41	49
Object	connect	prepareStatement

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.                     Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
....
49.                                 PreparedStatement
pstmt=con.prepareStatement("INSERT into UserMessages(recipient, sender,
subject, msg) values (?,?,?,?)");
```

Improper Resource Shutdown or Release\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=737
Status	New

The application's processRequest method in src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java defines and initializes the connect object at 36. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Line	41	47
Object	connect	createStatement

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.                     Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
....
47.                         Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=738
Status	New

The application's DBConnect method in src/main/webapp/admin/adminlogin.jsp defines and initializes the connect object at 10. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	10	18
Object	connect	createStatement

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
10.       Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/admin/adminlogin.jsp
Method Statement stmt = con.createStatement();

```
....
18.                                   Statement stmt =
con.createStatement();
```

Improper Resource Shutdown or Release\Path 16:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=739
Status New

The application's DBConnect method in src/main/webapp/admin/manageusers.jsp defines and initializes the connect object at 9. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/admin/manageusers.jsp	src/main/webapp/admin/manageusers.jsp
Line	9	10
Object	connect	createStatement

Code Snippet
File Name src/main/webapp/admin/manageusers.jsp
Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
9.     Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/admin/manageusers.jsp

Method Statement stmt = con.createStatement();

```
....
10.      Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=740
Status	New

The application's DBConnect method in src/main/webapp/changeCardDetails.jsp defines and initializes the connect object at 27. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	27	42
Object	connect	createStatement

Code Snippet

File Name src/main/webapp/changeCardDetails.jsp

Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
27.   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/changeCardDetails.jsp

Method Statement stmt = con.createStatement();

```
....
42.           Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=741
Status	New

The application's DBConnect method in src/main/webapp/ForgotPassword.jsp defines and initializes the connect object at 39. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	39	41
Object	connect	createStatement

Code Snippet

File Name src/main/webapp/ForgotPassword.jsp

Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
39.                     Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/ForgotPassword.jsp

Method Statement stmt = con.createStatement();

```
....
41.                     Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=742
Status	New

The application's DBConnect method in src/main/webapp/myprofile.jsp defines and initializes the connect object at 14. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/myprofile.jsp	src/main/webapp/myprofile.jsp
Line	14	19

Object	connect	createStatement

Code Snippet
File Name src/main/webapp/myprofile.jsp
Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
14.   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/myprofile.jsp

Method Statement stmt = con.createStatement();

```
....
19.          Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=743
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/csrf/change-info.jsp defines and initializes the connect object at 24. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/change-info.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	24	30
Object	connect	createStatement

Code Snippet
File Name src/main/webapp/vulnerability/csrf/change-info.jsp
Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
24.   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name	src/main/webapp/vulnerability/csrf/change-info.jsp
Method	Statement stmt = con.createStatement();

```
....
30.          Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=744
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/csrf/changepassword.jsp defines and initializes the connect object at 28. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	28	39
Object	connect	createStatement

Code Snippet

File Name	src/main/webapp/vulnerability/csrf/changepassword.jsp
Method	Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
28.   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name	src/main/webapp/vulnerability/csrf/changepassword.jsp
Method	Statement stmt = con.createStatement();

```
....
39.            Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=745
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/DisplayMessage.jsp defines and initializes the connect object at 9. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/DisplayMessage.jsp	src/main/webapp/vulnerability/DisplayMessage.jsp
Line	9	14
Object	connect	createStatement

Code Snippet

File Name src/main/webapp/vulnerability/DisplayMessage.jsp

Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
9.          Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/vulnerability/DisplayMessage.jsp

Method Statement stmt = con.createStatement();

```
....
14.                 Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=746
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/forum.jsp defines and initializes the connect object at 21. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/forum.jsp	src/main/webapp/vulnerability/forum.jsp
Line	21	58
Object	connect	createStatement

Code Snippet	
File Name	src/main/webapp/vulnerability/forum.jsp
Method	Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
21.           Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name	src/main/webapp/vulnerability/forum.jsp
Method	Statement stmt = con.createStatement();

```
....
58.               Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 24:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=747
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/forumposts.jsp defines and initializes the connect object at 7. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/forumposts.jsp	src/main/webapp/vulnerability/forumposts.jsp
Line	7	12
Object	connect	createStatement

Code Snippet	
File Name	src/main/webapp/vulnerability/forumposts.jsp
Method	Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
7.   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name	src/main/webapp/vulnerability/forumposts.jsp
Method	Statement stmt = con.createStatement();

```
....
12.          Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 25:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=748
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/forumUsersList.jsp defines and initializes the connect object at 7. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/forumUsersList.jsp	src/main/webapp/vulnerability/forumUsersList.jsp
Line	7	10
Object	connect	createStatement

Code Snippet

File Name src/main/webapp/vulnerability/forumUsersList.jsp

Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
7.   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/vulnerability/forumUsersList.jsp

Method Statement stmt = con.createStatement();

```
....
10.          Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 26:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=749
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/idor/change-email.jsp defines and initializes the connect object at 25. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/idor/change-email.jsp	src/main/webapp/vulnerability/idor/change-email.jsp
Line	25	31
Object	connect	createStatement

Code Snippet

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
25.   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/vulnerability/idor/change-email.jsp

Method Statement stmt = con.createStatement();

```
....
31.          Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 27:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=750
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/Messages.jsp defines and initializes the connect object at 9. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/Messages.jsp	src/main/webapp/vulnerability/Messages.jsp
Line	9	12
Object	connect	createStatement

Code Snippet

File Name src/main/webapp/vulnerability/Messages.jsp

Method	Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
9.          Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name	src/main/webapp/vulnerability/Messages.jsp
Method	Statement stmt = con.createStatement();

```
....
12.               Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 28:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=751
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/securitymisconfig/pages.jsp defines and initializes the connect object at 10. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/securitymisconfig/pages.jsp	src/main/webapp/vulnerability/securitymisconfig/pages.jsp
Line	10	17
Object	connect	createStatement

Code Snippet

File Name	src/main/webapp/vulnerability/securitymisconfig/pages.jsp
Method	Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
10.          Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name	src/main/webapp/vulnerability/securitymisconfig/pages.jsp
Method	Statement stmt = con.createStatement();

```
....
17.                 Statement stmt = con.createStatement();
```

Improper Resource Shutdown or Release\Path 29:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=752
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/sqli/download_id.jsp defines and initializes the connect object at 21. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	21	48
Object	connect	close

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
21.       Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method in.close();

```
....
48.                   in.close();
```

Improper Resource Shutdown or Release\Path 30:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=753
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/sqli/download_id_union.jsp defines and initializes the connect object at 21. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	21	48
Object	connect	close

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
21.       Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method in.close();

```
....
48.                   in.close();
```

Improper Resource Shutdown or Release\Path 31:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=754
Status	New

The application's DBConnect method in src/main/webapp/vulnerability/UserDetails.jsp defines and initializes the connect object at 7. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	src/main/webapp/vulnerability/UserDetails.jsp	src/main/webapp/vulnerability/UserDetails.jsp
Line	7	11
Object	connect	createStatement

Code Snippet

File Name src/main/webapp/vulnerability/UserDetails.jsp

Method	Connection con=new DBConnect().connect(getServletContext().getRealPath("/WEB-INF/config.properties"));

```
....
7.   Connection con=new
DBConnect().connect(getServletContext().getRealPath("/WEB-
INF/config.properties"));
```

File Name	src/main/webapp/vulnerability/UserDetails.jsp
Method	Statement stmt = con.createStatement();

```
....
11.          Statement stmt = con.createStatement();
```

Exposure of System Data

Query Path:
Java\Cx\Java Low Visibility\Exposure of System Data Version:1

Categories

OWASP Top 10 2013: A5-Security Misconfiguration
FISMA 2014: Configuration Management
NIST SP 800-53: AC-3 Access Enforcement (P1)
OWASP Top 10 2017: A6-Security Misconfiguration

Description

Exposure of System Data\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=525
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	37	70
Object	getWriter	print

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
37.          PrintWriter out = response.getWriter();
....
70.              out.print(e);
```

Exposure of System Data\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=526
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	37	55
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
37.          PrintWriter out = response.getWriter();
....
55.                      out.print("Successfully created the file: <a href='../pages/"+fileName+"'>"+fileName+"</a>");
```

Exposure of System Data\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=527
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	37	59

Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
37.          PrintWriter out = response.getWriter();
....
59.                      out.print("Failed to create the file");
```

Exposure of System Data\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=528
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	37	64
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
37.          PrintWriter out = response.getWriter();
....
64.                  out.print("filename or content Parameter is
missing");
```

Exposure of System Data\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=529
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java at line 36 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java at line 36.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Line	39	62
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.          PrintWriter out = response.getWriter();
....
62.              out.print(e);
```

Exposure of System Data\Path 6:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=530
Status New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java at line 36 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java at line 36.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Line	39	58
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.          PrintWriter out = response.getWriter();
....
58.                  out.print(json);
```

Exposure of System Data\Path 7:

Severity Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=531
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java at line 32 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java at line 32.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java	src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java
Line	35	46
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
35.          PrintWriter out = response.getWriter();
....
46.                  out.print("Location Parameter is missing");
```

Exposure of System Data\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=532
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	96
Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.             PrintWriter out = response.getWriter();
....
96.             out.println("</html>");
```

Exposure of System Data\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=533
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	95
Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.             PrintWriter out = response.getWriter();
....
95.             out.println("</body>");
```

Exposure of System Data\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=534
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	89

Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.              PrintWriter out = response.getWriter();
....
89.                  out.print("successfully installed");
```

Exposure of System Data\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=535
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	93
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.              PrintWriter out = response.getWriter();
....
93.                  out.print("Something went wrong. Unable to
install");
```

Exposure of System Data\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=536
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	86
Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.             PrintWriter out = response.getWriter();
....
86.             out.println("<body>");
```

Exposure of System Data\Path 13:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=537
Status New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	85
Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.             PrintWriter out = response.getWriter();
....
85.             out.println("</head>");
```

Exposure of System Data\Path 14:

Severity Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=538
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	84
Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.             PrintWriter out = response.getWriter();
....
84.             out.println("<title>Servlet install</title>");
```

Exposure of System Data\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=539
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	83
Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.             PrintWriter out = response.getWriter();
....
83.             out.println("<head>");
```

Exposure of System Data\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=540
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	82
Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.             PrintWriter out = response.getWriter();
....
82.             out.println("<html>");
```

Exposure of System Data\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=541
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	81

Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.             PrintWriter out = response.getWriter();
....
81.             out.println("<!DOCTYPE html>");
```

Exposure of System Data\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=542
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/Open.java at line 31 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/Open.java at line 31.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Line	35	43
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
35.             PrintWriter out = response.getWriter();
....
43.                out.print("Missing url parameter");
```

Exposure of System Data\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=543
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java at line 36 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java at line 36.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Line	39	62
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.          PrintWriter out = response.getWriter();
....
62.              out.print(e);
```

Exposure of System Data\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=544
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java at line 36 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java at line 36.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Line	39	58
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.          PrintWriter out = response.getWriter();
....
58.                  out.print(json);
```

Exposure of System Data\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=545

Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java at line 30 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java at line 30.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	33	69
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
33.          PrintWriter out = response.getWriter();
....
69.              out.print(e);
```

Exposure of System Data\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=546
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java at line 30 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java at line 30.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	33	54
Object	getWriter	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
33.          PrintWriter out = response.getWriter();
....
54.              out.println(name);
```

Exposure of System Data\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=547
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	41	55
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.          PrintWriter out = response.getWriter();
....
55.              out.print("<br/>");
```

Exposure of System Data\Path 24:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=548
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	41	60
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.          PrintWriter out = response.getWriter();
....
60.              out.print(ex);
```

Exposure of System Data\Path 25:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=549
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	41	54
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.          PrintWriter out = response.getWriter();
....
54.              out.print(nodes.item(i).getNodeName()+" : " +
nodes.item(i).getFirstChild().getNodeValue().toString());
```

Exposure of System Data\Path 26:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=550
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	41	52

Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.           PrintWriter out = response.getWriter();
....
52.             out.print("--------------------<br/>");
```

Exposure of System Data\Path 27:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=551
Status	New

The system data read by processRequest in the file src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38 is potentially exposed by processRequest found in src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	41	51
Object	getWriter	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.           PrintWriter out = response.getWriter();
....
51.             out.print("<br/>Result:<br/>");
```

Incorrect Permission Assignment For Critical Resources

Query Path:
Java\Cx\Java Low Visibility\Incorrect Permission Assignment For Critical Resources Version:1

Categories

FISMA 2014: Access Control
NIST SP 800-53: AC-3 Access Enforcement (P1)
OWASP Top 10 2017: A6-Security Misconfiguration

Description

Incorrect Permission Assignment For Critical Resources\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=771
Status	New

A file is created on the file system by f in src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	45	45
Object	f	f

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
45.             File f=new File(filePath);
```

Incorrect Permission Assignment For Critical Resources\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=772
Status	New

A file is created on the file system by file in src/main/webapp/vulnerability/idor/download.jsp at line 12 with potentially dangerous permissions.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	12	12
Object	file	file

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method File file = new File(getServletContext().getRealPath(context));

```
....
12.             File file = new
File(getServletContext().getRealPath(context));
```

Incorrect Permission Assignment For Critical Resources\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=773
Status	New

A file is created on the file system by file in src/main/webapp/vulnerability/sqli/download_id.jsp at line 31 with potentially dangerous permissions.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	31	31
Object	file	file

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method File file = new File(getServletContext().getRealPath(path));

```
....
31.                     File file = new
File(getServletContext().getRealPath(path));
```

Incorrect Permission Assignment For Critical Resources\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=774
Status	New

A file is created on the file system by file in src/main/webapp/vulnerability/sqli/download_id_union.jsp at line 31 with potentially dangerous permissions.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	31	31
Object	file	file

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method File file = new File(getServletContext().getRealPath(path));

```
....
31.                     File file = new
File(getServletContext().getRealPath(path));
```

Incorrect Permission Assignment For Critical Resources\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=775
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	37	37
Object	out	out

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
37.           PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=776
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java at line 36 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Line	39	39
Object	out	out

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.          PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=777
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java at line 32 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java	src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java
Line	35	35
Object	out	out

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/ForwardMe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
35.          PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=778
Status	New

A file is created on the file system by fileout in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	72	72
Object	fileout	fileout

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
72.           FileOutputStream fileout = new
FileOutputStream(configPath);
```

Incorrect Permission Assignment For Critical Resources\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=779
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	79	79
Object	out	out

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
79.              PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=780
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/Logout.java at line 32 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Logout.java	src/main/java/org/cysecurity/cspf/jvl/controller/Logout.java
Line	36	36
Object	out	out

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Logout.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=781
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/Open.java at line 31 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Line	35	35
Object	out	out

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
35.             PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=782
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/Register.java at line 37 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	41	41
Object	out	out

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.             PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=783
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java at line 34 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java
Line	40	40
Object	out	out

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/SendMessage.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
40.                 PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=784
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java at line 36 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Line	39	39

Object	out	out

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.          PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=785
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java at line 30 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	33	33
Object	out	out

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
33.          PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=786
Status	New

A file is created on the file system by out in src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java

Line	41	41
Object	out	out

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
41.          PrintWriter out = response.getWriter();
```

Incorrect Permission Assignment For Critical Resources\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=787
Status	New

A file is created on the file system by fileout in src/main/webapp/admin/Configure.jsp at line 22 with potentially dangerous permissions.

	Source	Destination
File	src/main/webapp/admin/Configure.jsp	src/main/webapp/admin/Configure.jsp
Line	22	22
Object	fileout	fileout

Code Snippet
File Name src/main/webapp/admin/Configure.jsp
Method FileOutputStream fileout = new FileOutputStream(configPath);

```
....
22.          FileOutputStream fileout = new
FileOutputStream(configPath);
```

Incorrect Permission Assignment For Critical Resources\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=788
Status	New

A file is created on the file system by fileout in src/main/webapp/vulnerability/baasm/SiteTitle.jsp at line 33 with potentially dangerous permissions.

	Source	Destination
File	src/main/webapp/vulnerability/baasm/SiteTitle.jsp	src/main/webapp/vulnerability/baasm/SiteTitle.jsp

Line	33	33
Object	fileout	fileout

Code Snippet
File Name src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Method FileOutputStream fileout = new FileOutputStream(configPath);

```
....
33.          FileOutputStream fileout = new
FileOutputStream(configPath);
```

Information Exposure Through an Error Message

Query Path:
Java\Cx\Java Low Visibility\Information Exposure Through an Error Message Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.5 - Improper error handling
OWASP Top 10 2013: A5-Security Misconfiguration
FISMA 2014: Configuration Management
NIST SP 800-53: SI-11 Error Handling (P2)
OWASP Top 10 2017: A6-Security Misconfiguration

Description
Information Exposure Through an Error Message\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=755
Status	New

Method processRequest, at line 34 of src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to print, in method processRequest of src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java, line 34.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	68	70
Object	e	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
68.          catch(Exception e)
....
70.              out.print(e);
```

Information Exposure Through an Error Message\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=756
Status	New

Method processRequest, at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to print, in method processRequest of src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java, line 36.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java
Line	60	62
Object	e	print

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/EmailCheck.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.          catch(Exception e)
....
62.              out.print(e);
```

Information Exposure Through an Error Message\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=757
Status	New

Method setup, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, handles an Exception or runtime Error ex. During the exception handling code, the application exposes the exception details to println, in method setup of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, line 103.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Line	172	176
Object	ex	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
172.                    catch(SQLException ex)
....
176.                      System.out.println("VendorError: " +
ex.getErrorCode());
```

Information Exposure Through an Error Message\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=758
Status	New

Method setup, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, handles an Exception or runtime Error ex. During the exception handling code, the application exposes the exception details to println, in method setup of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, line 103.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	172	175
Object	ex	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
172.                    catch(SQLException ex)
....
175.                      System.out.println("SQLState: " +
ex.getSQLState());
```

Information Exposure Through an Error Message\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=759
Status	New

Method setup, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, handles an Exception or runtime Error ex. During the exception handling code, the application exposes the exception details to println, in method setup of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, line 103.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	172	174
Object	ex	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
172.                    catch(SQLException ex)
....
174.                        System.out.println("SQLException: " +
ex.getMessage());
```

Information Exposure Through an Error Message\Path 6:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=760
Status New

Method setup, at line 103 of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, handles an Exception or runtime Error ex. During the exception handling code, the application exposes the exception details to print, in method setup of src/main/java/org/cysecurity/cspf/jvl/controller/Install.java, line 103.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	178	180
Object	ex	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected boolean setup(String i) throws IOException

```
....
178.                    catch(ClassNotFoundException ex)
....
180.                        System.out.print("JDBC Driver
Missing:<br/>"+ex);
```

Information Exposure Through an Error Message\Path 7:

Severity Low

Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=761
Status	New

Method processRequest, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, handles an Exception or runtime Error ex. During the exception handling code, the application exposes the exception details to println, in method processRequest of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, line 37.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	69	73
Object	ex	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
69.                 catch(SQLException ex)
....
73.                             System.out.println("VendorError: " +
ex.getErrorCode());
```

Information Exposure Through an Error Message\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=762
Status	New

Method processRequest, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, handles an Exception or runtime Error ex. During the exception handling code, the application exposes the exception details to println, in method processRequest of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, line 37.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	69	72
Object	ex	println

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
69.                 catch(SQLException ex)
....
72.                             System.out.println("SQLState: " +
ex.getSQLState());
```

Information Exposure Through an Error Message\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=763
Status	New

Method processRequest, at line 37 of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, handles an Exception or runtime Error ex. During the exception handling code, the application exposes the exception details to println, in method processRequest of src/main/java/org/cysecurity/cspf/jvl/controller/Register.java, line 37.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	69	71
Object	ex	println

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
69.                 catch(SQLException ex)
....
71.                              System.out.println("SQLException: " +
ex.getMessage());
```

Information Exposure Through an Error Message\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=764
Status	New

Method processRequest, at line 36 of src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to print, in method processRequest of src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java, line 36.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java	src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Line	60	62
Object	e	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/UsernameCheck.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.           catch(Exception e)
....
62.               out.print(e);
```

Information Exposure Through an Error Message\Path 11:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=765
Status New

Method processRequest, at line 30 of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to print, in method processRequest of src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java, line 30.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	67	69
Object	e	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
67.           catch(Exception e)
....
69.               out.print(e);
```

Information Exposure Through an Error Message\Path 12:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=766
Status	New

Method processRequest, at line 38 of src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java, handles an Exception or runtime Error ex. During the exception handling code, the application exposes the exception details to print, in method processRequest of src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java, line 38.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	58	60
Object	ex	print

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.           catch(Exception ex)
....
60.               out.print(ex);
```

Information Exposure Through an Error Message\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=767
Status	New

Method catch, at line 55 of src/main/webapp/changeCardDetails.jsp, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to print, in method out.print of src/main/webapp/changeCardDetails.jsp, line 57.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	55	57
Object	e	print

Code Snippet
File Name src/main/webapp/changeCardDetails.jsp
Method catch(Exception e)

```
....
55.      catch(Exception e)
```

File Name src/main/webapp/changeCardDetails.jsp

Method	out.print(e);

```
....
57.          out.print(e);
```

Information Exposure Through an Error Message\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=768
Status	New

Method catch, at line 53 of src/main/webapp/vulnerability/Injection/orm.jsp, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to print, in method out.print of src/main/webapp/vulnerability/Injection/orm.jsp, line 55.

	Source	Destination
File	src/main/webapp/vulnerability/Injection/orm.jsp	src/main/webapp/vulnerability/Injection/orm.jsp
Line	53	55
Object	e	print

Code Snippet

File Name	src/main/webapp/vulnerability/Injection/orm.jsp
Method	catch(Exception e)

```
....
53.          catch(Exception e)
```

File Name	src/main/webapp/vulnerability/Injection/orm.jsp
Method	out.print(e);

```
....
55.              out.print(e);
```

Information Exposure Through an Error Message\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=769
Status	New

Method catch, at line 32 of src/main/webapp/vulnerability/securitymisconfig/pages.jsp, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to print, in method out.print of src/main/webapp/vulnerability/securitymisconfig/pages.jsp, line 34.

	Source	Destination
File	src/main/webapp/vulnerability/securitymisconfig/pages.jsp	src/main/webapp/vulnerability/securitymisconfig/pages.jsp
Line	32	34
Object	e	print

Code Snippet
File Name src/main/webapp/vulnerability/securitymisconfig/pages.jsp
Method catch(SQLException e)

```
....
32.      catch(SQLException e)
```

File Name src/main/webapp/vulnerability/securitymisconfig/pages.jsp
Method out.print(e.getMessage());

```
....
34.          out.print(e.getMessage());
```

Information Exposure Through Query String

Query Path:
Java\Cx\Java Low Visibility\Information Exposure Through Query String Version:1

Description

Information Exposure Through Query String\Path 1:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=789
Status New

The password getParameter at src/main/java/org/cysecurity/cspf/jvl/controller/Register.java in line 37 is sent in a URL as a GET parameter by secret at src/main/java/org/cysecurity/cspf/jvl/controller/Register.java in line 37.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	47	47
Object	getParameter	secret

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
47.             String secret=request.getParameter("secret");
```

Information Exposure Through Query String\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=790
Status	New

The password getParameter at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in line 39 is sent in a URL as a GET parameter by pass at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in line 39.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	44
Object	getParameter	pass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
```

Information Exposure Through Query String\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=791
Status	New

The password getParameter at src/main/java/org/cysecurity/cspf/jvl/controller/Register.java in line 37 is sent in a URL as a GET parameter by pass at src/main/java/org/cysecurity/cspf/jvl/controller/Register.java in line 37.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	44	44
Object	getParameter	pass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password");
```

Information Exposure Through Query String\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=792
Status	New

The password getParameter at src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java in line 30 is sent in a URL as a GET parameter by pass at src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java in line 30.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	36	36
Object	getParameter	pass

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             String pass=request.getParameter("password");
```

Information Exposure Through Query String\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=793
Status	New

The password getParameter at src/main/webapp/vulnerability/csrf/changepassword.jsp in line 33 is sent in a URL as a GET parameter by pass at src/main/webapp/vulnerability/csrf/changepassword.jsp in line 33.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	33	33
Object	getParameter	pass

Code Snippet	
File Name	src/main/webapp/vulnerability/csrf/changepassword.jsp
Method	String pass=request.getParameter("password");

```
....
33.          String pass=request.getParameter("password");
```

Information Exposure Through Query String\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=794
Status	New

The password ""password"" at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in line 39 is sent in a URL as a GET parameter by getParameter
at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in line 39.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	44
Object	""password""	getParameter

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password").trim();
```

Information Exposure Through Query String\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=795
Status	New

The password ""password"" at src/main/java/org/cysecurity/cspf/jvl/controller/Register.java in line 37 is sent in a URL as a GET parameter by getParameter at src/main/java/org/cysecurity/cspf/jvl/controller/Register.java in line 37.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java	src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Line	44	44

Object	""password""	getParameter

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Register.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password");
```

Information Exposure Through Query String\Path 8:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=796
Status New

The password ""password"" at src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java in line 30 is sent in a URL as a GET parameter by getParameter at src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java in line 30.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java	src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Line	36	36
Object	""password""	getParameter

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/XPathQuery.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             String pass=request.getParameter("password");
```

Information Exposure Through Query String\Path 9:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=797
Status New

The password ""password"" at src/main/webapp/admin/adminlogin.jsp in line 12 is sent in a URL as a GET parameter by getParameter at src/main/webapp/admin/adminlogin.jsp in line 12.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp

Line	12	12
Object	""password""	getParameter

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method String pass=HashMe.hashMe(request.getParameter("password")); //Hashed Password

```
....
12.       String pass=HashMe.hashMe(request.getParameter("password"));
//Hashed Password
```

Information Exposure Through Query String\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=798
Status	New

The password ""password"" at src/main/webapp/vulnerability/csrf/changepassword.jsp in line 33 is sent in a URL as a GET parameter by getParameter at src/main/webapp/vulnerability/csrf/changepassword.jsp in line 33.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	33	33
Object	""password""	getParameter

Code Snippet
File Name src/main/webapp/vulnerability/csrf/changepassword.jsp
Method String pass=request.getParameter("password");

```
....
33.          String pass=request.getParameter("password");
```

Information Exposure Through Query String\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=799
Status	New

The password ""confirmpassword"" at src/main/webapp/vulnerability/csrf/changepassword.jsp in line 34 is sent in a URL as a GET parameter by getParameter at src/main/webapp/vulnerability/csrf/changepassword.jsp in line 34.

	Source	Destination

File	src/main/webapp/vulnerability/csrf/changepassword.jsp	src/main/webapp/vulnerability/csrf/changepassword.jsp
Line	34	34
Object	""confirmpassword""	getParameter

Code Snippet
File Name src/main/webapp/vulnerability/csrf/changepassword.jsp
Method String confirmPass=request.getParameter("confirmpassword");

```
....
34.          String confirmPass=request.getParameter("confirmpassword");
```

Data Leak Between Sessions

Query Path:
Java\Cx\Java Low Visibility\Data Leak Between Sessions Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.10 - Broken authentication and session management
OWASP Top 10 2013: A2-Broken Authentication and Session Management
NIST SP 800-53: SC-4 Information in Shared Resources (P1)
OWASP Top 10 2017: A2-Broken Authentication

Description
Data Leak Between Sessions\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=517
Status	New

The concurrent process dburl; found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 30 influences the shared resource Install in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 28. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	30	28
Object	dburl	Install

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method static String dburl;

```
....
30.          static String dburl;
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	public class Install extends HttpServlet {

```
....
28.  public class Install extends HttpServlet {
```

Data Leak Between Sessions\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=518
Status	New

The concurrent process jdbcdriver; found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 31 influences the shared resource Install in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 28. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	31	28
Object	jdbcdriver	Install

Code Snippet

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	static String jdbcdriver;

```
....
31.          static String jdbcdriver;
```

File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	public class Install extends HttpServlet {

```
....
28.  public class Install extends HttpServlet {
```

Data Leak Between Sessions\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=519
Status	New

The concurrent process dbuser; found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 32 influences the shared resource Install in the file

src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 28. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	32	28
Object	dbuser	Install

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method static String dbuser;

```
....
32.         static String dbuser;
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method public class Install extends HttpServlet {

```
....
28.  public class Install extends HttpServlet {
```

Data Leak Between Sessions\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=520
Status	New

The concurrent process dbpass; found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 33 influences the shared resource Install in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 28. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	33	28
Object	dbpass	Install

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method static String dbpass;

```
....
33.         static String dbpass;
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method public class Install extends HttpServlet {

```
....
28.  public class Install extends HttpServlet {
```

Data Leak Between Sessions\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=521
Status	New

The concurrent process dbname; found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 34 influences the shared resource Install in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 28. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	34	28
Object	dbname	Install

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method static String dbname;

```
....
34.          static String dbname;
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method public class Install extends HttpServlet {

```
....
28.  public class Install extends HttpServlet {
```

Data Leak Between Sessions\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=522
Status	New

The concurrent process siteTitle; found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 35 influences the shared resource Install in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 28. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	35	28
Object	siteTitle	Install

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method static String siteTitle;

```
....
35.         static String siteTitle;
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method public class Install extends HttpServlet {

```
....
28.  public class Install extends HttpServlet {
```

Data Leak Between Sessions\Path 7:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=523
Status New

The concurrent process adminuser; found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 36 influences the shared resource Install in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 28. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	36	28
Object	adminuser	Install

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method static String adminuser;

```
....
36.         static String adminuser;
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method public class Install extends HttpServlet {

```
....
28.  public class Install extends HttpServlet {
```

Data Leak Between Sessions\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=524
Status	New

The concurrent process adminpass; found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 37 influences the shared resource Install in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 28. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	37	28
Object	adminpass	Install

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method static String adminpass;

```
....
37.         static String adminpass;
```

File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java

Method public class Install extends HttpServlet {

```
....
28.  public class Install extends HttpServlet {
```

Race Condition

Query Path:

Java\Cx\Java Low Visibility\Race Condition Version:1

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: AC-3 Access Enforcement (P1)

Description

Race Condition\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=816
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dburl in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	54	54
Object	dburl	dburl

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
54.           dburl = request.getParameter("dburl");
```

Race Condition\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=817
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource jdbcdriver in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	55	55
Object	jdbcdriver	jdbcdriver

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
55.          jdbcdriver = request.getParameter("jdbcdriver");
```

Race Condition\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=818
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	56	56
Object	dbuser	dbuser

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
56.          dbuser = request.getParameter("dbuser");
```

Race Condition\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=819
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/co	src/main/java/org/cysecurity/cspf/jvl/co

	ntroller/Install.java	ntroller/Install.java
Line	57	57
Object	dbpass	dbpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
57.          dbpass = request.getParameter("dbpass");
```

Race Condition\Path 5:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=820
Status New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource dbname in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	58	58
Object	dbname	dbname

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
58.          dbname = request.getParameter("dbname");
```

Race Condition\Path 6:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=821
Status New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource siteTitle in

the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	59	59
Object	siteTitle	siteTitle

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
59.          siteTitle= request.getParameter("siteTitle");
```

Race Condition\Path 7:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=822
Status New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource adminuser in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	60	60
Object	adminuser	adminuser

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.          adminuser= request.getParameter("adminuser");
```

Race Condition\Path 8:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=823
Status	New

The concurrent process processRequest found in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49 influences the shared resource adminpass in the file src/main/java/org/cysecurity/cspf/jvl/controller/Install.java at line 49. When performed concurrently, an unexpected race condition may occur.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java	src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Line	61	61
Object	adminpass	adminpass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/Install.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
61.           adminpass=
HashMe.hashMe(request.getParameter("adminpass"));
```

Stored Boundary Violation

Query Path:
Java\Cx\Java Stored\Stored Boundary Violation Version:1

Categories

OWASP Top 10 2017: A5-Broken Access Control

Description
Stored Boundary Violation\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=894
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element rs. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	52	58
Object	rs	getString

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
....
58.                                     session.setAttribute("avatar",
rs.getString("avatar"));
```

Stored Boundary Violation\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=895
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element rs. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	52	57
Object	rs	getString

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
....
57.                                     session.setAttribute("user",
rs.getString("username"));
```

Stored Boundary Violation\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=896
Status	New

Method processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java gets user input from element rs. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in processRequest at line 39 of src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	52	56
Object	rs	getString

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
52.                                    rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"'");
....
56.                                    session.setAttribute("userid",
rs.getString("id"));
```

Stored Boundary Violation\Path 4:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=897
Status New

Method rs=stmt.executeQuery at line 19 of src/main/webapp/admin/adminlogin.jsp gets user input from element rs. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in session.setAttribute at line 25 of src/main/webapp/admin/adminlogin.jsp. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	19	25
Object	rs	getString

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

File Name src/main/webapp/admin/adminlogin.jsp

Method session.setAttribute("privilege", rs.getString("privilege"));

```
....
25.
session.setAttribute("privilege", rs.getString("privilege"));
```

Stored Boundary Violation\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=898
Status	New

Method rs=stmt.executeQuery at line 19 of src/main/webapp/admin/adminlogin.jsp gets user input from element rs. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in session.setAttribute at line 24 of src/main/webapp/admin/adminlogin.jsp. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	19	24
Object	rs	getString

Code Snippet

File Name src/main/webapp/admin/adminlogin.jsp

Method rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

File Name src/main/webapp/admin/adminlogin.jsp

Method session.setAttribute("avatar", rs.getString("avatar"));

```
....
24.                                     session.setAttribute("avatar",
rs.getString("avatar"));
```

Stored Boundary Violation\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=899
Status	New

Method rs=stmt.executeQuery at line 19 of src/main/webapp/admin/adminlogin.jsp gets user input from element rs. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in session.setAttribute at line 23 of src/main/webapp/admin/adminlogin.jsp. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	19	23
Object	rs	getString

Code Snippet

File Name src/main/webapp/admin/adminlogin.jsp

Method rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

File Name src/main/webapp/admin/adminlogin.jsp

Method session.setAttribute("user", rs.getString("username"));

```
....
23.                                     session.setAttribute("user",
rs.getString("username"));
```

Stored Boundary Violation\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=900
Status	New

Method rs=stmt.executeQuery at line 19 of src/main/webapp/admin/adminlogin.jsp gets user input from element rs. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in session.setAttribute at line 22 of src/main/webapp/admin/adminlogin.jsp. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp

Line	19	22
Object	rs	getString

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method rs=stmt.executeQuery("select * from users where username='"+user+"' and password='"+pass+"' and privilege='admin'");

```
....
19.                                     rs=stmt.executeQuery("select *
from users where username='"+user+"' and password='"+pass+"' and
privilege='admin'");
```

File Name src/main/webapp/admin/adminlogin.jsp
Method session.setAttribute("userid", rs.getString("id"));

```
....
22.                                     session.setAttribute("userid",
rs.getString("id"));
```

Relative Path Traversal

Query Path:
Java\Cx\Java Low Visibility\Relative Path Traversal Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.8 - Improper access control
OWASP Top 10 2013: A4-Insecure Direct Object References
OWASP Top 10 2017: A5-Broken Access Control

Description
Relative Path Traversal\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=810
Status	New

Method request.getParameter at line 11 of src/main/webapp/vulnerability/idor/download.jsp gets dynamic data from the ""file"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in File at line 13 of src/main/webapp/vulnerability/idor/download.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	11	13
Object	""file""	File

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method filePath = request.getParameter("file");

```
....
11.             filePath = request.getParameter("file");
```

File Name src/main/webapp/vulnerability/idor/download.jsp

Method file = new File(file.getParent()+"/docs/"+filePath);

```
....
13.             file = new File(file.getParent()+"/docs/"+filePath);
```

Relative Path Traversal\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=811
Status	New

Method processRequest at line 34 of src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java gets dynamic data from the ""filename"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in processRequest at line 34 of src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	39	45
Object	""filename""	File

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
39.             String fileName=request.getParameter("filename");
....
45.              File f=new File(filePath);
```

Relative Path Traversal\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=812
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id.jsp gets dynamic data from the ""fileid"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in = at line 37 of src/main/webapp/vulnerability/sqli/download_id.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	18	37
Object	""fileid""	File

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method String fileName = (new File(filePath)).getName();

```
....
37.                  String fileName = (new File(filePath)).getName();
```

Relative Path Traversal\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=813
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id_union.jsp gets dynamic data from the ""fileid"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in = at line 37 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	18	37
Object	""fileid""	File

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method String fileName = (new File(filePath)).getName();

```
....
37.                  String fileName = (new File(filePath)).getName();
```

Relative Path Traversal\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=814
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id.jsp gets dynamic data from the ""fileid"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in File at line 32 of src/main/webapp/vulnerability/sqli/download_id.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	18	32
Object	""fileid""	File

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method file = new File(file.getParent()+filePath);

```
....
32.                  file = new File(file.getParent()+filePath);
```

Relative Path Traversal\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=815
Status	New

Method fileid=request.getParameter at line 18 of src/main/webapp/vulnerability/sqli/download_id_union.jsp gets dynamic data from the ""fileid"" element. This element's value then flows through the code and is eventually used in a file path for local disk access in File at line 32 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	18	32
Object	""fileid""	File

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method String fileid=request.getParameter("fileid");

```
....
18.      String fileid=request.getParameter("fileid");
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method file = new File(file.getParent()+filePath);

```
....
32.                  file = new File(file.getParent()+filePath);
```

Open Redirect

Query Path:
Java\Cx\Java Low Visibility\Open Redirect Version:0

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.8 - Improper access control
OWASP Top 10 2013: A10-Unvalidated Redirects and Forwards
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)

Description

Open Redirect\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=802
Status	New

The potentially tainted value provided by ""password""
in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java at line 39 is used as a destination URL by sendRedirect in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java at line 39, potentially allowing attackers to perform an open redirection.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	68
Object	""password""	sendRedirect

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
....
68.
response.sendRedirect(response.encodeURL("ForwardMe?location=/index.jsp"
));
```

Open Redirect\Path 2:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=803
Status New

The potentially tainted value provided by ""username""
in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java at line 39 is used as a destination URL by sendRedirect in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java at line 39, potentially allowing attackers to perform an open redirection.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	68
Object	""username""	sendRedirect

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.          String user=request.getParameter("username").trim();
....
68.
response.sendRedirect(response.encodeURL("ForwardMe?location=/index.jsp"
));
```

Open Redirect\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=804
Status	New

The potentially tainted value provided by ""password""
in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java at line 39 is used as a destination URL by sendRedirect in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java at line 39, potentially allowing attackers to perform an open redirection.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	79
Object	""password""	sendRedirect

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String pass=request.getParameter("password").trim();
....
79.
response.sendRedirect("login.jsp?err=something went wrong");
```

Open Redirect\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=805
Status	New

The potentially tainted value provided by ""username""
in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java at line 39 is used as a destination URL by sendRedirect in src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java at line 39, potentially allowing attackers to perform an open redirection.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	79
Object	""username""	sendRedirect

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java

Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
79.
response.sendRedirect("login.jsp?err=something went wrong");
```

Open Redirect\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=806
Status	New

The potentially tainted value provided by ""url"" in src/main/java/org/cysecurity/cspf/jvl/controller/Open.java at line 31 is used as a destination URL by sendRedirect in src/main/java/org/cysecurity/cspf/jvl/controller/Open.java at line 31, potentially allowing attackers to perform an open redirection.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Line	36	39
Object	""url""	sendRedirect

Code Snippet	
File Name	src/main/java/org/cysecurity/cspf/jvl/controller/Open.java
Method	protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.             String url=request.getParameter("url");
....
39.                response.sendRedirect(url);
```

Reliance on Cookies in a Decision

Query Path:
Java\Cx\Java Low Visibility\Reliance on Cookies in a Decision Version:1

Categories

OWASP Top 10 2017: A2-Broken Authentication

Description

Reliance on Cookies in a Decision\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=824

Status	New

The cookie getCookies obtained by request.getCookies at src/main/webapp/login.jsp in line 7 uses its value in making a decision in if at src/main/webapp/login.jsp in line 13, without properly validating its contents.

	Source	Destination
File	src/main/webapp/login.jsp	src/main/webapp/login.jsp
Line	7	13
Object	getCookies	equals

Code Snippet
File Name src/main/webapp/login.jsp
Method Cookie[] cookies = request.getCookies();

```
....
7.   Cookie[] cookies = request.getCookies();
```

File Name src/main/webapp/login.jsp

Method else if("password".equals(c.getName()))

```
....
13.          else if("password".equals(c.getName()))
```

Reliance on Cookies in a Decision\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=825
Status	New

The cookie getCookies obtained by request.getCookies at src/main/webapp/login.jsp in line 7 uses its value in making a decision in if at src/main/webapp/login.jsp in line 10, without properly validating its contents.

	Source	Destination
File	src/main/webapp/login.jsp	src/main/webapp/login.jsp
Line	7	10
Object	getCookies	equals

Code Snippet
File Name src/main/webapp/login.jsp
Method Cookie[] cookies = request.getCookies();

```
....
7.   Cookie[] cookies = request.getCookies();
```

File Name	src/main/webapp/login.jsp
Method	if ("username".equals(c.getName())) {

```
....
10.          if ("username".equals(c.getName())) {
```

Reliance on Cookies in a Decision\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=826
Status	New

The cookie getCookies obtained by request.getCookies at src/main/webapp/vulnerability/baasm/SiteTitle.jsp in line 7 uses its value in making a decision in if at src/main/webapp/vulnerability/baasm/SiteTitle.jsp in line 15, without properly validating its contents.

	Source	Destination
File	src/main/webapp/vulnerability/baasm/SiteTitle.jsp	src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Line	7	15
Object	getCookies	&&

Code Snippet	
File Name	src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Method	Cookie[] cookies = request.getCookies();

```
....
7.   Cookie[] cookies = request.getCookies();
```

File Name	src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Method	if(!privilege.equalsIgnoreCase("") && privilege.equalsIgnoreCase("admin"))

```
....
15.   if(!privilege.equalsIgnoreCase("") &&
privilege.equalsIgnoreCase("admin"))
```

Reliance on Cookies in a Decision\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=827
Status	New

The cookie getCookies obtained by request.getCookies at src/main/webapp/vulnerability/baasm/SiteTitle.jsp in line 7 uses its value in making a decision in if at src/main/webapp/vulnerability/baasm/SiteTitle.jsp in line 10, without properly validating its contents.

	Source	Destination
File	src/main/webapp/vulnerability/baasm/SiteTitle.jsp	src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Line	7	10
Object	getCookies	equals

Code Snippet
File Name src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Method Cookie[] cookies = request.getCookies();

```
....
7.   Cookie[] cookies = request.getCookies();
```

File Name src/main/webapp/vulnerability/baasm/SiteTitle.jsp
Method if ("privilege".equals(c.getName())) {

```
....
10.          if ("privilege".equals(c.getName())) {
```

Sensitive Cookie in HTTPS Session Without Secure Attribute

Query Path:
Java\Cx\Java Low Visibility\Sensitive Cookie in HTTPS Session Without Secure Attribute Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Configuration Management
NIST SP 800-53: SC-8 Transmission Confidentiality and Integrity (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description
Sensitive Cookie in HTTPS Session Without Secure Attribute\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=833
Status	New

The src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java application configuration file, at line 39, does not define sensitive application cookies with the "secure" flag, which could cause the client to send those cookies in plaintext over an insecure network communication (HTTP). This may lead to a Session Hijacking attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/co	src/main/java/org/cysecurity/cspf/jvl/co

	ntroller/LoginValidator.java	ntroller/LoginValidator.java
Line	60	60
Object	privilege	privilege

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
60.                                    response.addCookie(privilege);
```

Sensitive Cookie in HTTPS Session Without Secure Attribute\Path 2:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=834
Status New

The src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java application configuration file, at line 39, does not define sensitive application cookies with the "secure" flag, which could cause the client to send those cookies in plaintext over an insecure network communication (HTTP). This may lead to a Session Hijacking attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	65	65
Object	username	username

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
65.
response.addCookie(username);
```

Sensitive Cookie in HTTPS Session Without Secure Attribute\Path 3:

Severity Low
Result State To Verify
Online Results http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=835
Status New

The src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java application configuration file, at line 39, does not define sensitive application cookies with the "secure" flag, which could cause the client to send

those cookies in plaintext over an insecure network communication (HTTP). This may lead to a Session Hijacking attack.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	66	66
Object	password	password

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
66.
response.addCookie(password);
```

Sensitive Cookie in HTTPS Session Without Secure Attribute\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=836
Status	New

The src/main/webapp/admin/adminlogin.jsp application configuration file, at line 29, does not define sensitive application cookies with the "secure" flag, which could cause the client to send those cookies in plaintext over an insecure network communication (HTTP). This may lead to a Session Hijacking attack.

	Source	Destination
File	src/main/webapp/admin/adminlogin.jsp	src/main/webapp/admin/adminlogin.jsp
Line	29	29
Object	privilege	privilege

Code Snippet
File Name src/main/webapp/admin/adminlogin.jsp
Method response.addCookie(privilege);

```
....
29.                                    response.addCookie(privilege);
```

Suspected XSS

Query Path:
Java\Cx\Java Low Visibility\Suspected XSS Version:1

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-15 Information Output Filtering (P0)

Description

Suspected XSS\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=837
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 19 of src/main/webapp/vulnerability/Messages.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the rs=stmt.executeQuery method with rs, at line 14 of src/main/webapp/vulnerability/Messages.jsp. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/Messages.jsp	src/main/webapp/vulnerability/Messages.jsp
Line	14	19
Object	rs	print

Code Snippet

File Name src/main/webapp/vulnerability/Messages.jsp

Method rs=stmt.executeQuery("select * from UserMessages where recipient='"+session.getAttribute("user")+"'");

```
....
14.                rs=stmt.executeQuery("select * from UserMessages where
recipient='"+session.getAttribute("user")+"'");
```

File Name src/main/webapp/vulnerability/Messages.jsp

Method out.print("
"+rs.getString("subject")+"
");

```
....
19.                    out.print("<li><a
href='DisplayMessage.jsp?msgid="+rs.getString("msgid")+"
'>"+rs.getString("subject")+"</a></li>");
```

Suspected XSS\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=838
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 52 of src/main/webapp/changeCardDetails.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the id=session.getAttribute method with id, at line 29 of src/main/webapp/changeCardDetails.jsp. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/changeCardDetails.jsp	src/main/webapp/changeCardDetails.jsp
Line	29	52
Object	id	print

Code Snippet

File Name src/main/webapp/changeCardDetails.jsp

Method String id=session.getAttribute("userid").toString(); //Gets User ID

```
....
29.     String id=session.getAttribute("userid").toString();    //Gets
User ID
```

File Name src/main/webapp/changeCardDetails.jsp

Method out.print("

Return to Profile Page >>");

```
....
52.       out.print("<br/><br/><a
href='"+path+"/myprofile.jsp?id="+id+"'>Return to Profile Page
&gt;&gt;</a>");
```

Suspected XSS\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=839
Status	New

The application's out.print embeds untrusted data in the generated output with print, at line 35 of src/main/webapp/vulnerability/csrf/change-info.jsp. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.

The attacker would be able to alter the returned web page by saving malicious data in a data-store ahead of time. The attacker's modified data is then read from the database by the id=session.getAttribute method with

id, at line 27 of src/main/webapp/vulnerability/csrf/change-info.jsp. This untrusted data then flows through the code straight to the output web page, without sanitization.

This can enable a Stored Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/webapp/vulnerability/csrf/change-info.jsp	src/main/webapp/vulnerability/csrf/change-info.jsp
Line	27	35
Object	id	print

Code Snippet
File Name src/main/webapp/vulnerability/csrf/change-info.jsp
Method String id=session.getAttribute("userid").toString();

```
....
27.     String id=session.getAttribute("userid").toString();
```

File Name src/main/webapp/vulnerability/csrf/change-info.jsp

Method out.print("

Return to Profile Page >>");

```
....
35.      out.print("<br/><br/><a
href='"+path+"/myprofile.jsp?id="+id+"'>Return to Profile Page
&gt;&gt;</a>");
```

Plaintext Storage in a Cookie

Query Path:
Java\Cx\Java Low Visibility\Plaintext Storage in a Cookie Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage
OWASP Top 10 2013: A6-Sensitive Data Exposure
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description
Plaintext Storage in a Cookie\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=808
Status	New

The input ""username"" at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in line 39 is set as a value in a cookie by user at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in line 39.

	Source	Destination

File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	43	63
Object	""username""	user

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
43.         String user=request.getParameter("username").trim();
....
63.                                             Cookie username=new
Cookie("username",user);
```

Plaintext Storage in a Cookie\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=809
Status	New

The input ""password"" at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in line 39 is set as a value in a cookie by pass at src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java in line 39.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java	src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Line	44	64
Object	""password""	pass

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/LoginValidator.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.            String pass=request.getParameter("password").trim();
....
64.                                             Cookie password=new
Cookie("password",pass);
```

Stored Absolute Path Traversal

Query Path:
Java\Cx\Java Low Visibility\Stored Absolute Path Traversal Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.8 - Improper access control

OWASP Top 10 2013: A4-Insecure Direct Object References
OWASP Top 10 2017: A5-Broken Access Control

Description

Stored Absolute Path Traversal\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=829
Status	New

Method rs=stmt.executeQuery at line 24 of src/main/webapp/vulnerability/sqli/download_id.jsp gets dynamic data from the rs element. This element's value then flows through the code and is eventually used in a file path for local disk access in = at line 37 of src/main/webapp/vulnerability/sqli/download_id.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	24	37
Object	rs	File

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.                rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method String fileName = (new File(filePath)).getName();

```
....
37.                    String fileName = (new File(filePath)).getName();
```

Stored Absolute Path Traversal\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=830
Status	New

Method rs=stmt.executeQuery at line 24 of src/main/webapp/vulnerability/sqli/download_id_union.jsp gets dynamic data from the rs element. This element's value then flows through the code and is eventually used in a file path for local disk access in = at line 37 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	24	37
Object	rs	File

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.                rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method String fileName = (new File(filePath)).getName();

```
....
37.                    String fileName = (new File(filePath)).getName();
```

Stored Relative Path Traversal

Query Path:
Java\Cx\Java Low Visibility\Stored Relative Path Traversal Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.8 - Improper access control
OWASP Top 10 2013: A4-Insecure Direct Object References
OWASP Top 10 2017: A5-Broken Access Control

Description
Stored Relative Path Traversal\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=831
Status	New

Method rs=stmt.executeQuery at line 24 of src/main/webapp/vulnerability/sqli/download_id.jsp gets dynamic data from the rs element. This element's value then flows through the code and is eventually used in a file path for local disk access in File at line 32 of src/main/webapp/vulnerability/sqli/download_id.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	24	32
Object	rs	File

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.                rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method file = new File(file.getParent()+filePath);

```
....
32.                    file = new File(file.getParent()+filePath);
```

Stored Relative Path Traversal\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=832
Status	New

Method rs=stmt.executeQuery at line 24 of src/main/webapp/vulnerability/sqli/download_id_union.jsp gets dynamic data from the rs element. This element's value then flows through the code and is eventually used in a file path for local disk access in File at line 32 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	24	32
Object	rs	File

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.                rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method file = new File(file.getParent()+filePath);

```
....
32.                    file = new File(file.getParent()+filePath);
```

Creation of Temp File in Dir with Incorrect Permissions

Query Path:
Java\Cx\Java Low Visibility\Creation of Temp File in Dir with Incorrect Permissions Version:1

Categories

OWASP Top 10 2013: A7-Missing Function Level Access Control
OWASP Top 10 2017: A5-Broken Access Control

Description

Creation of Temp File in Dir with Incorrect Permissions\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=516
Status	New

A file is created on the file system by createNewFile in src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java at line 34 with potentially dangerous permissions.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	45	50
Object	File	createNewFile

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
45.              File f=new File(filePath);
....
50.                  if(f.createNewFile())
```

Potential Clickjacking on Legacy Browsers

Query Path:
JavaScript\Cx\JavaScript Low Visibility\Potential Clickjacking on Legacy Browsers Version:1

Description

Potential Clickjacking on Legacy Browsers\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=770
Status	New

The application does not protect the web page src/main/webapp/ForgotPassword.jsp from clickjacking attacks in legacy browsers, by using framebusting scripts.

	Source	Destination
File	src/main/webapp/ForgotPassword.jsp	src/main/webapp/ForgotPassword.jsp
Line	1	1
Object	CxJSNS_9d1dd3a9	CxJSNS_9d1dd3a9

Code Snippet
File Name src/main/webapp/ForgotPassword.jsp
Method

```
....
1.
```

Information Leak Through Shell Error Message

Query Path:
Java\Cx\Java Low Visibility\Information Leak Through Shell Error Message Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description

Information Leak Through Shell Error Message\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=800
Status	New

Sensitive information could leak to console output at src/main/webapp/vulnerability/baasm/URLRewriting.jsp in line 4 by the print element.

	Source	Destination
File	src/main/webapp/vulnerability/baasm/URLRewriting.jsp	src/main/webapp/vulnerability/baasm/URLRewriting.jsp
Line	4	4
Object	getId	print

Code Snippet
File Name src/main/webapp/vulnerability/baasm/URLRewriting.jsp
Method out.print("Your Session ID:"+session.getId());

```
....
4.    out.print("<b class='success'>Your Session
ID:</b>"+session.getId());
```

Missing X Frame Options

Query Path:
Java\Cx\Java Low Visibility\Missing X Frame Options Version:1

Categories

NIST SP 800-53: SC-18 Mobile Code (P2)
OWASP Top 10 2017: A6-Security Misconfiguration

Description

Missing X Frame Options\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=801
Status	New

The web-application does not properly utilize the "X-FRAME-OPTIONS" header to restrict embedding web-pages inside of a frame.

	Source	Destination
File	src/main/webapp/WEB-INF/web.xml	src/main/webapp/WEB-INF/web.xml
Line	1	1
Object	CxXmlConfigClass599923495	CxXmlConfigClass599923495

Code Snippet
File Name src/main/webapp/WEB-INF/web.xml
Method <!DOCTYPE web-app PUBLIC

```
....
1.  <!DOCTYPE web-app PUBLIC
```

Missing Content Security Policy

Query Path:
Java\Cx\Java Low Visibility\Missing Content Security Policy Version:1

Categories

OWASP Top 10 2017: A6-Security Misconfiguration

Description

Missing Content Security Policy\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=807
Status	New

A Content Security Policy is not explicitly defined within the web-application.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	36	36

Object	""text/html;charset=UTF-8""	""text/html;charset=UTF-8""

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
36.          response.setContentType("text/html;charset=UTF-8");
```

Reversible One Way Hash

Query Path:
Java\Cx\Java Low Visibility\Reversible One Way Hash Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Media Protection
NIST SP 800-53: SC-13 Cryptographic Protection (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description
Reversible One Way Hash\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=828
Status	New

The application is using a weak hashing primitive getInstance, in src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java at line 11

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java	src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java
Line	16	16
Object	""MD5""	getInstance

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java
Method public static String hashMe(String str)

```
....
16.              MessageDigest md = MessageDigest.getInstance("MD5");
```

Unrestricted File Upload

Query Path:
Java\Cx\Java Low Visibility\Unrestricted File Upload Version:1

Categories

FISMA 2014: Configuration Management

NIST SP 800-53: SC-18 Mobile Code (P2)
OWASP Top 10 2017: A1-Injection

Description

Unrestricted File Upload\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=840
Status	New

The uploaded file request in src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38 is stored in src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java at line 38 without validating the size of the file being saved.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java	src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java
Line	38	44
Object	request	getInputStream

Code Snippet

File Name src/main/java/org/cysecurity/cspf/jvl/controller/xxe.java

Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
38.      protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
....
44.            InputStream xml=request.getInputStream();
```

Use of Broken or Risky Cryptographic Algorithm

Query Path:
Java\Cx\Java Low Visibility\Use of Broken or Risky Cryptographic Algorithm Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.4 - Insecure communications
OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Configuration Management
NIST SP 800-53: SC-13 Cryptographic Protection (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description

Use of Broken or Risky Cryptographic Algorithm\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=841
Status	New

In hashMe, the application protects sensitive data using a cryptographic algorithm, getInstance, that is considered weak or even trivially broken, in src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java at line 11.

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java	src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java
Line	16	16
Object	getInstance	getInstance

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/model/HashMe.java
Method public static String hashMe(String str)

```
....
16.             MessageDigest md = MessageDigest.getInstance("MD5");
```

Use of Non Cryptographic Random

Query Path:
Java\Cx\Java Low Visibility\Use of Non Cryptographic Random Version:1

Description

Use of Non Cryptographic Random\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=842
Status	New

Method response.setHeader at line 38 of src/main/webapp/vulnerability/sqli/download_id.jsp uses a weak method nextInt to produce random values. These values might be used as personal identifiers, session tokens or cryptographic input; however, due to their insufficient randomness, an attacker may be able to derive their value.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id.jsp	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	38	38
Object	nextInt	nextInt

Code Snippet
File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method response.setHeader("Content-Disposition", "attachment; filename=\"" +new Random().nextInt(10000)+ "\"");

```
....
38.                 response.setHeader("Content-Disposition",
"attachment; filename=\"" +new Random().nextInt(10000)+ "\"");
```

Stored HTTP Response Splitting

Query Path:
Java\Cx\Java Stored\Stored HTTP Response Splitting Version:1

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description

Stored HTTP Response Splitting\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=893
Status	New

Method rs=stmt.executeQuery at line 24 of src/main/webapp/vulnerability/sqli/download_id_union.jsp gets user input from the rs element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in response.setHeader at line 38 of src/main/webapp/vulnerability/sqli/download_id_union.jsp. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/webapp/vulnerability/sqli/download_id_union.jsp	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	24	38
Object	rs	setHeader

Code Snippet

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method rs=stmt.executeQuery("select * from FilesList where fileid="+fileid);

```
....
24.                 rs=stmt.executeQuery("select * from FilesList where
fileid="+fileid);
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method response.setHeader("Content-Disposition", "attachment; filename=\"" + fileName + "\"");

```
....
38.                     response.setHeader("Content-Disposition",
"attachment; filename=\"" + fileName + "\"");
```

Portability Flaw In File Separator

Query Path:
Java\Cx\Java Best Coding Practice\Portability Flaw In File Separator Version:1

Description

Portability Flaw In File Separator\Path 1:

Severity	Information
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=901
Status	New

	Source	Destination
File	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java	src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Line	44	45
Object	""/""	File

Code Snippet
File Name src/main/java/org/cysecurity/cspf/jvl/controller/AddPage.java
Method protected void processRequest(HttpServletRequest request, HttpServletResponse response)

```
....
44.             String filePath=pagesDir+"/"+fileName;
45.             File f=new File(filePath);
```

Portability Flaw In File Separator\Path 2:

Severity	Information
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=902
Status	New

	Source	Destination
File	src/main/webapp/vulnerability/idor/download.jsp	src/main/webapp/vulnerability/idor/download.jsp
Line	13	13
Object	""/docs/""	File

Code Snippet
File Name src/main/webapp/vulnerability/idor/download.jsp
Method file = new File(file.getParent()+"/docs/"+filePath);

```
....
13.             file = new File(file.getParent()+"/docs/"+filePath);
```

Portability Flaw In File Separator\Path 3:

Severity	Information
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid

	=903
Status	New

	Source	Destination
File	src/main/webapp/WEB-INF/config.properties	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	8	32
Object	"jdbc:mysql://mysql:3306/"	File

Code Snippet
File Name src/main/webapp/WEB-INF/config.properties
Method # To change this license header, choose License Headers in Project Properties.

```
....
8.  dburl=jdbc:mysql://mysql:3306/
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method file = new File(file.getParent()+filePath);

```
....
32.                 file = new File(file.getParent()+filePath);
```

Portability Flaw In File Separator\Path 4:

Severity	Information
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=904
Status	New

	Source	Destination
File	src/main/webapp/WEB-INF/config.properties	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	8	37
Object	"jdbc:mysql://mysql:3306/"	File

Code Snippet
File Name src/main/webapp/WEB-INF/config.properties
Method # To change this license header, choose License Headers in Project Properties.

```
....
8.  dburl=jdbc:mysql://mysql:3306/
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp
Method String fileName = (new File(filePath)).getName();

```
....
37.                 String fileName = (new File(filePath)).getName();
```

Portability Flaw In File Separator\Path 5:

Severity	Information
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=905
Status	New

	Source	Destination
File	src/main/webapp/WEB-INF/config.properties	src/main/webapp/vulnerability/sqli/download_id.jsp
Line	8	41
Object	"jdbc:mysql://mysql:3306/"	FileInputStream

Code Snippet

File Name src/main/webapp/WEB-INF/config.properties

Method # To change this license header, choose License Headers in Project Properties.

```
....
8.  dburl=jdbc:mysql://mysql:3306/
```

File Name src/main/webapp/vulnerability/sqli/download_id.jsp

Method DataInputStream in = new DataInputStream(new FileInputStream(file));

```
....
41.                 DataInputStream in = new DataInputStream(new
FileInputStream(file));
```

Portability Flaw In File Separator\Path 6:

Severity	Information
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=906
Status	New

	Source	Destination
File	src/main/webapp/WEB-INF/config.properties	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	8	37
Object	"jdbc:mysql://mysql:3306/"	File

Code Snippet

File Name	src/main/webapp/WEB-INF/config.properties
Method	# To change this license header, choose License Headers in Project Properties.

```
....
8.  dburl=jdbc:mysql://mysql:3306/
```

File Name	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method	String fileName = (new File(filePath)).getName();

```
....
37.                 String fileName = (new File(filePath)).getName();
```

Portability Flaw In File Separator\Path 7:

Severity	Information
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=907
Status	New

	Source	Destination
File	src/main/webapp/WEB-INF/config.properties	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	8	32
Object	"jdbc:mysql://mysql:3306/"	File

Code Snippet

File Name	src/main/webapp/WEB-INF/config.properties
Method	# To change this license header, choose License Headers in Project Properties.

```
....
8.  dburl=jdbc:mysql://mysql:3306/
```

File Name	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Method	file = new File(file.getParent()+filePath);

```
....
32.                 file = new File(file.getParent()+filePath);
```

Portability Flaw In File Separator\Path 8:

Severity	Information
Result State	To Verify
Online Results	http://HAPPYY-LAPTOP/CxWebClient/ViewerMain.aspx?scanid=1000038&projectid=36&pathid=908
Status	New

	Source	Destination
File	src/main/webapp/WEB-INF/config.properties	src/main/webapp/vulnerability/sqli/download_id_union.jsp
Line	8	41
Object	"jdbc:mysql://mysql:3306/"	FileInputStream

Code Snippet

File Name src/main/webapp/WEB-INF/config.properties

Method # To change this license header, choose License Headers in Project Properties.

```
....
8.  dburl=jdbc:mysql://mysql:3306/
```

File Name src/main/webapp/vulnerability/sqli/download_id_union.jsp

Method DataInputStream in = new DataInputStream(new FileInputStream(file));

```
....
41.                 DataInputStream in = new DataInputStream(new
FileInputStream(file));
```

Connection String Injection

Risk

What might happen

If an attacker could manipulate the application's connection string to the database server, they might be able do any of the following:

- Damage application performance (by increasing the MIN POOL SIZE)
- Tamper with the network connection (for example, via TRUSTED CONNECTION)
- Direct the application to the attacker's bogus database
- Discover the password to the system account on the database (by a brute-force attack).

Cause

How does it happen

In order to communicate with the database, or with another external server (for example, Active Directory), the application dynamically constructs a connection string. This connection string includes untrusted data, which may be controlled by a malicious user. Since it is not constrained or properly sanitized, the untrusted data could be used to maliciously manipulate the connection string.

General Recommendations

How to avoid it

- Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
  - Data type

- Size
- Range
- Format
- Expected values

- Do not allow users to control the database connection string. Avoid dynamically building connection strings based on untrusted data, especially user input.
- Store all connection strings in appropriate configuration mechanisms. If it is necessary to dynamically construct a connection string at runtime, do not include untrusted data directly in the connection string; instead, allow users to select from pre-defined connection strings.

Source Code Examples

Java

Connect to Database using User Input

```
private Connection openConnection_unsafe(HttpServletRequest request)
		throws ServletException, SQLException {
	Connection conn = null;
	String dbServer = request.getParameter("Server");
	String dbName = request.getParameter("Database");

	String userUrl = String.format(JDBC_URL_TEMPLATE, dbServer, dbName);

	try {
		conn = DriverManager.getConnection(userUrl, DB_USER, DB_PASSWORD);
	} catch (Exception ex) {
		handleExceptions(ex);
	}
	return conn;
}
```

Select Pre-Configured Database Connection by User Input

```
private Connection openConnection_SafeSelection(HttpServletRequest request)
		throws ServletException, SQLException {
	Connection conn = null;
	int appId = Integer.parseInt(request.getParameter("AppId"));

	String userUrl;
	switch(appId) {
		case APP_ID1:
			userUrl = JDBC_URL_APP1;
			break;
		case APP_ID2:
			userUrl = JDBC_URL_APP2;
			break;
		case APP_ID3:
			userUrl = JDBC_URL_APP3;
			break;
		default:
			userUrl = JDBC_URL_DEFAULT;
	}

	try {
```

```
            conn = DriverManager.getConnection(userUrl, DB_USER, DB_PASSWORD);
      } catch (Exception ex) {
            handleExceptions(ex);
      }
      return conn;
}
```

Second Order SQL Injection

Risk

What might happen

An attacker could directly access all of the system's data. The attacker would likely be able to steal any sensitive information stored by the system, including private user information, credit card details, proprietary business data, and any other secret data. Likewise, the attacker could possibly modify or erase existing data, or even add new bogus data. In some scenarios, it may even be possible to execute code on the database.

In addition to disclosing or altering confidential information directly, this vulnerability might also be used to achieve secondary effects, such as bypassing authentication, subverting security checks, or forging a data trail.

Further increasing the likelihood of exploit is the fact that this flaw is easy for attackers to find, and easy to exploit.

Cause

How does it happen

The application stores and manages data in a database, by submitting a textual SQL query to the database engine for processing. The application creates the query by simple string concatenation, embedding untrusted data. However, there is no separation between data and code; furthermore, the embedded data is neither checked for data type validity nor subsequently sanitized. Thus, the untrusted data could contain SQL commands, or modify the intended query. The database would interpret the altered query and commands as if they originated from the application, and execute them accordingly.

In order to exploit this vulnerability, an attacker would load the malicious payload into the database, typically via forms on other web pages. Afterwards, the application reads this data from the database, and embeds it within the SQL query, as SQL commands.

General Recommendations

How to avoid it

- Validate all untrusted data, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns.
- In particular, check for:
  - Data type
  - Size
  - Range
  - Format
  - Expected values.
- Restrict access to database objects and functionality, according to the Principle of Least Privilege.
- Do not use dynamically concatenate strings to construct SQL queries.
- Prefer using DB Stored Procedures for all data access, instead of ad-hoc dynamic queries.
- Instead of unsafe string concatenation, use secure database components such as parameterized queries and object bindings (for example, commands and parameters).
- Alternatively, an even better solution is to use an ORM library, in order to pre-define and encapsulate the allowed commands enabled for the application, instead of dynamically accessing the database directly. In this way the code plane and data plane should be isolated from each other.
- Data validation can be performed effectively using a secure library, such as OWASP's Encoder or ESAPI libraries.

- Prefer using `PreparedStatement` for parameterizing the queries, or even better `CallableStatement`. Add dynamic data via the `.set*()` methods, instead of string concatenation.
- Consider using an ORM package, such as `Hibernate`, `myBatis`, or others.

Source Code Examples

Java

Create SQL Query Using String Concatenation Based on Arbitrary Data

```
public String getActiveContact_Unsafe(HttpServletRequest request)
		throws ServletException, IOException {
	String userNameFromDB;
	String contact;

	try {
		Connection conn = getConnection();

		CallableStatement readNameStmt = conn.prepareCall("{call getActiveUser (?)}");
		readNameStmt.registerOutParameter(1, java.sql.Types.VARCHAR);
		readNameStmt.execute();
		userNameFromDB = readNameStmt.getString(1);

		String sql = "SELECT [Contact] FROM [AppUsers] WHERE [UserName] = '" +
userNameFromDB + "' " ;
		Statement readDetailsStmt = conn.createStatement();
		ResultSet data = readDetailsStmt.executeQuery(sql);

		contact = data.getString(1);
	} catch (SQLException ex) {
		handleExceptions(ex);
	}
	finally {
		closeQuietly(data);
		closeQuietly(readNameStmt);
		closeQuietly(readDetailsStmt);
		closeQuietly(conn);
	}

	return contact;
}
```

Build PreparedStatement to Call Stored Procedure and Set Data to Parameters

```
public String getActiveContact_SafeParameterizedQuery(HttpServletRequest request)
		throws ServletException, IOException {
	String userNameFromDB;
	String contact;

	String sqlStoredProc = "{call getUserId (?, ?)}";

	try {
		Connection conn = getConnection();

		CallableStatement readNameStmt = conn.prepareCall("{call getActiveUser (?)}");
		readNameStmt.registerOutParameter(1, java.sql.Types.VARCHAR);
		readNameStmt.execute();
```

```
            userNameFromDB = readNameStmt.getString(1);

            CallableStatement readDetailsStmt = conn.prepareCall(sqlStoredProc);

            readDetailsStmt.setString(1, userNameFromDB);
            readDetailsStmt.registerOutParameter(2, java.sql.Types.VARCHAR);

            readDetailsStmt.execute();
            contact = readDetailsStmt.getString(2);
      } catch (SQLException ex) {
            handleExceptions(ex);
      }
      finally {
            closeQuietly(readNameStmt);
            closeQuietly(readDetailsStmt);
            closeQuietly(conn);
      }

      return contact;
}
```

XPath Injection

Risk

What might happen

An attacker that can modify the XPath query with an arbitrary expression will be able to control which nodes in the XML document will be selected, and thus what data the application will process. This can have various effects depending on the type of XML document and its usage, including retrieval of secret information, control of application flow, modification of sensitive data, reading arbitrary files, or even authentication bypass, impersonation, and privilege escalation.

Cause

How does it happen

The application queries an XML document by using a textual XPath query. The application creates the query by simply concatenating strings, including untrusted data potentially under an attacker's control. Since the external data is neither checked for data type validity nor subsequently sanitized, the data could be maliciously crafted to cause the application to select the wrong information from the XML document.

General Recommendations

How to avoid it

- Validate all external data, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
  - Data type
  - Size
  - Range
  - Format
  - Expected values
- Avoid making the XPath query dependent on external data.
- If it is absolutely necessary to include untrusted data in the query, the data must at least be first properly validated or sanitized.
- If possible, it is preferable to map XPath queries to external parameters, maintaining separation between data and code.
- Input sanitization, while not preferred, can be done safely when required with OWASP's `ESAPI` library, using `Encoder.encodeForXPath()`. Additional libraries may also provide similar functionality.
- Prefer parameterized XPath queries over data sanitization, to fully separate the input from the expression. Precompile the XPath query using `XPath.compile()`, after defining the variable resolution mechanism using `XPath.setXPathVariableResolver()` and a custom `XPathVariableResolver` class. Afterwards the variables can be provided to the `XPathVariableResolver`, and evaluated using the resulting `XPathExpression.evaluate()`.

Source Code Examples

Java

Select XML Node using XPath Expression With User Input

```
private String readUserGroup_Unsafe(HttpServletRequest request)
            throws ServletException, IOException {
      String resultGroup = "";

      String userId = request.getParameter("UserID");
      String expr = "//USERS/USER[UserID/text()='" + userId + "']/GROUP/text()";

      try {
            DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document docUsers = builder.parse(new File(USERS_XML_FILE));

            XPath navigator = XPathFactory.newInstance().newXPath();
            resultGroup = navigator.evaluate(expr, docUsers);
      } catch (Exception ex) {
            handleExceptions(ex);
      }

      return resultGroup;
}
```

Evaluate XPath Expression with Sanitized Input

```
public String readUserGroup_SafeSanitized(HttpServletRequest request)
            throws ServletException, IOException {
      String resultGroup = "";

      String param = request.getParameter("UserID");

      // Sanitize input using OWASP's ESAPI Encoder library
      Encoder esapiEncoder = new DefaultEncoder();
      String sanitizedUserId = esapiEncoder.encodeForXPath(param);

      /*
      // Alternatively, can validate input by casting to integer
      String sanitizedUserId = Integer.toString(Integer.parseInt(param));
      */

      String expr = "//USERS/USER[UserID/text()='" + sanitizedUserId + "']/GROUP/text()";

      try {
            DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document docUsers = builder.parse(new File(USERS_XML_FILE));

            XPath navigator = XPathFactory.newInstance().newXPath();
            resultGroup = navigator.evaluate(expr, docUsers);
      } catch (Exception ex) {
            handleExceptions(ex);
      }

      return resultGroup;
}
```

Evaluate XPath Expression with Parameterized Variables

```
public String readUserGroup_SafeParameterized(HttpServletRequest request)
```

```
            throws ServletException, IOException {
      String resultGroup = "";

      String param = request.getParameter("UserID");

      Encoder esapiEncoder = new DefaultEncoder();
      String sanitizedUserId = esapiEncoder.encodeForXPath(param);

      String expr = "//USERS/USER[UserID/text()=$userId]/GROUP/text()";

      try {
            DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document docUsers = builder.parse(new File(USERS_XML_FILE));

            // Define variables and resolver
            final Map<String, Object> vars = new HashMap<String, Object>();
            XPath navigator = XPathFactory.newInstance().newXPath();
            navigator.setXPathVariableResolver(new XPathVariableResolver() {
                  public Object resolveVariable(QName name) {
                        return vars.get(name.getLocalPart());
                  }
            });
            XPathExpression groupExpression = navigator.compile(expr);

            // Evaluate expression with safe parameter
            vars.put("userId", sanitizedUserId);
            resultGroup = groupExpression.evaluate(docUsers);
      } catch (Exception ex) {
            handleExceptions(ex);
      }

      return resultGroup;
}
```

Reflected XSS All Clients

Risk

What might happen

A successful XSS exploit would allow an attacker to rewrite web pages and insert malicious scripts which would alter the intended output. This could include HTML fragments, CSS styling rules, arbitrary JavaScript, or references to third party code. An attacker could use this to steal users' passwords, collect personal data such as credit card details, provide false information, or run malware. From the victim's point of view, this is performed by the genuine website, and the victim would blame the site for incurred damage.

The attacker could use social engineering to cause the user to send the website modified input, which will be returned in the requested web page.

Cause

How does it happen

The application creates web pages that include untrusted data, whether from user input, the application's database, or from other external sources. The untrusted data is embedded directly in the page's HTML, causing the browser to display it as part of the web page. If the input includes HTML fragments or JavaScript, these are displayed too, and the user cannot tell that this is not the intended page. The vulnerability is the result of directly embedding arbitrary data without first encoding it in a format that would prevent the browser from treating it like HTML or code instead of plain text.

Note that an attacker can exploit this vulnerability either by modifying the URL, or by submitting malicious data in the user input or other request fields.

General Recommendations

How to avoid it

- Fully encode all dynamic data, regardless of source, before embedding it in output.
- Encoding should be context-sensitive. For example:
  - HTML encoding for HTML content
  - HTML Attribute encoding for data output to attribute values
  - JavaScript encoding for server-generated JavaScript
- It is recommended to use the platform-provided encoding functionality, or known security libraries for encoding output.
- Implement a Content Security Policy (CSP) with explicit whitelists for the application's resources only.
- As an extra layer of protection, validate all untrusted data, regardless of source (note this is not a replacement for encoding). Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
  - Data type
  - Size
  - Range
  - Format
  - Expected values
- In the `Content-Type` HTTP response header, explicitly define character encoding (charset) for the entire page.
- Set the `HTTPOnly` flag on the session cookie for "Defense in Depth", to prevent any successful XSS exploits from stealing the cookie.

Source Code Examples

Java

Returning Data To Clients Without Encoding

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");

      PrintWriter out = response.getWriter();
      String loc = request.getParameter("location");

      out.println("<h1> Location: " + loc + "<h1>");
}
```

Returning Data to Clients After Encoding The User Input

```
// Using HtmlEscapers by Google Guava

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");

      PrintWriter out = response.getWriter();
      String loc = request.getParameter("location");
      String escapedLocation = HtmlEscapers.htmlEscaper().escape(loc);

      out.println("<h1> Location: " + escapedLocation + "<h1>");
}
```

SQL Injection

Risk

What might happen

An attacker could directly access all of the system's data. The attacker would likely be able to steal any sensitive information stored by the system, including private user information, credit card details, proprietary business data, and any other secret data. Likewise, the attacker could possibly modify or erase existing data, or even add new bogus data. In some scenarios, it may even be possible to execute code on the database.

In addition to disclosing or altering confidential information directly, this vulnerability might also be used to achieve secondary effects, such as bypassing authentication, subverting security checks, or forging a data trail.

Further increasing the likelihood of exploit is the fact that this flaw is easy for attackers to find, and easy to exploit.

Cause

How does it happen

The application stores and manages data in a database, by submitting a textual SQL query to the database engine for processing. The application creates the query by simple string concatenation, embedding untrusted data. However, there is no separation between data and code; furthermore, the embedded data is neither checked for data type validity nor subsequently sanitized. Thus, the untrusted data could contain SQL commands, or modify the intended query. The database would interpret the altered query and commands as if they originated from the application, and execute them accordingly.

Note that an attacker can exploit this vulnerability either by modifying the URL, or by submitting malicious data in the user input or other request fields.

General Recommendations

How to avoid it

- Validate all untrusted data, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns.
- In particular, check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values.
- Restrict access to database objects and functionality, according to the Principle of Least Privilege.
- Do not use dynamically concatenate strings to construct SQL queries.
- Prefer using DB Stored Procedures for all data access, instead of ad-hoc dynamic queries.
- Instead of unsafe string concatenation, use secure database components such as parameterized queries and object bindings (for example, commands and parameters).
- Alternatively, an even better solution is to use an ORM library, in order to pre-define and encapsulate the allowed commands enabled for the application, instead of dynamically accessing the database directly. In this way the code plane and data plane should be isolated from each other.
- Data validation can be performed effectively using a secure library, such as OWASP's Encoder or ESAPI libraries.

- Prefer using `PreparedStatement` for parameterizing the queries, or even better `CallableStatement`. Add dynamic data via the `.set*()` methods, instead of string concatenation.
- Consider using an ORM package, such as `Hibernate`, `myBatis`, or others.

Source Code Examples

Java

Create SQL query using string concatenation

```
public int getUserId_Unsafe(HttpServletRequest request)
		throws ServletException, IOException {
	int userId = 0;

	String userName = request.getParameter("UserName");
	String sql = "SELECT [UserID] FROM [AppUsers] WHERE [UserName] = '" + userName + "' " ;

	try {
		Connection conn = getConnection();
		Statement stmt = conn.createStatement();
		ResultSet data = stmt.executeQuery(sql);

		userId = data.getInt(1);
	} catch (SQLException ex) {
		handleExceptions(ex);
	}
	finally {
		closeQuietly(data);
		closeQuietly(stmt);
		closeQuietly(conn);
	}

	return userId;
}
```

Create SQL query using Sanitized Username

```
public int getUserId_SafeSanitized(HttpServletRequest request)
		throws ServletException, IOException {
	int userId = 0;

	String userName = request.getParameter("UserName");

	// Sanitize input using OWASP's ESAPI Encoder library
	//   Still not complete solution!
	Encoder esapiEncoder = new DefaultEncoder();
	String sanitizedUserName = esapiEncoder.encodeForSQL(new OracleCodec(), userName);

	String sql = "SELECT [UserID] FROM [AppUsers] WHERE [UserName] = '" + sanitizedUserName
+ "' " ;

	try {
		Connection conn = getConnection();
		Statement stmt = conn.createStatement();
		ResultSet data = stmt.executeQuery(sql);

		userId = data.getInt(1);
```

```
	} catch (SQLException ex) {
		handleExceptions(ex);
	}
	finally {
		closeQuietly(data);
		closeQuietly(stmt);
		closeQuietly(conn);
	}

	return userId;
}
```

Build PreparedStatement to call Stored Procedure and set input to parameters

```
public int getUserId_SafeParameterizedQuery(HttpServletRequest request)
		throws ServletException, IOException {
	int userId = 0;

	String userName = request.getParameter("UserName");
	String sqlStoredProc = "{call getUserId (?, ?)}";

	try {
		Connection conn = getConnection();
		CallableStatement stmt = conn.prepareCall(sqlStoredProc);

		stmt.setString(1, userName);
		stmt.registerOutParameter(2, java.sql.Types.INTEGER);

		stmt.execute();
		userId = stmt.getInt(2);
	} catch (SQLException ex) {
		handleExceptions(ex);
	}
	finally {
		closeQuietly(stmt);
		closeQuietly(conn);
	}

	return userId;
}
```

Stored XSS

Risk

What might happen

A successful XSS exploit would allow an attacker to rewrite web pages and insert malicious scripts which would alter the intended output. This could include HTML fragments, CSS styling rules, arbitrary JavaScript, or references to third party code. An attacker could use this to steal users' passwords, collect personal data such as credit card details, provide false information, or run malware. From the victim’s point of view, this is performed by the genuine website, and the victim would blame the site for incurred damage.

An attacker could use legitimate access to the application to submit modified data to the application’s data-store. This would then be used to construct the returned web page, triggering the attack.

Cause

How does it happen

The application creates web pages that include untrusted data, whether from user input, the application’s database, or from other external sources. The untrusted data is embedded directly in the page's HTML, causing the browser to display it as part of the web page. If the input includes HTML fragments or JavaScript, these are displayed too, and the user cannot tell that this is not the intended page. The vulnerability is the result of directly embedding arbitrary data without first encoding it in a format that would prevent the browser from treating it like HTML or code instead of plain text.

In order to exploit this vulnerability, an attacker would load the malicious payload into the data-store, typically via regular forms on other web pages. Afterwards, the application reads this data from the data-store, and embeds it within the web page as displayed for another user.

General Recommendations

How to avoid it

- Fully encode all dynamic data, regardless of source, before embedding it in output.
- Encoding should be context-sensitive. For example:
  - HTML encoding for HTML content
  - HTML Attribute encoding for data output to attribute values
  - JavaScript encoding for server-generated JavaScript
- It is recommended to use the platform-provided encoding functionality, or known security libraries for encoding output.
- Implement a Content Security Policy (CSP) with explicit whitelists for the application's resources only.
- As an extra layer of protection, validate all untrusted data, regardless of source (note this is not a replacement for encoding). Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
  - Data type
  - Size
  - Range
  - Format
  - Expected values
- In the `Content-Type` HTTP response header, explicitly define character encoding (charset) for the entire page.
- Set the `HTTPOnly` flag on the session cookie for "Defense in Depth", to prevent any successful XSS exploits from stealing the cookie.

Source Code Examples

Java

Data Obtained From The Execution of an SQL Command Is Output to a Label

```
public static void getUser(PerparedStatement stmt, int userId) throws SQLException {
      stmt = connection.perpareStatement("SELECT * FROM user WHERE user = ?");
      stmt.setInt(1, userId);
      Label label = new Label();
      ResultSet rs = stmt.executeQuery();

      String lastName = "";
      while (rs.next()) {
            lastName += rs.getString("Lname") + ", ";
      }
      label.setText("User's last name: " + lastName);
}
```

The Output String is Encoded to Hardcoded String Before It Is Displayed in The Label

```
public static void getUser(PreparedStatement stmt, int userId) throws SQLException {

      Label label = new Label();

    stmt = connection.perpareStatement("SELECT * FROM user WHERE user = ?");
      stmt.setInt(1, userId);
      ResultSet rs = stmt.executeQuery();

      while (rs.next()) {
            lastName = StringEscapeUtils.escapeHtml4(rs.getString("Lname"));
      }
      label.setText("User's last name is: " + lastName);
}
```

Absolute Path Traversal

Risk

What might happen

An attacker could define arbitrary file path for the application to use, potentially leading to:

- Stealing sensitive files, such as configuration or system files
- Overwriting files such as program binaries, configuration files, or system files
- Deleting critical files, causing denial of service (DoS).

Cause

How does it happen

The application uses user input in the file path for accessing files on the application server's local disk.

General Recommendations

How to avoid it

1. Ideally, avoid depending on dynamic data for file selection.
2. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
3. Accept dynamic data only for the filename, not for the path and folders.
4. Ensure that file path is fully canonicalized.
5. Explicitly limit the application to use a designated folder that is separate from the applications binary folder.
6. Restrict the privileges of the application's OS user to necessary files and folders. The application should not be able to write to the application binary folder, and should not read anything outside of the application folder and data folder.

Source Code Examples

Cross Site History Manipulation

Risk

What might happen

An attacker could compromise the browser's Same Origin Policy and violate a user's privacy, by manipulating the browser's History object in JavaScript. This could allow the attacker in certain situations to detect whether the user is logged in, track the user's activity, or infer the state of other conditional values. This may also enhance Cross Site Request Forgery (XSRF) attacks, by leaking the result of the initial attack.

Cause

How does it happen

Modern browsers expose the user's browsing history to local JavaScript as a stack of previously visited URLs. While the browsers enforce a strict Same Origin Policy (SOP) to prevent pages from one website from reading visited URLs on other websites, the History object does leak the size of the history stack. Using only this information, in some situations the attacker can discover the results of certain checks the application server performs.

For example - if the application redirects an unauthenticated user to the login page, a script on another website can detect whether or not the user is logged in, by checking the length of the history object. This is done by first making a request to the page being redirected to (e.g. "/login"), then replacing that with a redirecting page that only redirects users if that user is not yet authenticated (e.g. "/profile") - if the length of history object remains the same, redirection has occurred back to the page being redirected to, and the history stack is not updated. If the history stack length is updated, that means the page did not redirect the user, causing the new page to be stored in the history stack.

This information leakage is enabled when the application redirects the user's browser based on the value of some condition, the state of the user's server-side session. e.g. whether the user is authenticated to the application, if the user has visited a certain page with specific parameters, or the value of some application data.

Note that this issue does not affect all browsers, and depends on the browser's implementation of Javascript's history object behavior.

General Recommendations

How to avoid it

- Add the response header "X-Frame-Options: DENY" to all sensitive pages in the application, to protect against the IFrame version of XSHM in modern browser versions.
- Add a random value to all redirection URLs as a parameter to ensure that they are unique when inserted into the history stack

---

Source Code Examples

Java

Example of code that leaks the variable state via browser history

```
If (!isAuthenticated)
    response.sendRedirect("Login.jsp");
```

Example code that prevents history leakage via random token

```
if (request.getParameter("r") == null)
    response.sendRedirect("Login.jsp?r=" + (new Random()).nextInt());

If (!isAuthenticated)
    response.sendRedirect("Login.jsp?r=" + (new Random()).nextInt());
```

Download of Code Without Integrity Check

Risk

What might happen

At best, code that fails an integrity check may be damaged, altered or may not match the intended code that it originally was, resulting in unexpected behavior. At worst, attackers who are able to compromise the loaded code, such as locally or via a Man-in-the-Middle attack, may alter the loaded code either at storage or in transit, which may result in malicious code execution and significant system compromise.

Cause

How does it happen

Integrity signatures, as derived from any data set, can allow a recepient to identify that the received data set is the one they intended to obtain. For externally loaded code this is particularly important, as such code can be compromised via access to local storage, Man-in-the-Middle attacks and more.

General Recommendations

How to avoid it

Perform strict integrity checks to ensure code obtained from external sources is verified, and has a known and trusted signature.

Source Code Examples

Java

Calculate Hash of File And Compare to Trusted Hash

```
byte[] dataBytes = new byte[1024];
MessageDigest md = MessageDigest.getInstance("SHA-256");
int nread = 0;

//Create byte hash from file
FileInputStream fis = new FileInputStream(file);
while ((nread = fis.read(dataBytes)) != -1) {
      md.update(dataBytes, 0, nread);
};
fis.close();

//Convert bytes to hex
byte[] mdbytes = md.digest();
StringBuffer sb = new StringBuffer();
for (int i = 0; i < mdbytes.length; i++) {
      sb.append(Integer.toString((mdbytes[i] & 0xff) + 0x100, 16).substring(1));
}
String sha256ofFile = sb.toString();

if (sha256ofFile.equals(TRUSTED_SHA256_OF_FILE)) {
      // Handle trusted files here
}
else {
      // Handle untrusted files here
```

```
}
```

Class Function Loaded From Untested External File

```
private void LoadClass (File file) {
      URL url = file.toURI().toURL();
      URL[] urls = new URL[]{url};
      ClassLoader cl = new URLClassLoader(urls);
      Class cls = cl.loadClass("com.packagename.classname");
      Constructor cons = cls.getConstructor();
      Object simpleClassObj = cons.newInstance();
      Method method = cls.getMethod("functionname");
      method.invoke(simpleClassObj);
}
```

External Control of System or Config Setting

Risk

What might happen

If an external user can control the application configuration or environment settings, she can disrupt application service, or cause the application to behave unexpectedly, possibly creating additional vulnerabilities in the application, or even causing the application to perform malicious actions.

Cause

How does it happen

The application receives unvalidated input from the user, and directly sets that value to the application configuration or environment settings, without properly sanitizing or constraining the input.

General Recommendations

How to avoid it

- Do not allow user input, or otherwise untrusted data, to control sensitive values, particularly configuration or environment settings.
- Always validate data received from users, before using it for any internal use.

Source Code Examples

Java

User Input Controlling System Properties

```
public void init() {
    string dbUrl = request.getParameter("dbUrl");
    string remoteWsUrl = request.getParameter("wsUrl");

    Properties props = System.getProperties();
    props.setProperty("dbUrl", dbUrl);
    props.setProperty("wsUrl", remoteWsUrl);

}
```

Read Properties from Configuration File

```
public void init() {
    Properties props = new Properties(System.getProperties());
    props.loadFromXml(new FileInputStream(CONFIG_FILE_PATH));

}
```

Heap Inspection

Risk

What might happen

All variables stored by the application in unencrypted memory can potentially be retrieved by an unauthorized user, with privileged access to the machine. For example, a privileged attacker could attach a debugger to the running process, or retrieve the process's memory from the swapfile or crash dump file. Once the attacker finds the user passwords in memory, these can be reused to easily impersonate the user to the system.

Cause

How does it happen

String variables are immutable - in other words, once a string variable is assigned, its value cannot be changed or removed. Thus, these strings may remain around in memory, possibly in multiple locations, for an indefinite period of time until the garbage collector happens to remove it. Sensitive data, such as passwords, will remain exposed in memory as plaintext with no control over their lifetime.

While it may still be possible to retrieve data from memory, even if it uses a mutable container that is cleared, or retrieve a decryption key and decrypt sensitive data from memory - layering sensitive data with these types of protection would significantly increase the required effort to do so. By setting a high bar for retrieving sensitive data from memory, and reducing the amount and exposure of sensitive data in memory, an adversary is significantly less likely to succeed in obtaining valuable data.

General Recommendations

How to avoid it

When it comes to avoiding Heap Inspection, it is important to note that, given any read access to memory or a memory dump of an application, it is always likely to disclose some sensitive data to an adversary - these suggestions are part of defense-in-depth principles for protection of sensitive data in cases where such memory read access is successfully obtained. These recommendations will enable significant reduction in the lifespan and exposure of sensitive data in memory; however - given enough time, effort and unlimited access to memory, they will only go so far in protecting sensitive data being used by the application. The only way to handle Heap Inspection issues is to minimize and reduce data exposure, and obscure it in memory wherever possible.

- Do not store sensitive data, such as passwords or encryption keys, in memory in plain-text, even for a short period of time.
- Prefer to use specialized classes that store encrypted data in memory to ensure it cannot be trivially retrieved from memory.
- When required to use sensitive data in its raw form, temporarily store it in mutable data types, such as byte arrays, to reduce readability from memory, and then promptly zeroize the memory locations, to reduce exposure duration of this data while in memory.
- Ensure that memory dumps are not exchanged with untrusted parties, as even by ensuring all of the above - it may still be possible to reverse-engineer encrypted containers, or retrieve bytes of sensitive data from memory and rebuild it.
- In Java, do not store passwords in immutable strings - prefer using an encrypted memory object, such as SealedObject.

Source Code Examples

Java

Plaintext Password in Immutable String

```
class Heap_Inspection
{
  private String password;

  public void setPassword(String password)
 {
      this.password = password;
  }
}
```

Password Protected in Memory

```
class Heap_Inspection_Fixed
{
  private SealedObject password;

  public void setPassword(Character[] input)
 {
      Key key = getKeyFromConfiguration();
            Cipher c = Cipher.getInstance(CIPHER_NAME);
            c.init(Cipher.ENCRYPT_MODE, key);
            List<Character> characterList = Arrays.asList(input);
            password = new SealedObject((Serializable) characterList, c);
            Arrays.fill(input, '\0'); // Zero out input. Will also overwrite the values in
characterList by reference.
  }
}
```

HTTP Response Splitting

Risk

What might happen

If the header setting code is of a vulnerable version, an attacker could:

- Arbitrarily change the application server's response header to a victim's HTTP request by manipulating headers
- Arbitrarily change the application server's response body by injecting two consecutive line breaks, which may result in Cross-Site Scripting (XSS) attacks
- Cause cache poisoning, potentially controlling any site's HTTP responses going through the same proxy as this application.

Cause

How does it happen

Since user input is being used in an HTTP response header, an attacker could include NewLine characters to make the header look like multiple headers with engineered content, potentially making the response look like multiple responses (for example, by engineering duplicate content-length headers). This can cause an organizational proxy server to provide the second, engineered response to a victim's subsequent request; or, if the proxy server also performs response caching, the attacker can send an immediate subsequent request to another site, causing the proxy server to cache the engineered response as a response from this second site and to later serve the response to other users.

Many modern web frameworks mitigate this issue, by offering sanitization for new line characters in strings inserted into headers by default. However, since many older versions of web frameworks fail to automatically mitigate this issue, manual sanitization of input may be required.

General Recommendations

How to avoid it

1. Validate all input, regardless of source (including cookies). Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
2. Additionally, remove or URL-encode all special (non-alphanumeric) user input before including it in the response header.
3. Make sure to use an up-to-date framework.

Source Code Examples

Java

User Input Affects a Response Header

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      // Vulnerable in legacy versions of Java
    String username = request.getParameter("username");
    response.setContentType("text/html");
    Cookie cookie = new Cookie("user", username);
    cookie.setMaxAge(3600);
    response.addCookie(cookie);
}
```

User Input URL Encoded Prior Setting A Response Header

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      // Defense In Depth - URLEncode user input before setting a response header
    String username = request.getParameter("username");
    response.setContentType("text/html");
    Cookie cookie = new Cookie("user", URLEncoder.encode(username, "UTF-8"));
    cookie.setMaxAge(3600);
    response.addCookie(cookie);
}
```

HttpOnlyCookies In Config

Risk

What might happen

Cookies that contain the user's session identifier, and other sensitive application cookies, are typically accessible by client-side scripts, such as JavaScript. Unless the web application explicitly prevents this using the "httpOnly" cookie flag, these cookies could be read and accessed by malicious client scripts, such as Cross-Site Scripting (XSS). This flag would mitigate the damage done in case XSS vulnerabilities are discovered, according to Defense in Depth.

Cause

How does it happen

The web application framework, by default, does not set the "httpOnly" flag for the application's sessionid cookie and other sensitive application cookies. Likewise, the application does not explicitly use the "httpOnly" cookie flag, thus allowing client scripts to access the cookies by default.

General Recommendations

How to avoid it

- Always set the "httpOnly" flag for any sensitive server-side cookie. - It is highly recommended to implement HTTP Strict Transport Security (HSTS) in order to ensure that the cookie will be sent over a secured channel. - Configure the application to always use "httpOnly" cookies in the site-wide configuration file. - In your application's web.xml configuration file, add `<http-only>true</http-only>` under the `<cookie-config>` element, in the `<session-config>` element.

Source Code Examples

Java

Insecure Cookie Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
            http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
            version="3.0">

    <session-config>
        <cookie-config>
            <http-only>false</http-only>
        </cookie-config>
    </session-config>

</web-app>
```

Web.xml Configuration File with Secure Cookies

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
            http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
            version="3.0">

    <session-config>
        <session-timeout>30</session-timeout>
        <cookie-config>
            <http-only>true</http-only>
            <secure>true</secure>
        </cookie-config>
    </session-config>

</web-app>
```

HttpOnlyCookies

Risk

What might happen

Cookies that contain the user's session identifier, and other sensitive application cookies, are typically accessible by client-side scripts, such as JavaScript. Unless the web application explicitly prevents this using the "httpOnly" cookie flag, these cookies could be read and accessed by malicious client scripts, such as Cross-Site Scripting (XSS). This flag would mitigate the damage done in case XSS vulnerabilities are discovered, according to Defense in Depth.

Cause

How does it happen

The web application framework, by default, does not set the "httpOnly" flag for the application's sessionid cookie and other sensitive application cookies. Likewise, the application does not explicitly use the "httpOnly" cookie flag, thus allowing client scripts to access the cookies by default.

General Recommendations

How to avoid it

- Always set the "httpOnly" flag for any sensitive server-side cookie. - It is highly recommended to implement HTTP Strict Transport Security (HSTS) in order to ensure that the cookie will be sent over a secured channel. - Explicitly set the "httpOnly" flag for each cookie set by the application. - In particular, explicitly call the `Cookie.setHttpOnly()` method, on any cookie being added to the response. - Consider configuring the web application framework to automatically set httpOnly to all cookies, by adding `<http-only>true</http-only>` under the `<cookie-config>` element, in your application's web.xml configuration file. - If the cookie is set to the response via the `.setHeader()` method with the `"Set-Cookie"` header name, append `";httpOnly;"` to the end of the cookie value.

Source Code Examples

Java

Creating Cookie in Code

```
private void setCookiesToResponse( HttpServletResponse response, String deptId) {
      Cookie appCookie = new Cookie("department", deptId);
      response.addCookie(appCookie);
}
```

Explicitly setting the HttpOnly flag on Cookies

```
private void setCookiesToResponse( HttpServletResponse response, String deptId ) {
      Cookie appCookie = new Cookie("department", deptId);
```

```
        appCookie.setMaxAge(360);
        appCookie.setSecure(true);

        appCookie.setHttpOnly(true);

        response.addCookie(appCookie);
}
```

Manually Writing Cookie Headers

```
private void setCookiesToResponse( HttpServletResponse response, String deptId ) {
        response.setHeader("Set-Cookie", "department=" + deptId);
}
```

Manual Cookie Headers with HttpOnly Attribute

```
private void setCookiesToResponse( HttpServletResponse response, String deptId ) {
        response.setHeader("Set-Cookie", "department=" + deptId + ";httpOnly;secure;Expires=Thu,
31 Dec 2020 12:00:00 GMT;");
}
```

Improper Restriction of XXE Ref

Risk

What might happen

An application that will parse and replace DTD entity references, in an XML document that the user controls, can allow an attacker to craft an XML document to read arbitrary server files. This XML document could contain an XML entity reference, which refers to an embedded DTD entity definition that points to any local file. This would enable the attacker to retrieve any arbitrary system file on the server.

Cause

How does it happen

An attacker could upload an XML document that contains a DTD declaration, in particular an entity defintion that refers to a local file on the server's disk, e.g. `<!ENTITY xxe SYSTEM "file:///c:/boot.ini">`. The attacker would then include an XML entity reference that refers back to that entity definition, e.g. `<div>&xxe;</div>` . If the parsed XML document is then returned to the user, the result will include the contents of the sensitive system file.

This is caused by the XML parser, which is configured to automatically parse DTD declarations and resolve entity references, instead of disabling both DTD and external references altogether.

General Recommendations

How to avoid it

Generic Guidance:

- Avoid processing user input directly, where possible.
- If necessary to receive XML from the user, ensure the XML parser is restricted and constrained.
- In particular, disable DTD parsing and resolving of entities. Apply a strict XML schema on the server, and validate the input XML accordingly.

Specific Recommendations:

- Use safe XML parsers, and disable DTD parsing and entity resolving.
- Do not enable DTD parsing or entity resolving.

Source Code Examples

Java

Vulnerable XML Parsing with XMLStreamReader

```
@Override
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {

   String strContent = request.getParameter("xmlInput");
   PrintWriter out = response.getWriter();
```

```
   Reader reader = new StringReader(strContent);
   XMLInputFactory factory = XMLInputFactory.newInstance();
   XMLStreamReader xmlReader = null;

   try {
      xmlReader = factory.createXMLStreamReader(reader);

      while (xmlReader.hasNext()) {
         int event = xmlReader.next();
         if (event == XMLStreamConstants.START_ELEMENT) {
            if (xmlReader.getLocalName().equals("username")) {
               String username = xmlReader.getElementText();
               if (username != null)
                  out.println(username);
            }
         }
      }
   } catch (Exception e) {
      //Handle Exception
   } finally {
      try{
         out.close();
         xmlReader.close();
         reader.close();
      } catch (Exception ex) {
         //Handle Exception
      }
   }
}

//Example of Malicious XML input:
//Read from "secret.txt" in the parent directory
/*
<!DOCTYPE note [
<!ENTITY xxe SYSTEM "../secret.txt">
]>
<credentials>
      <username>&xxe;</username>
</credentials>
*/
```

Disabling DTD Entities with XMLStreamReader

```
@Override
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {

   String strContent = request.getParameter("xmlInput");
   PrintWriter out = response.getWriter();
   Reader reader = new StringReader(strContent);
   XMLInputFactory factory = XMLInputFactory.newInstance();
   XMLStreamReader xmlReader = null;

   try {

      factory.setProperty(XMLInputFactory.IS_SUPPORTING_EXTERNAL_ENTITIES,
false);//Remediation
    factory.setProperty(XMLInputFactory.SUPPORT_DTD, false); //Remediation
     xmlReader = factory.createXMLStreamReader(reader);

      while (xmlReader.hasNext()) {
         int event = xmlReader.next();
         if (event == XMLStreamConstants.START_ELEMENT) {
            if (xmlReader.getLocalName().equals("username")) {
```

```
                String username = xmlReader.getElementText();
                if (username != null)
                   out.println(username);
             }
          }
       }
    } catch (Exception e) {
       //Handle Exception
    } finally {
       try{
          out.close();
          xmlReader.close();
          reader.close();
       } catch (Exception ex) {
          //Handle Exception
       }
    }
}
```

Input Path Not Canonicalized

Risk

What might happen

An attacker could define arbitrary file path for the application to use, potentially leading to:

- Stealing sensitive files, such as configuration or system files
- Overwriting files such as program binaries, configuration files, or system files
- Deleting critical files, causing denial of service (DoS).

Cause

How does it happen

The application uses user input in the file path for accessing files on the application server's local disk.

General Recommendations

How to avoid it

1. Ideally, avoid depending on dynamic data for file selection.
2. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
3. Accept dynamic data only for the filename, not for the path and folders.
4. Ensure that file path is fully canonicalized.
5. Explicitly limit the application to use a designated folder that is separate from the applications binary folder.
6. Restrict the privileges of the application's OS user to necessary files and folders. The application should not be able to write to the application binary folder, and should not read anything outside of the application folder and data folder.

Source Code Examples

Plaintext Storage of a Password

Risk

What might happen

Passwords that are stored in a database in plaintext form, could be easily retrieved by an attacker that has read access to it.

Cause

How does it happen

Sensitive information was stored in the database as plaintext.

General Recommendations

How to avoid it

Generic Guidance:

- Do not store any sensitive information, such as database passwords, in plain text.
- Safely encrypt application passwords, using proper methods of encryption. If the original password must be retrievable, e.g. to connect to the database, use AES-CBC or GCM with a strong encryption key and random IV.
- Implement proper key management, including dynamically generating random keys, protecting keys, and replacing keys as necessary.
- Alternatively, encrypt the password using platform mechanisms or a hardware device.

Source Code Examples

Java

Plaintext Database Password

```
public Connection createConnection(Properties connectionProperties) {

    string user = connectionProperties.getProperty("user");
    string password = connectionProperties.getProperty("password");

    Connection conn = DriverManager.getConnection(CONN_STRING, user, password);

    return conn;
}
```

Encrypted Database Password

```
public Connection createConnection(Properties connectionProperties) {

    string user = connectionProperties.getProperty("user");
```

```
    string password = connectionProperties.getProperty("password");

    // Call encryption framework to decrypt password with protected key
    password = decryptPassword(password);

    try {
        Connection conn = DriverManager.getConnection(CONN_STRING,
                                                      user, password);
    }
    catch (SQLException e)  {
        handleException(e);
    }

    return conn;
}
```

Privacy Violation

Risk

What might happen

A user's personal information could be stolen by a malicious programmer, or an attacker that intercepts the data.

Cause

How does it happen

The application sends user information, such as passwords, account information, or credit card numbers, outside the application, such as writing it to a local text or log file or sending it to an external web service.

General Recommendations

How to avoid it

1. Personal data should be removed before writing to logs or other files.
2. Review the need and justification of sending personal data to remote web services.

Source Code Examples

Java

Leaking a Password Back to the User Constitutes a Privacy Violation

```
public void doPost (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      PrintWriter out = response.getWriter();
      HttpSession session = request.getSession();
      boolean isAuthenticated = session.getAttribute("isAuthenticated");
      if (isAuthenticated) {
            byte[] password = request.getParameter("password").getBytes();
            updatePassword(session, password);
      out.println("New password is " + (new String(password));
      } else {
            out.println("Authentication Failed");
      }
}
```

Missing HSTS Header

Risk

What might happen

Failure to set an HSTS header and provide it with a reasonable "max-age" value of at least one year may leave users vulnerable to Man-in-the-Middle attacks.

Cause

How does it happen

Many users browse to websites by simply typing the domain name into the address bar, without the protocol prefix. The browser will automatically assume that the user's intended protocol is HTTP, instead of the encrypted HTTPS protocol.

When this initial request is made, an attacker can perform a Man-in-the-Middle attack and manipulate it to redirect users to a malicious web-site of the attacker's choosing. To protect the user from such an occurence, the HTTP Strict Transport Security (HSTS) header instructs the user's browser to disallow use of an unsecure HTTP connection to the the domain associated with the HSTS header.

Once a browser that supports the HSTS feature has visited a web-site and the header was set, it will no longer allow communicating with the domain over an HTTP connection.

Once an HSTS header was issued for a specific website, the browser is also instructed to prevent users from manually overriding and accepting an untrusted SSL certificate for as long as the "max-age" value still applies. The recommended "max-age" value is for at least one year in seconds, or 31536000.

General Recommendations

How to avoid it

- Before setting the HSTS header - consider the implications it may have:
  - Forcing HTTPS will prevent any future use of HTTP, which could hinder some testing
  - Disabling HSTS is not trivial, as once it is disabled on the site, it must also be disabled on the browser
- Set the HSTS header either explicitly within application code, or using web-server configurations.
- Ensure the "max-age" value for HSTS headers is set to 31536000 to ensure HSTS is strictly enforced for at least one year.
- Include the "includeSubDomains" to maximize HSTS coverage, and ensure HSTS is enforced on all sub-domains under the current domain
  - Note that this may prevent secure browser access to any sub-domains that utilize HTTP; however, use of HTTP is very severe and highly discouraged, even for websites that do not contain any sensitive information, as their contents can still be tampered via Man-in-the-Middle attacks to phish users under the HTTP domain.
- Once HSTS has been enforced, submit the web-application's address to an HSTS preload list - this will ensure that, even if a client is accessing the web-application for the first time (implying HSTS has not yet been set by the web-application), a browser that respects the HSTS preload list would still treat the web-application as if it had already issued an HSTS header. Note that this requires the server to have a trusted SSL certificate, and issue an HSTS header with a maxAge of 1 year (31536000)
- Note that this query is designed to return one result per application. This means that if more than one vulnerable response without an HSTS header is identified, only the first identified instance of this issue will be highlighted as a result. If a misconfigured instance of HSTS is identified (has a short lifespan, or is missing the "includeSubDomains" flag), that result will be flagged. Since HSTS is required to be

enforced across the entire application to be considered a secure deployment of HSTS functionality, fixing this issue only where the query highlights this result is likely to produce subsequent results in other sections of the application; therefore, when adding this header via code, ensure it is uniformly deployed across the entire application. If this header is added via configuration, ensure that this configuration applies to the entire application.

- Note that misconfigured HSTS headers that do not contain the recommended max-age value of at least one year or the "includeSubDomains" flag will still return a result for a missing HSTS header.

Source Code Examples

Java

Setting an HSTS Header in an HTTP Response

```
response.setHeader("Strict-Transport-Security", "max-age=31536000; includeSubDomains");
```

XML

Spring - Setting HSTS via Configuration Web.Xml

```
<http>
      <!-- This is a default value -->
      <headers>
            <hsts
                  include-subdomains="true"
                  max-age-seconds="31536000" />
      </headers>
</http>
```

JBoss - Setting HSTS via Configuration Web.Xml

```
<system.webServer>
    <httpProtocol>
        <customHeaders>
            <add name="Strict-Transport-Security" value="max-age=31536000;
includeSubDomains"/>
        </customHeaders>
    </httpProtocol>
</system.webServer>
```

Tomcat - Enable Header Security in Web.Xml

```
<filter>
      <filter-name>httpHeaderSecurity</filter-name>
    <filter-class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-class>
      <init-param>
```

```
            <param-name>hstsMaxAgeSeconds</param-name>
            <param-value>31536000</param-value>
      </init-param>
      <init-param>
            <param-name>includeSubDomains</param-name>
            <param-value>true</param-value>
      </init-param>
</filter>
```

Trust Boundary Violation

Risk

What might happen

Code that reads from Session variables may trust them as server-side variables, but they may have been tainted by user inputs. This can lead to tampering with parameters used to authenticate or authorize users. Further, tainted Session variables offer an additional attack surface against the application - if untrusted data taints a Session variable, and that Session variable is then used elsewhere without sanitization as if it were trusted, it could lead to further attacks such as Cross-Site Scripting, SQL Injection and more.

Cause

How does it happen

Server-side Session variables, or objects, are values assigned to a specific session, which is associated with a specific user. Often, they hold data relevant to that user's session, such as specific identifiers, user-type, authorization, authentication information and more. As such, the paradigm often associated to the Session object is that its contents can be trusted, as users cannot generally set these values themselves.

The application places user input, which is untrusted data, in the server-side Session object, which is considered a trusted location. This could lead developers to treat untrusted data as trusted.

General Recommendations

How to avoid it

1. Validate and sanitize all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
2. Don’t mix untrusted user input with trusted data.

Source Code Examples

Java

Setting User Role by Relying on User Input, Allowing Users to Tamper Its Value and Elevate Their Privilege

```
public void doPost (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      HttpSession session = request.getSession();
      String username = request.getParameter("username");
      byte[] password = request.getParameter("password").getBytes();
      if (isAuthenticated(username, password)) {
            String role = request.getParameter("role"); // Role can be tampered by user
            session.setAttribute("isAuthenticated",true);
            session.setAttribute("role", role);
            // Render page //
```

```
	} else {
		// Authentication error //
	}
}
```

Derive User Role from An Internal Mechanism Based On The Current User

```
public void doPost (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
	HttpSession session = request.getSession();
	String username = request.getParameter("username");
	byte[] password = request.getParameter("password").getBytes();
	if (isAuthenticated(username, password)) {
		String role = getUserRole(username); // Role is not derived from user input, but
some post authentication mechanism
		session.setAttribute("isAuthenticated",true);
		session.setAttribute("role", role);
		// Render page //
	} else {
		// Authentication error //
	}
}
```

SSRF

Risk

What might happen

An attacker can abuse this flaw to make arbitrary requests, originating from the application server. This can be exploited to scan internal services; proxy attacks into a protected network; bypass network controls; download unauthorized files; access internal services and management interfaces; and possibly control the contents of requests and even steal server credentials.

Cause

How does it happen

The application accepts a URL (or other data) from the user, and uses this to make a request to another remote server.

However, the attacker can inject an arbitrary URL into the request, causing the application to connect to any server the attacker wants. Thus, the attacker can abuse the application to gain access to services that would not otherwise be accessable, and cause the request to ostensibly originate from the application server.

General Recommendations

How to avoid it

- Do not connect to arbitrary services based on user input.
- If possible, the application should have the user's browser retrieve the desired information directly.
- If it is necessary for the application to proxy the request on the server, explicitly whitelist the allowed target URLs, and do not include any sensitive server information.

Source Code Examples

Java

Retrieve and Display Contents of URL

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
  if (request.getParameterMap().containsKey("url")) {
    String url = request.getParameter("url");
    PrintWriter out = response.getWriter();
    URL u = new URL(url);
    InputStreamReader sr = new InputStreamReader(u.openConnection().getInputStream());
    BufferedReader reader = new BufferedReader(sr);
    String line = reader.readLine();
    while (line != null) {
      out.write(line);
      line = reader.readLine();
    }
  }
}
```

Validate and Redirect User's Browser

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
  if (request.getParameterMap().containsKey("url")) {
    String url = request.getParameter("url");
    if (url.startsWith("/") && !url.startsWith("//")) {
      response.sendRedirect(url);
    } else {
      response.sendRedirect("/");
    }
  }
}
```

Unchecked Input for Loop Condition

Risk

What might happen

An attacker could input a very high value, potentially causing a denial of service (DoS).

Cause

How does it happen

The application performs some repetitive task in a loop, and defines the number of times to perform the loop according to user input. A very high value could cause the application to get stuck in the loop and to be unable to continue to other operations.

General Recommendations

How to avoid it

Ideally, don't base a loop on user-provided data. If it is necessary to do so, the user input must be first validated and its range should be limited.

Source Code Examples

Java

Loop Condition Is Not Bounded By Any Value

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    int loopCount = 0;
    try{
        loopCount = Integer.parseInt(request.getParameter("loopCount"));
    } catch(NumberFormatException e){
        return DEFAULT_VAL;
    }
    for(int i=0; i < loopCount; i++){
        //Do Something
    }
}
```

Loop Condition is Bounded With MAX_LOOPS

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    int loopCount = 0;
    try{
        loopCount = Integer.parseInt(request.getParameter("loopCount"));
    } catch(NumberFormatException e){
```

```
        return DEFAULT_VAL;
    }
    if(loopCount > MAX_LOOPS){
        loopCount = MAX_LOOPS;
    }
    for(int i=0; i < loopCount; i++){
        //Do Something
    }
}
```

Use of a One Way Hash with a Predictable Salt

Risk

What might happen

If an attacker gains access to the hashed passwords, she would likely be able to reverse the hash due to this weakness, and retrieve the original password. Once the passwords are discovered, the attacker can impersonate the users, and take full advantage of their privileges and access their personal data. Furthermore, this would likely not be discovered, as the attacker is being identified solely by the victims' credentials.

Cause

How does it happen

Typical cryptographic hashes, such as SHA-1 and MD5, are incredibly fast. Combined with attack techniques such as precomputed Rainbow Tables, it is relatively easy for attackers to reverse the hashes, and discover the original passwords. Lack of a unique, random salt added to the password makes brute force attacks even simpler.

General Recommendations

How to avoid it

Generic Guidance: - Always use strong, modern algorithms for encryption, hashing, and so on. - Do not use weak, outdated, or obsolete algorithms. - Ensure you select the correct cryptographic mechanism according to the specific requirements.

Specific Recommendations: - Passwords should be protected using a password hashing algorithm, instead of a general cryptographic hash. This includes adaptive hashes such as bcrypt, scrypt, PBKDF2 and Argon2. - Tune the work factor, or cost, of the adaptive hash function according to the designated environment and risk profile. - Do not use a regular cryptographic hash, such as SHA-1 or MD5, to protect passwords, as these are too fast. - If it is necessary to use a common hash to protect passwords, add several bytes of unique, random data ("salt") to the password before hashing it. Store the salt with the hashed password, and do not reuse the same salt for multiple passwords.

Source Code Examples

Java

Unsalted Hashed Password

```
private String protectPassword(String password) {
      byte[] data = password.getBytes();
      byte[] hash = null;

      MessageDigest md = MessageDigest.getInstance("MD5");
      hash = md.digest(data);

      return Base64.getEncoder().encodeToString(hash);
}
```

Fast Hash with Salt

```
private String protectPassword(String password) {
      byte[] data = password.getBytes("UTF-8");
      byte[] hash = null;

      try {
            MessageDigest md = MessageDigest.getInstance("SHA-1");

            SecureRandom rand = new SecureRandom();
            byte[] salt = new byte[32];
            rand.nextBytes(salt);

            md.update(salt);
            md.update(data);

            hash = md.digest();
    }
      catch (GeneralSecurityException gse) {
            handleCryptoErrors(gse);
    }
      finally {
            Arrays.fill(data, 0);
    }

      return Base64.getEncoder().encodeToString(hash);
}
```

Slow, Adaptive Password Hash

```
private String protectPassword(String password) {
      byte[] data = password.getBytes("UTF-8");
      byte[] hash = null;

      try {
            SecureRandom rand = new SecureRandom();
            byte[] salt = new byte[32];
            rand.nextBytes(salt);

            SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
            PBEKeySpec spec = new PBEKeySpec(data, salt, ITERATION_COUNT, KEY_LENGTH);
            // ITERATION_COUNT should be configured by environment, KEY_LENGTH should be 256
            SecretKey key = skf.generateSecret(spec);

            hash = key.getEncoded();
    }
      catch (GeneralSecurityException gse) {
            handleCryptoErrors(gse);
    }
      finally {
            Arrays.fill(data, 0);
    }

      return Base64.getEncoder().encodeToString(hash);
}
```

XSRF

Risk

What might happen

An attacker could cause the victim to perform any action for which the victim is authorized, such as transferring funds from the victim’s account to the attacker’s. The action will be logged as being performed by the victim, in the context of their account, and potentially without their knowledge that this action has occurred.

Cause

How does it happen

The application performs some action that modifies database contents, based purely on HTTP request content, and does not require per-request renewed authentication (such as transaction authentication or a synchronizer token), instead relying solely on session authentication. This means that an attacker could use social engineering to cause a victim to browse to a link which contains a transaction request to the vulnerable application, submitting that request from the user's browser. Once the application receives the request, it would trust the victim’s session, and would perform the action. This type of attack is known as Cross-Site Request Forgery (XSRF or CSRF).

A Cross-Site Request Forgery attack relies on the trust between a server and an authenticated client. By only validating the session, the server ensures that a request has emerged from a client's web-browser. However, any website may submit GET and POST requests to other websites, to which the browser will automatically add the session token if it is in a cookie. This cross-site request can then be trusted as arriving from the user's browser, but does not validate that it was their intent was to make this request.

General Recommendations

How to avoid it

Mitigating XSRF requires an additional layer of authentication that is built into the request validation mechanism. This mechanism would attach an additional token that only applies to the given user; this token would be available within the user's web-page, but will not be attached automatically to a request from a different website (e.g. not stored in a cookie). Since the token is not automatically attached to the request, and is not available to the attacker, and is required by the server to process the request, it would be completely impossible for the attacker to fill in a valid cross-site form that contains this token.

Many platforms offer built-in XSRF mitigation functionality which should be used, and perform this type of token management under the hood. Alternatively, use a known or trusted library which adds this functionality.

If implementing XSRF protection is required, this protection should adhere to the following rules:

- Any state altering form (Create, Update, Delete operations) should enforce XSRF protection, by adding an XSRF token to every state altering form submission on the client.
- An XSRF token should be generated, and be unique per-user per-session (and, preferably, per request).
- The XSRF token should be inserted into the client side form, and be submitted to the server as part of the form request. For example, it could be a hidden field in an HTML form, or a custom header added by a Javascript request.
- The XSRF token in the request body or custom header must then be verified as belonging to the current user by the server, before a request is authorized and processed as valid.

Source Code Examples

Java

Change E-Mail Functionality Vulnerable to XSRF, Allowing Attackers to Hijack User Accounts with the "Forgot Password" Functionality

```
public void doPost (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      HttpSession session = request.getSession();
      String email = request.getParameter("NewEmail");
      if (isAuthenticated(session)) {
            if (isValidEmail(email)) {
                  // update Email
            }
            else {
                  // reject invalid Email
            }
      } else {
                  // Require authentication
      }
}
```

Change E-Mail Functionality Protected By XSRF Tokens, By Embedding Token in Form and Comparing It To a Session Variable

```
public void doPost (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      HttpSession session = request.getSession();
      String email = request.getParameter("NewEmail");
      String csrfFormToken = request.getParameter("csrfToken");
      String csrfSessionToken = session.getAttribute("csrfToken"); /* Token should be attached
to every state-changing form;


            * must be cryptographically secure and unique per-session
                                    * at the very least (could also be unique per-request)
                                    */

      if (csrfFormToken == null || !csrfFormToken.equals(csrfSessionToken)) {
            // Reject request and log potential XSRF attempt
    } else {
            if (isAuthenticated(session)) {
                  if (isValidEmail(email)) {
                        // update Email
                  }
                  else {
                        // reject invalid Email
                  }
            } else {
                  // Require authentication
            }
      }
}
```

Blind SQL Injections

Risk

What might happen

An attacker could directly access all of the system's data. The attacker would likely be able to steal any sensitive information stored by the system, including private user information, credit card details, proprietary business data, and any other secret data. Likewise, the attacker could possibly modify or erase existing data, or even add new bogus data. In some scenarios, it may even be possible to execute code on the database.

In addition to disclosing or altering confidential information directly, this vulnerability might also be used to achieve secondary effects, such as bypassing authentication, subverting security checks, or forging a data trail.

Further increasing the likelihood of exploit is the fact that this flaw is easy for attackers to find, and easy to exploit.

In this case, while the actual exploit is constrained to single bit of information at a time, it is still possible to eventually retrieve all data from the system, though this process is more time consuming and would be rather noisy. Note that other consequences, such as data modification and creation, code execution, etc. are unaffected, and still equally exploitable.

Cause

How does it happen

The application stores and manages data in a database, by submitting a textual SQL query to the database engine for processing. The application creates the query by simple string concatenation, embedding untrusted data. However, there is no separation between data and code; furthermore, the embedded data is neither checked for data type validity nor subsequently sanitized. Thus, the untrusted data could contain SQL commands, or modify the intended query. The database would interpret the altered query and commands as if they originated from the application, and execute them accordingly.

In this case, the attacker does not need to rely on the application returning data from the database. Instead, it is possible to leverage existing tools that perform a series of boolean tests based on varying user input, relying only on the existence of application errors to indicate server state. Thus, the full database content can gradually be obtained, one bit at a time.

General Recommendations

How to avoid it

- Validate all untrusted data, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns.
- In particular, check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values.
- Restrict access to database objects and functionality, according to the Principle of Least Privilege.
- Do not use dynamically concatenate strings to construct SQL queries.
- Prefer using DB Stored Procedures for all data access, instead of ad-hoc dynamic queries.
- Instead of unsafe string concatenation, use secure database components such as parameterized queries and object bindings (for example, commands and parameters).

- Alternatively, an even better solution is to use an ORM library, in order to pre-define and encapsulate the allowed commands enabled for the application, instead of dynamically accessing the database directly. In this way the code plane and data plane should be isolated from each other.
- Do not allow the user to dynamically provide the name of the queried table. Furthermore, if possible, completely avoid dynamically specifying table names.
- Ensure that all exceptions are properly handled, without leaking information on the errors, server state, or that an error occurred at all.
- Data validation can be performed effectively using a secure library, such as OWASP's Encoder or ESAPI libraries.
- Prefer using `PreparedStatement` for parameterizing the queries, or even better `CallableStatement`. Add dynamic data via the `.set*()` methods, instead of string concatenation.
- Consider using an ORM package, such as `Hibernate`, `myBatis`, or others.

Source Code Examples

Java

Create SQL INSERT query using string concatenation

```
public String acceptUserTOS_Unsafe(HttpServletRequest request)
		throws ServletException, IOException {
	int rowCount = 0;

	String userName = request.getParameter("UserName");
	String sql = "INSERT INTO [UserTOS] ([UserName], [AcceptDate]) VALUES ('" + userName +
"', CURRENT_TIMESTAMP) " ;

	try {
		Connection conn = getConnection();
		Statement stmt = conn.createStatement();

		rowCount = stmt.executeUpdate(sql);
	} catch (SQLException ex) {
		handleExceptions(ex);
	}
	finally {
		closeQuietly(stmt);
		closeQuietly(conn);
	}

	if (rowCount > 0)
		return "Success";
	else
		return "Failed!";
}
```

Build PreparedStatement to call write-only Stored Procedure and set input to parameters

```
public String acceptUserTOS_SafeParameterizedQuery(HttpServletRequest request)
		throws ServletException, IOException {
	int rowCount = 0;

	String userName = request.getParameter("UserName");
```

```
	String sqlStoredProc = "{call acceptUserTOS(?)}";

	try {
		Connection conn = getConnection();
		CallableStatement stmt = conn.prepareCall(sqlStoredProc);

		stmt.setString(1, userName);

		rowCount = stmt.executeUpdate();
	} catch (SQLException ex) {
		handleExceptions(ex);
	}
	finally {
		closeQuietly(stmt);
		closeQuietly(conn);
	}

	if (rowCount > 0)
		return "Success";
	else
		return "Failed!";
}
```

Creation of Temp File in Dir with Incorrect Permissions

Risk

What might happen

Files with implicit or dangerous permissions may allow attackers to retrieve sensitive data from the contents of these files, tamper their contents or potentially execute them.

Cause

How does it happen

A file or directory is created with dangerous permissions, either by setting these permissions explicitly or relying on unsafe default permissions.

General Recommendations

How to avoid it

- Always create files with permissions being set explicitly
- Never set dangerous permissions on files
    - Always consider the principle of least privilege when determining who may read, write or execute a file, if these permissions are to be granted at all

---

Source Code Examples

Java

Writing A File with Implicit Permissions

```
File tempFile = File.createTempFile(TEMP_FILE_PREFIX,TEMP_FILE_SUFFIX, new
File(TEMP_FOLDER));
FileWriter fw = new FileWriter(tempFile);
fw.write(CONTENT);
```

Writing A File with Explicit Permissions

```
File tempFile = File.createTempFile(TEMP_FILE_PREFIX,TEMP_FILE_SUFFIX, new
File(TEMP_FOLDER));
FileWriter fw = new FileWriter(tempFile);
tempFile.setExecutable(false);
tempFile.setReadable(true);
tempFile.setWritable(true);
fw.write(CONTENT);
```

Data Leak Between Sessions

Risk

What might happen

A race condition can cause erratic or unexpected application behavior. Additionally, if a race condition can be influenced directly by users, an attacker may choose to replay a race condition until they obtain a desired result, allowing them to induce certain application behavior that is not part of its intentional design.

Cause

How does it happen

Most modern environments, and web in particular, require a high level of concurrent actions, invoking many instances of an object, or a singleton object multiple times, using an internal scheduler that decides which thread should be operating at the moment. This scheduler is often out of a developer's control, and may halt and resume concurrent threads. These threads often operate over shared resources, which are part of a functionality that is being invoked simultaneously across multiple logic flows; for example - static member variables are shared across all instances of an object type, while in singleton objects (such as servlets) any member variable is shared across all calls to the singleton.

When working in an environment with concurrency, failure to ensure all operations on a shared resource are done atomically and in a synchronized manner may result in threads or processes overriding each other's actions or outputs, often contradicting intended logic. For example, consider the "Time of Check, Time of Use" problem (TOCTOU) - if two threads attempt to increment the same counter, thread 1 reads the value n, and is then halted by the scheduler, thread 2 will read the value n, and then both threads will attempt to update the value to n+1, where logic would dictate that it would be n+2. Had this counter been properly synchronized, thread 2 would have to wait for thread 1 to complete its entire atomic operation of checking and updating its value.

General Recommendations

How to avoid it

- Consider allocating independent resources for concurrent logical processes, unless required.
- Where resource sharing is required, utilize a locking mechanism when handling shared resources to ensure that actions are performed atomically by the current process, so that these logical processes are not overridden or disrupted by any other concurrent process.
- When dealing with static member variables - always consider that these are shared across all instances of an object
- When dealing with singleton objects (such as servlets) being invoked by multiple concurrent logical processes - always synchronize either the entire singleton or its member variables; otherwise any thread processing the singleton's static methods may cause TOCTOU issues

---

Source Code Examples

Java

Threads Incrementing The Same Variable Without Locking

```
public static class FooIncrement {
	static int foo = 0;
	public static int generateFoo() throws InterruptedException {
		IncrementFooThread[] threadArray = new IncrementFooThread[10000];
		for (int i = 0; i < threadArray.length; i++) {
			threadArray[i] = new IncrementFooThread();
			threadArray[i].start();
		}
		for (IncrementFooThread threaderThread : threadArray) {
			threaderThread.join();
		}
		return foo;
	}
	public static class IncrementFooThread extends Thread {
		public void run()
		{
			foo++; //two threads may separately read the same value of foo before any
of them update it,
			   //resulting in two threads setting the same value of foo
		}

	}
}
```

Adding A Lock Object to Synchronize a Shared Resource

```
public static class FooIncrement {
	static int foo = 0;
	static Object lock = new Object();
	public static int generateFoo() throws InterruptedException {
		IncrementFooThread[] threadArray = new IncrementFooThread[10000];
		for (int i = 0; i < threadArray.length; i++) {
			threadArray[i] = new IncrementFooThread();
			threadArray[i].start();
		}
		for (IncrementFooThread threaderThread : threadArray) {
			threaderThread.join();
		}
		return foo;
	}
	public static class IncrementFooThread extends Thread {
		public void run()
		{
			synchronized (lock) {
				foo++; //foo is atomically incremented
			}
		}

	}
```

HttpServlet are Singletons - Therefore, Objects Set as Their Attributes Will Leak Between Calls to The Same Servlet

```
public class ShopServlet extends HttpServlet {
    private User user;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
	HttpSession session = request.getSession(true);
		user = (User) session.getAttribute("user");
		// Rest of doGet()
```

```
    }
}
```

Avoiding Attributes in HttpServlets

```
public class ShopServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
      HttpSession session = request.getSession(true);
            User user = (User) session.getAttribute("user");
            // Rest of doGet()
    }
}
```

Exposure of System Data

Risk

What might happen

System data can provide attackers with valuable insights on systems and services they are targeting - any type of system data, from service version to operating system fingerprints, can assist attackers to hone their attack, correlate data with known vulnerabilities or focus efforts on developing new attacks against specific technologies.

Cause

How does it happen

System data is read and subsequently exposed where it might be read by untrusted entities.

General Recommendations

How to avoid it

Consider the implications of exposure of the specified input, and expected level of access to the specified output. If not required, consider removing this code, or modifying exposed information to exclude potentially sensitive system data.

Source Code Examples

Java

Leaking Environment Variables in JSP Web-Page

```
String envVarValue = System.getenv(envVar);
if (envVarValue == null) {
    out.println("Environment variable is not defined:");
    out.println(System.getenv());
} else {
    //[..]
};
```

Improper Exception Handling

Risk

What might happen

An attacker could maliciously cause an exception that could crash the application, potentially resulting in a denial of service (DoS) or unexpected behavior under certain erroneous conditions. Exceptions may also occur without any malicious intervention, resulting in general instability.

Cause

How does it happen

The application performs some operation, such as database or file access, that could throw an exception. Since the application is not designed to properly handle the exception, the application could crash.

General Recommendations

How to avoid it

Any method that could cause an exception should be wrapped in a try-catch block that:

- Explicitly handles expected exceptions
- Includes a default solution to explicitly handle unexpected exceptions

---

Source Code Examples

Java

Loading a Library without Catching

```
public static void loadLib() {
      System.loadLibrary(LIB_NAME); // If LIB_NAME does not exist, an unhandled exception will
be thrown
}
```

Handle All Possible Exceptions within the Error-Prone Method

```
public static void loadLib() {
      try {
            System.loadLibrary(LIB_NAME);
    } catch (SecurityException se) {
                  // Handle SecurityException
      } catch (UnsatisfiedLinkError sle) {
                  // Handle UnsatisfiedLinkError
      } catch (NullPointerException npe) {
                  // Handle NullPointerException
      }
}
```

```
```

Aggregate Potential Exceptions to Calling Code

```
public static void loadLib() throws UnsatisfiedLinkError, NullPointerException,
SecurityException {
      System.loadLibrary(LIB_NAME);
}
```

Improper Resource Access Authorization

Risk

What might happen

Unauthorized actions may allow attackers to write malicious content or code to files, databases and other I\Os or read sensitive I\O contents. Impact of this issue varies, depending on implementation, but may allow:

- Remote code execution, in case an attacker is able to inject malicious data into a writable I\O, which would then be interpreted or compiled as code
- Overwriting or leaking of configuration files
- Compromising confidentiality or integrity of stored data

Cause

How does it happen

A logic flow in code triggers I/O and is not authorized. If an attacker can trigger it, it may leave it vulnerable to attack.

General Recommendations

How to avoid it

When logic flows are affected by user input or behavior, always ensure the user is authorized to trigger them.

Source Code Examples

Java

Writing to File Without Any Authorization Checks

```
Part filePart = request.getPart("file");
if (filePart != null) {
      InputStream filecontent = null;
      filecontent = filePart.getInputStream();
      Path path = Paths.get(filename);
      byte[] contentByteArray = new byte[filecontent.available()];
      filecontent.read(contentByteArray);
      Files.write(path, contentByteArray);
}
```

Using a Basic Authorization Check Based on Session Variables

```
HttpSession session = request.getSession();
String role = (String)session.getAttribute("role");
if (role.equals(ADMIN)) {
      Part filePart = request.getPart("file");
      if (filePart != null) {
            InputStream filecontent = null;
```

```
            filecontent = filePart.getInputStream();
            Path path = Paths.get(filename);
            byte[] contentByteArray = new byte[filecontent.available()];
            filecontent.read(contentByteArray);
            Files.write(path, contentByteArray);
        }
    }
```

Improper Resource Shutdown or Release

Risk

What might happen

Unreleased resources can cause a drain of those available for system use, eventually causing general reliability and availability problems, such as performance degradation, process bloat, and system instability. If a resource leak can be intentionally exploited by an attacker, it may be possible to cause a widespread DoS (Denial of Service) attack. This might even expose sensitive information between unprivileged users, if the resource continues to retain data or user id between subsequent allocations.

Cause

How does it happen

The application code allocates resource objects, but does not ensure these are always closed and released in a timely manner. This can include database connections, file handles, network sockets, or any other resource that needs to be released. In some cases, these might be released - but only if everything works as planned; if there is any runtime exception during the normal course of system operations, resources start to leak.

Note that even in managed-memory languages such as Java, these resources must be explicitly released. Many types of resource are not released even when the Garbage Collector runs; and even if the the object would eventually release the resource, we have no control over when the Garbage Collector does run.

General Recommendations

How to avoid it

- Always close and release all resources.
- Ensure resources are released (along with any other necessary cleanup) in a `finally { }` block. Do not close resources in a `catch { }` block, since this is not ensured to be called.
- Explicitly call .close() on any instance of a class that implements the `Closable` or `AutoClosable` interfaces.
- Alternatively, an even better solution is to use the try-with-resources idiom, in order to automatically close any defined `AutoClosable` instances.

Source Code Examples

Java

Unreleased Database Connection

```
private MyObject getDataFromDb(int id)  {
      MyObject data = null;
      Connection con = null;
      try {
            Connection con = DriverManager.getConnection(CONN_STRING);
            data = queryDb(con, id);
      }
      catch ( SQLException e ) {
            handleError(e);
```

```
        }
}
```

Explicit Release of Database Connection

```
private MyObject getDataFromDb(int id)  {
      MyObject data = null;
      Connection con = null;
      try {
            Connection con = DriverManager.getConnection(CONN_STRING);
            data = queryDb(con, id);
      }
      catch ( SQLException e ) {
            handleError(e);
      }
      finally {
            if ((con != null) && (!con.isClosed())) {
            con.close();
         }
     }
}
```

Automatic Implicit Release Using Try-With-Resources

```
private MyObject getDataFromDb(int id)  {
      MyObject data = null;
      Connection con = null;
      try (Connection con = DriverManager.getConnection(CONN_STRING)) {
            data = queryDb(con, id);
      }
      catch ( SQLException e ) {
            handleError(e);
      }
}
```

Information Exposure Through an Error Message

Risk

What might happen

Exposed details about the application's environment, users, or associated data (for example, stack trace) could enable an attacker to find another flaw and help the attacker to mount an attack. This may also leak sensitive data, e.g. passwords or database fields.

Cause

How does it happen

The application handles exceptions in an insecure manner, including raw details directly in the error message. This could occur in various ways: by not handling the exception; printing it directly to the output or file; explicitly returning the exception object; or by configuration. These exception details may include sensitive information that could leak to the users due to the occurrence of the runtime error.

General Recommendations

How to avoid it

- Do not expose exception data directly to the output or users, instead return an informative, generic error message. Log the exception details to a dedicated log mechanism.
- Any method that could throw an exception should be wrapped in an exception handling block that:
  - Explicitly handles expected exceptions.
  - Includes a default solution to explicitly handle unexpected exceptions.
- Configure a global handler to prevent unhandled errors from leaving the application.

Source Code Examples

Java

Handle Exception by Printing To Output

```
private void wrapCallToDB_Unsafe(HttpServletRequest request)
            throws ServletException, IOException {
      String paramValue = request.getParameter("Param");

      try {
            callDbProc(paramValue);
      } catch (SQLException ex) {
            ex.printStackTrace();
      }
}
```

Write Exception Details to Log, Send Generic Error Message

```
private void wrapCallToDB_SafePrintToLog(HttpServletRequest request)
            throws ServletException, IOException {
      String paramValue = request.getParameter("Param");

      try {
            callDbProc(paramValue);
      } catch (SQLException ex) {
            writeExceptionToLog(ex);
            System.err.println("Database Error, see log for details");
      }
}
```

Potential Clickjacking on Legacy Browsers

Risk

What might happen

Clickjacking attacks allow an attacker to "hijack" a user's mouse clicks on a webpage, by invisibly framing the application, and superimposing it in front of a bogus site. When the user is convinced to click on the bogus website, e.g. on a link or a button, the user's mouse is actually clicking on the target webpage, despite being invisible.

This could allow the attacker to craft an overlay that, when clicked, would lead the user to perform undesirable actions in the vulnerable application, e.g. enabling the user's webcam, deleting all the user's records, changing the user's settings, or causing clickfraud.

Cause

How does it happen

The root cause of vulnerability to a clickjacking attack, is that the application's web pages can be loaded into a frame of another website. The application does not implement a proper frame-busting script, that would prevent the page from being loaded into another frame. Note that there are many types of simplistic redirection scripts that still leave the application vulnerable to clickjacking techniques, and should not be used.

When dealing with modern browsers, applications mitigate this vulnerability by issuing appropriate Content-Security-Policy or X-Frame-Options headers to indicate to the browser to disallow framing. However, many legacy browsers do not support this feature, and require a more manual approach by implementing a mitigation in Javascript. To ensure legacy support, a framebusting script is required.

General Recommendations

How to avoid it

Generic Guidance:

- Define and implement a a Content Security Policy (CSP) on the server side, including a frame-ancestors directive. Enforce the CSP on all relevant webpages.
- If certain webpages are required to be loaded into a frame, define a specific, whitelisted target URL.
- Alternatively, return a "X-Frame-Options" header on all HTTP responses. If it is necessary to allow a particular webpage to be loaded into a frame, define a specific, whitelisted target URL.
- For legacy support, implement framebusting code using Javascript and CSS to ensure that, if a page is framed, it is never displayed, and attempt to navigate into the frame to prevent attack. Even if navigation fails, the page is not displayed and is therefore not interactive, mitigating potential clickjacking attacks.

Specific Recommendations:

- Implement a proper framebuster script on the client, that is not vulnerable to frame-buster-busting attacks.
    - Code should first disable the UI, such that even if frame-busting is successfully evaded, the UI cannot be clicked. This can be done by setting the CSS value of the "display" attribute to "none" on either the "body" or "html" tags. This is done because, if a frame attempts to redirect and become the parent, the malicious parent can still prevent redirection via various techniques.

- Code should then determine whether no framing occurs by comparing self === top; if the result is true, can the UI be enabled. If it is false, attempt to navigate away from the framing page by setting the top.location attribute to self.location.

Source Code Examples

JavaScript
Clickjackable Webpage

```
<html>
    <body>
     <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Bustable Framebuster

```
<html>
    <head>
     <script>
            if ( window.self.location != window.top.location ) {
                window.top.location = window.self.location;
            }
        </script>
    </head>

    <body>
     <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Proper Framebusterbusterbusting

```
<html>
    <head>
     <style> html {display : none; } </style>
        <script>
            if ( self === top ) {
                document.documentElement.style.display = 'block';
            }
            else {
                top.location = self.location;
            }
        </script>
    </head>
```

```
    <body>
     <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Incorrect Permission Assignment For Critical Resources

Risk

What might happen

Files with implicit or dangerous permissions may allow attackers to retrieve sensitive data from the contents of these files, tamper their contents or potentially execute them.

Cause

How does it happen

A file or directory is created with dangerous permissions, either by setting these permissions explicitly or relying on unsafe default permissions.

General Recommendations

How to avoid it

- Always create files with permissions being set explicitly
- Never set dangerous permissions on files
    - Always consider the principle of least privilege when determining who may read, write or execute a file, if these permissions are to be granted at all

---

Source Code Examples

Information Exposure Through Query String

Risk

What might happen

Sending sensitive information in a GET parameter as part of the URL's query string will result in this information potentially becoming cached by the browser, proxies, web-caches or be written into access logs. An attacker with access to any of the above will be able to retrieve this sensitive information.

Cause

How does it happen

A password is being sent in a GET request as a query string parameter, either via concatenation of the password value to a URL, or by sending the password as a parameter in a GET request. Sending parameters in a GET request is caused either by explicitly setting the method to GET, or implicitly by using a mechanism that defaults to a GET request without changing the method to another method (such as POST).

General Recommendations

How to avoid it

- Never send sensitive information in the URL. This includes:
  - Credentials
  - Session or access tokens
  - Personal information

Source Code Examples

Java

Receiving an access token through GET method.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    PrintWriter out = response.getWriter();
    String temp = request.getParameter("secret_token");
    if (temp==null) {
            out.print("Unauthorized");
    }else{
        out.print("<html><body><h1 align='center'>" + new Date().toString() +
"</h1></body></html>");
    }
}
```

Receiving an access token through POST method.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
```

```
    PrintWriter out = response.getWriter();
    String temp = request.getParameter("secret_token");

    if (temp==null) {
        out.print("Unauthorized");
    }else{
        out.print("<html><body><h1 align='center'>" +
        new Date().toString() + "</h1></body></html>");
    }
}
```

Information Leak Through Shell Error Message

Risk

What might happen

Sensitive data written out to console may leak to logs, errors messages and more.

Cause

How does it happen

Console output, which is sometimes used for debugging or ad-hoc logging, is often mistaken for a safe output stream. However, it may also redirect its content unexpected output streams - for example, console output may write to the actual console, but simultaenously also log these outputs in a file. Additionally, the contents and history of a console can be read by other processes. As such, a console should be considered an exposed form of output, and should not have any sensitive data written to it.

General Recommendations

How to avoid it

- Refrain from writing any sensitive data to the console, as the contents of a console may leak elsewhere.

Source Code Examples

Java

Print SessionId To Console Output

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
{
    HttpSession session = request.getSession();
      String sessionId = session.getId();
      System.out.println("SessionId: " + sessionId);
}
```

Missing X Frame Options

Risk

What might happen

Allowing setting of web-pages inside of a frame in an untrusted web-page will leave these web-pages vulnerable to Clickjacking, otherwise known as a redress attack. This may allow an attacker to redress a vulnerable web-page by setting it inside a frame within a malicious web-page. By crafting a convincing malicious web-page, the attacker can then use the overlayed redress to convince the user to click a certain area of the screen, unknowingly clicking inside the frame containing the vulnerable web-page, and thus performing actions within the user's context on the attacker's behalf.

Cause

How does it happen

Failure to utilize the "X-FRAME-OPTIONS" header will likely allow attackers to perform Clickjacking attacks. Properly utilizing the "X-FRAME-OPTIONS" header would indicate to the browser to disallow embedding the web-page within a frame, mitigating this risk, if the browser supports this header. All modern browsers support this header by default.

General Recommendations

How to avoid it

Utilize the "X-FRAME-OPTIONS" header flags according to business requirements to restrict browsers that support this header from allowing embedding web-pages in a frame:

· "X-Frame-Options: DENY" will indicate to the browser to disallow embedding any web-page inside a frame, including the current web-site.

· "X-Frame-Options: SAMEORIGIN" will indicate to the browser to disallow embedding any web-page inside a frame, excluding the current web-site.

· "X-Frame-Options: ALLOW-FROM https://example.com/" will indicate to the browser to disallow embedding any web-page inside a frame, excluding the web-site listed after the ALLOW-FROM parameter.

Source Code Examples

Java

Setting the "DENY" Flag on a Response

```
response.addHeader("X-Frame-Options", "DENY");
```

Open Redirect

Risk

What might happen

An attacker could use social engineering to get a victim to click a link to the application, so that the user will be immediately redirected to another site of the attacker's choice. An attacker can then craft a destination website to fool the victim; for example - they may craft a phishing website with an identical looking UI as the previous website's login page, and with a similar looking URL, convincing the user to submit their access credentials in the attacker's website. Another example would be a phishing website with an identical UI as that of a popular payment service, convincing the user to submit their payment information.

Cause

How does it happen

The application redirects the user’s browser to a URL provided by a tainted input, without first ensuring that URL leads to a trusted destination, and without warning users that they are being redirected outside of the current site. An attacker could use social engineering to get a victim to click a link to the application with a parameter defining another site to which the application will redirect the user’s browser. Since the user may not be aware of the redirection, they may be under the misconception that the website they are currently browsing can be trusted.

General Recommendations

How to avoid it

1. Ideally, do not allow arbitrary URLs for redirection. Instead, create a mapping from user-provided parameter values to legitimate URLs.
2. If it is necessary to allow arbitrary URLs:
    - For URLs inside the application site, first filter and encode the user-provided parameter, and then either:
        - Create a white-list of allowed URLs inside the application
        - Use variables as a relative URL as an absolute one, by prefixing it with the application site domain - this will ensure all redirection will occur inside the domain
    - For URLs outside the application (if necessary), either:
        - White-list redirection to allowed external domains by first filtering URLs with trusted prefixes. Prefixes must be tested up to the third slash [/] - `scheme://my.trusted.domain.com/`, to prevent evasion. For example, if the third slash [/] is not validated and scheme://my.trusted.domain.com is trusted, the URL scheme://my.trusted.domain.com.evildomain.com would be valid under this filter, but the domain actually being browsed is evildomain.com, not domain.com.
        - For fully dynamic open redirection, use an intermediate disclaimer page to provide users with a clear warning that they are leaving the site.

Source Code Examples

Java

Java Servlet Vulnerable to Open Redirection

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      String redirectUrl = request.getParameter("redirectUrl");
      if (redirectUrl != null) {
            response.sendRedirect(redirectUrl);
      } else {
            response.sendRedirect("/");
    }
}
```

Whitelisting an Allowed External Domain, Preventing Open Redirection

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      String redirectUrl = request.getParameter("redirectUrl");
      if (redirectUrl != null && redirectUrl.startsWith("https://www.trusteddomain.com/")) {
            response.sendRedirect(redirectUrl);
      } else {
            response.sendRedirect("/");
    }
}
```

Missing Content Security Policy

Risk

What might happen

The Content-Security-Policy header enforces that the source of content, such as the origin of a script, embedded (child) frame, embedding (parent) frame or image, are trusted and allowed by the current web-page; if, within the web-page, a content's source does not adhere to a strict Content Security Policy, it is promptly rejected by the browser. Failure to define a policy may leave the application's users exposed to Cross-Site Scripting (XSS) attacks, Clickjacking attacks, content forgery and more.

Cause

How does it happen

The Content-Security-Policy header is used by modern browsers as an indicator for trusted sources of content, including media, images, scripts, frames and more. If these policies are not explicitly defined, default browser behavior would allow untrusted content.

The application creates web responses, but does not properly set a Content-Security-Policy header.

General Recommendations

How to avoid it

Explicitly set the Content-Security-Policy headers for all applicable policy types (frame, script, form, script, media, img etc.) according to business requirements and deployment layout of external file hosting services. Specifically, do not use a wildcard, '*', to specify these policies, as this would allow content from any external resource.

The Content-Security-Policy can be explicitly defined within web-application code, as a header managed by web-server configurations, or within `<meta>` tags in the HTML `<head>` section.

Source Code Examples

Java

HTTP Response With CSP Header Set

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
  // handle request
  response.setHeader("Content-Security-Policy", "default-src 'self'"); // default-src is the
most restric mode of CSP and covers all applicable policy types
}
```

HTTP Response with CSP Header in Spring

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Override
protected void configure(HttpSecurity http) throws Exception {
 http
   .headers()
   .contentSecurityPolicy("default-src 'self'"); // default-src is the most restric mode of
CSP and covers all applicable policy types
     }
}
```

Plaintext Storage in a Cookie

Risk

What might happen

The application stores data in plain-text in a cookie. If this data is sensitive, setting it in plain-text as the contents of a cookie will increase its exposure significantly, by ensuring it is transmitted back-and-forth as part of every request made. If the cookie is then used by the application in additional contexts, it may allow further attack depending on implementation.

Cause

How does it happen

The application sets a cookie with user-provided data.

General Recommendations

How to avoid it

- Do not rely on user-input to craft cookie content
- Never store sensitive data in cookies

Source Code Examples

Java

Setting A User-Provided Password in a Cookie

```
byte[] passwordBytes = request.getParameter("password").getBytes();
response.addCookie(new Cookie("secret",new String(passwordBytes)));
```

Setting A User-Provided Password in a Cookie as a Base64 Encoded Value

```
byte[] passwordBytes = request.getParameter("password").getBytes();
byte[] encodedPassword = Base64.getEncoder().encode(passwordBytes); // Encoding is as
ineffective as plain-text in keeping data secret
response.addCookie(new Cookie("secret",new String(encodedPassword)));
```

Relative Path Traversal

Risk

What might happen

An attacker could define arbitrary file path for the application to use, potentially leading to:

- Stealing sensitive files, such as configuration or system files
- Overwriting files such as program binaries, configuration files, or system files
- Deleting critical files, causing denial of service (DoS).

Cause

How does it happen

The application uses user input in the file path for accessing files on the application server's local disk.

General Recommendations

How to avoid it

1. Ideally, avoid depending on dynamic data for file selection.
2. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
3. Accept dynamic data only for the filename, not for the path and folders.
4. Ensure that file path is fully canonicalized.
5. Explicitly limit the application to use a designated folder that is separate from the applications binary folder.
6. Restrict the privileges of the application's OS user to necessary files and folders. The application should not be able to write to the application binary folder, and should not read anything outside of the application folder and data folder.

Source Code Examples

Java

Absolute Path Traversal in "filename" Parameter

```
private String getFileContents(HttpServletRequest request) throws ServletException,
FileNotFoundException, IOException {
      String filename = request.getParameter("filename");
      Path path = Paths.get(filename);
      byte[] fileContentBytes = Files.readAllBytes(path);
      String fileContents = new String(fileContentBytes, FILE_CONTENT_ENCODING_STRING);
      return fileContents;
```

```
}
```

Relative Path Traversal in "filename" Parameter

```
private String getFileContents(HttpServletRequest request) throws ServletException,
FileNotFoundException, IOException {
      String filename = request.getParameter("filename");
      Path path = Paths.get(SERVED_FILES_DIR + filename);
      byte[] fileContentBytes = Files.readAllBytes(path);
      String fileContents = new String(fileContentBytes, FILE_CONTENT_ENCODING_STRING);
      return fileContents;
}
```

Path Traversal Mitigate via Sanitization of Path Variable

```
private static String sanitizePathTraversal(String filename) {
      Path p = Paths.get(filename);
      return p.getFileName().toString();
}

private String getFileContents_fixed(HttpServletRequest request) throws ServletException,
FileNotFoundException, IOException {
      String filename = sanitizePathTraversal(request.getParameter("filename")); // Ensures
access only to files in a given folder, no traversal
      Path path = Paths.get(SERVED_FILES_DIR + filename);
      byte[] fileContentBytes = Files.readAllBytes(path);
      String fileContents = new String(fileContentBytes, FILE_CONTENT_ENCODING_STRING);
      return fileContents;
}
```

Race Condition

Risk

What might happen

A race condition can cause erratic or unexpected application behavior. Additionally, if a race condition can be influenced directly by users, an attacker may choose to replay a race condition until they obtain a desired result, allowing them to induce certain application behavior that is not part of its intentional design.

Cause

How does it happen

Most modern environments, and web in particular, require a high level of concurrent actions, invoking many instances of an object, or a singleton object multiple times, using an internal scheduler that decides which thread should be operating at the moment. This scheduler is often out of a developer's control, and may halt and resume concurrent threads. These threads often operate over shared resources, which are part of a functionality that is being invoked simultaneously across multiple logic flows; for example - static member variables are shared across all instances of an object type, while in singleton objects (such as servlets) any member variable is shared across all calls to the singleton.

When working in an environment with concurrency, failure to ensure all operations on a shared resource are done atomically and in a synchronized manner may result in threads or processes overriding each other's actions or outputs, often contradicting intended logic. For example, consider the "Time of Check, Time of Use" problem (TOCTOU) - if two threads attempt to increment the same counter, thread 1 reads the value n, and is then halted by the scheduler, thread 2 will read the value n, and then both threads will attempt to update the value to n+1, where logic would dictate that it would be n+2. Had this counter been properly synchronized, thread 2 would have to wait for thread 1 to complete its entire atomic operation of checking and updating its value.

General Recommendations

How to avoid it

- Consider allocating independent resources for concurrent logical processes, unless required.
- Where resource sharing is required, utilize a locking mechanism when handling shared resources to ensure that actions are performed atomically by the current process, so that these logical processes are not overridden or disrupted by any other concurrent process.
- When dealing with static member variables - always consider that these are shared across all instances of an object
- When dealing with singleton objects (such as servlets) being invoked by multiple concurrent logical processes - always synchronize either the entire singleton or its member variables; otherwise any thread processing the singleton's static methods may cause TOCTOU issues

---

Source Code Examples

Reliance on Cookies in a Decision

Risk

What might happen

Cookie values are used in making a decision without validating their contents - this may allow users to tamper with these cookies thus affecting the outcome of this decision, despite not being authorized to do so. This may lead to authorization and authentication bypasses, whose exploitation may lead to account theft, elevation of privilege and more, depending on implementation.

Cause

How does it happen

Cookie values are usually set by the server, via the Set-Cookie header, or via client-side code - thereby resulting in these values not being transparent to end-users. However, tampering with the values of these cookies is trivial, and therefore their contents should be treated as potentially malicious or tainted inputs.

General Recommendations

How to avoid it

For any session-related decisions, never rely on user inputs alone - always validate user authority and authenticity from trusted credentials, such as a session variable, before relying on cookie values in a decision.

Source Code Examples

Java

User Creation Relying on Admin Role With Role Derived From Cookies

```
Cookie[] cookiejar = request.getCookies();
if (cookies != null) {
      for (Cookie cookie : cookiejar) {
            if (cookie.getName().equals("role") && cookie.getValue().equals("admin")) {
                  addUser(username, email, password, role);
            }
         else
         {
            unauthorizedError();
         }
      }
}
else
{
      unauthorizedError();
}
```

User Creation Relying on Admin Role With Role Derived From Session Variables

```
String role = request.getSession(true).getAttribute("role"); // role attribute must never be
set with input from users
if (role != null) {
      if (role.equals("admin")) {
            addUser(username, email, password, role);
      }
      else
     {
      unauthorizedError();
     }
}
else
{
      unauthorizedError();
}
```

Reversible One Way Hash

Risk

What might happen

Applications depend on cryptography in order to protect secrets and other sensitive or personally identifiable data. When there is a flaw in a cryptographic implementation, it may compromise the integrity, authenticity or confidentiality of the application's data.

Hashing functions are used for several purposes in cryptography, but mainly they are linked with key derivation functions and signatures. When hashing functions are used in an insecure manner, the integrity and confidentiality of the password hash or signature can be at stake.

When applications rely on weak or broken hash functions to perform cryptographic operations to provide integrity or authentication features, attackers can leverage the known attacks against them to break signatures or passwords hashes. This could result in loss of confidentiality, integrity and authenticity of data.

Cause

How does it happen

There are issues in cryptography-related functionality of the application.

The application is using a weak hashing primitive. MD4, MD5, SHA-1, etc. have been found to have collisions and other weaknesses that make them unsuitable for current deployments.

General Recommendations

How to avoid it

Update the hashing function to a safer alternative, like:

- BLAKE2B (modern, fast in software, safe against length-extension attacks)
- SHA-2 Family hashes (SHA-256,SHA-384,SHA-512)

Source Code Examples

Java

Weak Hash

```
import java.security.MessageDigest;

public class WeakHash{
    public static void main(String[] args) throws Exception{
        byte[] testData = {0x00};
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] hash = md.digest(testData);
    }
}
```

Stored Absolute Path Traversal

Risk

What might happen

An attacker could define arbitrary file path for the application to use, potentially leading to:

- Stealing sensitive files, such as configuration or system files
- Overwriting files such as program binaries, configuration files, or system files
- Deleting critical files, causing denial of service (DoS).

Cause

How does it happen

The application relies on data from storage to determine the file path for accessing files on the application server's local disk. If this data can be tainted by user input, an attacker may store their own value to perform path traversal via the vulnerable method.

General Recommendations

How to avoid it

1. Ideally, avoid depending on dynamic data for file selection.
2. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
3. Accept dynamic data only for the filename, not for the path and folders.
4. Ensure that file path is fully canonicalized.
5. Explicitly limit the application to use a designated folder that is separate from the applications binary folder.
6. Restrict the privileges of the application's OS user to necessary files and folders. The application should not be able to write to the application binary folder, and should not read anything outside of the application folder and data folder.

Source Code Examples

Stored Relative Path Traversal

Risk

What might happen

An attacker could define arbitrary file path for the application to use, potentially leading to:

- Stealing sensitive files, such as configuration or system files
- Overwriting files such as program binaries, configuration files, or system files
- Deleting critical files, causing denial of service (DoS).

Cause

How does it happen

The application relies on data from storage to determine the file path for accessing files on the application server's local disk. If this data can be tainted by user input, an attacker may store their own value to perform path traversal via the vulnerable method.

General Recommendations

How to avoid it

1. Ideally, avoid depending on dynamic data for file selection.
2. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
3. Accept dynamic data only for the filename, not for the path and folders.
4. Ensure that file path is fully canonicalized.
5. Explicitly limit the application to use a designated folder that is separate from the applications binary folder.
6. Restrict the privileges of the application's OS user to necessary files and folders. The application should not be able to write to the application binary folder, and should not read anything outside of the application folder and data folder.

Source Code Examples

Java

Stored Path Traversal from DB

```
String query = "SELECT username FROM users WHERE id = ?";
PreparedStatement pstmt = con.prepareStatement(query);
pstmt.setInt(userID, 1);

ResultSet resultSet = pstmt.executeQuery();
```

```
resultSet.next();
path = resultSet.getString(1);
con.close();

File file = new File(path);
Scanner scanner = new Scanner(file);
System.out.println(scanner.nextLine());
scanner.close();
```

Stored Path Traversal from DB Mitigated by Utilizing Paths.get() and getName()

```
String query = "SELECT username FROM users WHERE id = ?";
PreparedStatement pstmt = con.prepareStatement(query);
pstmt.setInt(userID, 1);

ResultSet resultSet = pstmt.executeQuery();
resultSet.next();
path = resultSet.getString(1);
con.close();

File file = new File(path);
String fileName = file.getName();
Path safePath = Paths.get("userFiles", fileName);
file = new File(safePath.toString());

Scanner scanner = new Scanner(file);
System.out.println(scanner.nextLine());
scanner.close();
```

Sensitive Cookie in HTTPS Session Without Secure Attribute

Risk

What might happen

Cookies that contain the user's session identifier, and other sensitive application cookies, should be sent to the server over a secure network communication (HTTPS) in order to prevent attackers from sniffing the traffic and stealing those cookies. Unless the web application explicitly prevents this by using the "secure" cookie flag, these cookies will also be sent over insecure traffic, which can lead to session hijacking and impersonation.

Cause

How does it happen

The web application framework by default does not set the "secure" flag for the application's sessionID cookie, and other sensitive application cookies. Likewise, the application does not explicitly use the "secure" cookie flag, allowing them to be sent in plaintext over an insecure session.

General Recommendations

How to avoid it

- Always set the "secure" flag for any sensitive server-side cookies.
- If the application explicitly handles cookies directly in application code, set the "secure" flag for sensitive cookies set by the application. **Secure-Code Approach**
- Configure the application to always use "secure" cookies, in the site-wide configuration file.
- Enable the Secure flag or use the relevant Set-Secure API in the code.

Source Code Examples

Java

Web.xml Configuration File with Secure Cookies

```
<session-config>
 <cookie-config>
  <secure>true</secure>
 </cookie-config>
</session-config>
```

Explicit Cookie Handling in Code

```
Cookie newCookie = new Cookie("name","value");
newCookie.setSecure(true);
```

Suspected XSS

Risk

What might happen

A successful XSS exploit would allow an attacker to rewrite web pages and insert malicious scripts which would alter the intended output. This could include HTML fragments, CSS styling rules, arbitrary JavaScript, or references to third party code. An attacker could use this to steal users' passwords, collect personal data such as credit card details, provide false information, or run malware. From the victim’s point of view, this is performed by the genuine website, and the victim would blame the site for incurred damage.

An attacker could use legitimate access to the application to submit modified data to the application’s data-store. This would then be used to construct the returned web page, triggering the attack.

Cause

How does it happen

The application creates web pages that include untrusted data, whether from user input, the application’s database, or from other external sources. The untrusted data is embedded directly in the page's HTML, causing the browser to display it as part of the web page. If the input includes HTML fragments or JavaScript, these are displayed too, and the user cannot tell that this is not the intended page. The vulnerability is the result of directly embedding arbitrary data without first encoding it in a format that would prevent the browser from treating it like HTML or code instead of plain text.

In order to exploit this vulnerability, an attacker would load the malicious payload into the data-store, typically via regular forms on other web pages. Afterwards, the application reads this data from the data-store, and embeds it within the web page as displayed for another user.

General Recommendations

How to avoid it

- Fully encode all dynamic data, regardless of source, before embedding it in output.
- Encoding should be context-sensitive. For example:
  - HTML encoding for HTML content
  - HTML Attribute encoding for data output to attribute values
  - JavaScript encoding for server-generated JavaScript
- It is recommended to use the platform-provided encoding functionality, or known security libraries for encoding output.
- Implement a Content Security Policy (CSP) with explicit whitelists for the application's resources only.
- As an extra layer of protection, validate all untrusted data, regardless of source (note this is not a replacement for encoding). Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
  - Data type
  - Size
  - Range
  - Format
  - Expected values
- In the `Content-Type` HTTP response header, explicitly define character encoding (charset) for the entire page.
- Set the `HTTPOnly` flag on the session cookie for "Defense in Depth", to prevent any successful XSS exploits from stealing the cookie.

Source Code Examples

Unrestricted File Upload

Risk

What might happen

Allowing users to save files of unrestricted size may allow attackers to fill file storage with junk, or conduct long writing operations which would strain systems conducting the saving operation. Exhausting this storage space or constraining it to the point where it is unavailable will result in denial of service.

Cause

How does it happen

Application code does not validate file size before saving files uploaded by users to storage, potentially allowing upload of files of any size.

General Recommendations

How to avoid it

Constrain intended file size in code to prevent attackers from uploading files of arbitrary sizes by performing size checks. Do not rely on client-side size checks or any size parameters provided by users; evaluate the size of the file on the server, instead.

Source Code Examples

Java

Use of Broken or Risky Cryptographic Algorithm

Risk

What might happen

Using a weak or broken algorithm ruins the protection granted by using cryptographic mechanisms in the first place, harming the confidentiality or integrity of sensitive user data. This could allow an attacker to steal secret information, alter sensitive data, or forge the source of modified messages.

Cause

How does it happen

The application code specifies the name of the selected cryptographic algorithm, either via a String argument, a factory method, or a specific implementation class. These algorithms have fatal cryptographic weaknesses, that make it trivial to break in a reasonable timeframe. Strong algorithms should withstand attacks far beyond the realm of possible.

General Recommendations

How to avoid it

- Only use strong, approved cryptographic algorithms, including AES, RSA, ECC, and SHA-256 respectively, amongst others.
- Do not use weak algorithms that are considered completely broken, such as DES, RC4, and MD5, amongst others.
- Avoid, where possible, using legacy algorithms that are not considered "future-proof" with sufficient safety margins, even though they are considered "safe enough" for today. This includes algorithms that are weaker than they should be, and have stronger replacements, even if they are not yet fatally broken - such as SHA-1, 3DES,
- Consider using a relevant official set of classifications, such as NIST or ENISA. If possible, use only FIPS 140-2 certified algorithm implementations.

Source Code Examples

Java

Legacy Encryption with DES

```
private static byte[] encrypt(String plainText) {
      byte[] encData;

    SecretKey key = new SecretKeySpec(SECRET_KEY, "DES");

      Cipher cipher = Cipher.getInstance("DES/ECB/NoPadding");
      cipher.init(Cipher.ENCRYPT_MODE, key);

      encData = cipher.doFinal(plainText.getBytes());
      byte[] iv = cipher.getIV();

      return encData;
```

```
}
```

Go

Modern Encryption with AES

```
private static byte[] encrypt(String plainText) {
      byte[] encData;

      try {
         SecretKey key = new SecretKeySpec(SECRET_KEY_FROM_CONFIG, "AES");

         Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                  // Consider a stronger mode, such as GCM / CCM / or other AEAD mode, if
required
         cipher.init(Cipher.ENCRYPT_MODE, key);

             encData = cipher.doFinal(plainText.getBytes("UTF-8"));
         byte[] iv = cipher.getIV();

     } catch (GeneralSecurityException e) {
         handleError(e);
     }
      return encData;
}
```

Use of Non Cryptographic Random

Risk

What might happen

Random values are often used as a mechanism to prevent malicious users from knowing or predicting a given value, such as a password, encryption key, or session identifier. Depending on what this random value is used for, an attacker would be able to predict the next numbers generated, or previously generated values, based on sources often used to derive certain randomness; however, while they may seem random, large statistical samples would demonstrate that they are insufficiently random, producing a much smaller space of possible "random" values than a truly random sample would. This could enable an attacker to derive or guess this value, and thus hijack another user's session, impersonate another user, or crack an encryption key (depending on what the pseudo-random value was used for).

Cause

How does it happen

The application uses a weak method of generating pseudo-random values, such that other numbers could be determined from a relatively small sample size. Since the pseudo-random number generator used is designed for statistically uniform distribution of values, it is approximately deterministic. Thus, after collecting a few generated values, it would be possible for an attacker to calculate past or future values.
Specifically, if this pseudo-random value is used in any security context, such as one-time passwords, keys, secret identifiers or salts - an attacker would likely be able to predict the next value generated and steal it, or guess a previously generated value and spoof its original intent.

General Recommendations

How to avoid it

- Always use a cryptographically secure pseudo-random number generator, instead of basic random methods, particularly when dealing with a security context
- Use the cryptorandom generator that is built-in to your language or platform, and ensure it is securely seeded. Do not seed the generator with a weak, non-random seed. (In most cases, the default is securely random).
- Ensure you use a long enough random value, thus making brute-force attacks unfeasible.

Source Code Examples

Java

Use of Cryptographically Insecure PRNG

```
Random random = new Random();
Long sessNum = random.nextLong();
String sessionId = sessNum.toString();
```

Use of Cryptographically Secure PRNG

```
SecureRandom random = new SecureRandom();
byte sessBytes[] = new byte[32];
random.nextBytes(sessBytes);
String sessionId = new String(sessBytes);
```

Unsynchronized Access To Shared Data

Risk

What might happen

A race condition can cause erratic or unexpected application behavior. Additionally, if a race condition can be influenced directly by users, an attacker may choose to replay a race condition until they obtain a desired result, allowing them to induce certain application behavior that is not part of its intentional design.

Cause

How does it happen

Most modern environments, and web in particular, require a high level of concurrent actions, invoking many instances of an object, or a singleton object multiple times, using an internal scheduler that decides which thread should be operating at the moment. This scheduler is often out of a developer's control, and may halt and resume concurrent threads. These threads often operate over shared resources, which are part of a functionality that is being invoked simultaneously across multiple logic flows; for example - static member variables are shared across all instances of an object type, while in singleton objects (such as servlets) any member variable is shared across all calls to the singleton.

When working in an environment with concurrency, failure to ensure all operations on a shared resource are done atomically and in a synchronized manner may result in threads or processes overriding each other's actions or outputs, often contradicting intended logic. For example, consider the "Time of Check, Time of Use" problem (TOCTOU) - if two threads attempt to increment the same counter, thread 1 reads the value n, and is then halted by the scheduler, thread 2 will read the value n, and then both threads will attempt to update the value to n+1, where logic would dictate that it would be n+2. Had this counter been properly synchronized, thread 2 would have to wait for thread 1 to complete its entire atomic operation of checking and updating its value.

General Recommendations

How to avoid it

- Consider allocating independent resources for concurrent logical processes, unless required.
- Where resource sharing is required, utilize a locking mechanism when handling shared resources to ensure that actions are performed atomically by the current process, so that these logical processes are not overridden or disrupted by any other concurrent process.
- When dealing with static member variables - always consider that these are shared across all instances of an object
- When dealing with singleton objects (such as servlets) being invoked by multiple concurrent logical processes - always synchronize either the entire singleton or its member variables; otherwise any thread processing the singleton's static methods may cause TOCTOU issues

Source Code Examples

Stored HTTP Response Splitting

Risk

What might happen

If the header setting code is of a vulnerable version, an attacker could:

- Arbitrarily change the application server's response header to a victim's HTTP request by manipulating headers
- Arbitrarily change the application server's response body by injecting two consecutive line breaks, which may result in Cross-Site Scripting (XSS) attacks
- Cause cache poisoning, potentially controlling any site's HTTP responses going through the same proxy as this application.

Cause

How does it happen

Since user input is being used in an HTTP response header, an attacker could include NewLine characters to make the header look like multiple headers with engineered content, potentially making the response look like multiple responses (for example, by engineering duplicate content-length headers). This can cause an organizational proxy server to provide the second, engineered response to a victim's subsequent request; or, if the proxy server also performs response caching, the attacker can send an immediate subsequent request to another site, causing the proxy server to cache the engineered response as a response from this second site and to later serve the response to other users.

Many modern web frameworks mitigate this issue, by offering sanitization for new line characters in strings inserted into headers by default. However, since many older versions of web frameworks fail to automatically mitigate this issue, manual sanitization of input may be required.

General Recommendations

How to avoid it

1. Validate all input, regardless of source (including cookies). Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
2. Additionally, remove or URL-encode all special (non-alphanumeric) user input before including it in the response header.
3. Make sure to use an up-to-date framework.

Source Code Examples

Java

Stored Data Affects a Response Header

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      // Vulnerable in legacy versions of Java
    String user = request.getParameter("user");
      // Perform a SQL Query
      ResultSet rs = getUserLocation(user);
      String userlocation = rs.getString("userLocation");
    response.setContentType("text/html");
    Cookie cookie = new Cookie("location", userLocation);
    cookie.setMaxAge(3600);
    response.addCookie(cookie);
}
```

Stored Data URLEncoded Prior Setting a Response Header

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      // Vulnerable in legacy versions of Java
    String user = request.getParameter("user");
      // Perform a SQL Query
      ResultSet rs = getUserLocation(user);
      String userlocation = rs.getString("userLocation");
    response.setContentType("text/html");
    Cookie cookie = new Cookie("location", URLEncoder.encode(userLocation, "UTF-8"));
    cookie.setMaxAge(3600);
    response.addCookie(cookie);
}
```

Stored Boundary Violation

Risk

What might happen

Code that reads from Session variables may trust them as server-side variables, but they may have been tainted by user inputs. This can lead to tampering with parameters used to authenticate or authorize users. Further, tainted Session variables offer an additional attack surface against the application - if untrusted data taints a Session variable, and that Session variable is then used elsewhere without sanitization as if it were trusted, it could lead to further attacks such as Cross-Site Scripting, SQL Injection and more.

Cause

How does it happen

Server-side Session variables, or objects, are values assigned to a specific session, which is associated with a specific user. Often, they hold data relevant to that user's session, such as specific identifiers, user-type, authorization, authentication information and more. As such, the paradigm often associated to the Session object is that its contents can be trusted, as users cannot generally set these values themselves.

The application places user input, which is untrusted data, in the server-side Session object, which is considered a trusted location. This could lead developers to treat untrusted data as trusted.

General Recommendations

How to avoid it

1. Validate and sanitize all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
2. Don’t mix untrusted user input with trusted data.

Source Code Examples

Use of Function with Inconsistent Implementations

Weakness ID: 474 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The code uses a function that has inconsistent implementations across operating systems and versions, which might cause security-relevant portability problems.

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

C: *(Often)*

PHP: *(Often)*

All

Potential Mitigations

Do not accept inconsistent behavior from the API specifications when the deviant behavior increase the risk level.

Other Notes

The behavior of functions in this category varies by operating system, and at times, even by operating system version. Implementation differences can include:

- Slight differences in the way parameters are interpreted leading to inconsistent results.
- Some implementations of the function carry significant security risks.
- The function might not be defined on all platforms.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	398	Indicator of Poor Code Quality	**Development Concepts (primary)699 Seven Pernicious Kingdoms (primary)700 Research Concepts (primary)1000**
ParentOf	Weakness Variant	589	Call to Non-ubiquitous API	**Research Concepts (primary)1000**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Inconsistent Implementations

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	7 Pernicious Kingdoms		Externally Mined
Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Eric Dalci	Cigital	External
	updated Potential Mitigations, Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Relationships, Other Notes, Taxonomy Mappings		
Previous Entry Names			
Change Date	**Previous Entry Name**		
2008-04-11	Inconsistent Implementations		

BACK TO TOP

Scanned Languages

Language	Hash Number	Change Date
Java	1417405572051301	10/23/2020
JavaScript	0128891942401620	10/23/2020
Common	3450320970348195	10/23/2020