00027_gpf_dsifs Scan Report

Project Name	00027_gpf_dsifs
Scan Start	Tuesday, November 29, 2022 9:06:37 AM
Preset	Checkmarx Default
Scan Time	00h:01m:27s
Lines Of Code Scanned	3122
Files Scanned	35
Report Creation Time	Tuesday, November 29, 2022 9:12:42 AM
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112
Team	PIC-CGI-GPF
Checkmarx Version	8.9.0.210 HF30
Scan Type	Full
Source Origin	LocalPath
Density	10/1000 (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

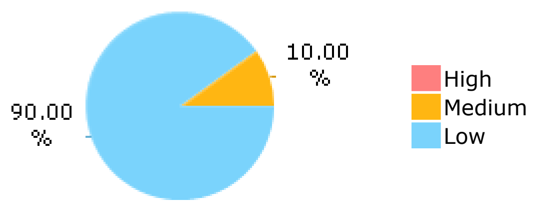
Results Limit

Results limit per query was set to 50

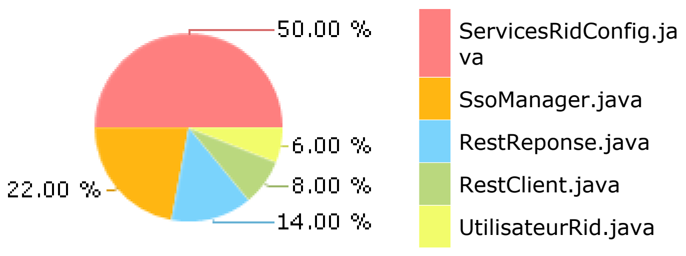
Selected Queries

Selected queries are listed in Result Summary

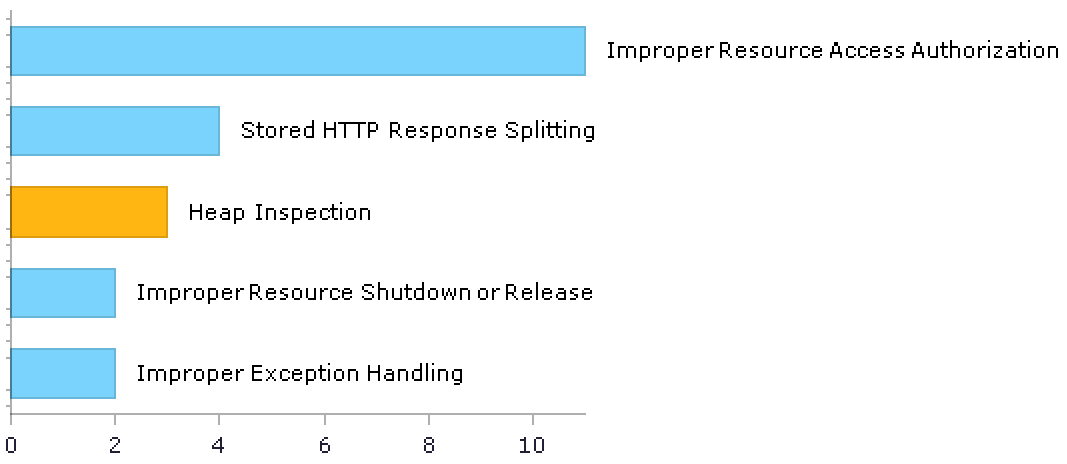
Result Summary



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	5	2
A2-Broken Authentication*	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A3-Sensitive Data Exposure*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	5	5
A4-XML External Entities (XXE)	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	0	0
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	1	1
A6-Security Misconfiguration *	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	2	2
A7-Cross-Site Scripting (XSS)	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
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	1	1
A2-Broken Authentication and Session Management*	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	0	0
A3-Cross-Site Scripting (XSS)	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	0	0
A4-Insecure Direct Object References	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	1	1
A5-Security Misconfiguration *	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	2	2
A6-Sensitive Data Exposure*	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	4	4
A7-Missing Function Level Access Control*	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	0	0
A8-Cross-Site Request Forgery (CSRF)	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
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	2	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 - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection	1	1
PCI DSS (3.2) - 6.5.2 - Buffer overflows	0	0
PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage*****	0	0
PCI DSS (3.2) - 6.5.4 - Insecure communications*****	0	0
PCI DSS (3.2) - 6.5.5 - Improper error handling*****	4	4
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.2) - 6.5.8 - Improper access control*****	1	1
PCI DSS (3.2) - 6.5.9 - Cross-site request forgery	0	0
PCI DSS (3.2) - 6.5.10 - Broken authentication and session management	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 - 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.	0	0
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.	2	2
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.	12	12
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.	3	3
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.	7	3

* 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)*	11	11
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)	0	0
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)	0	0
SC-23 Session Authenticity (P1)*	0	0
SC-28 Protection of Information at Rest (P1)*	1	1
SC-4 Information in Shared Resources (P1)	3	3
SC-5 Denial of Service Protection (P1)*	4	3
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	7	3
SI-11 Error Handling (P2)*	2	2
SI-15 Information Output Filtering (P0)	0	0
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.	0	0
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

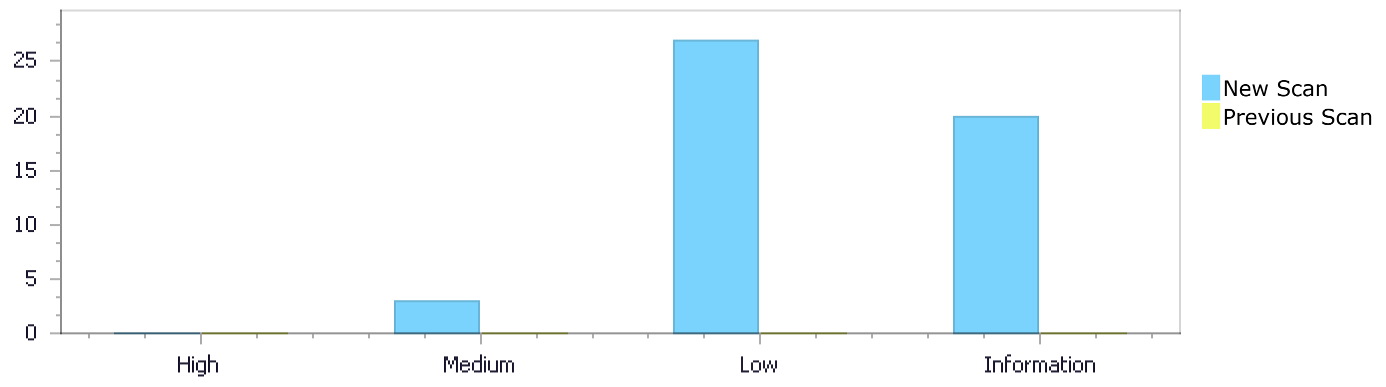
Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	0	3	27	20	50
Recurrent Issues	0	0	0	0	0
Total	0	3	27	20	50

Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	0	0	0	0	0
To Verify	0	3	27	20	50
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	0	3	27	20	50

Result Summary

Vulnerability Type	Occurrences	Severity
Heap Inspection	3	Medium
Improper Resource Access Authorization	11	Low
Stored HTTP Response Splitting	4	Low
Improper Exception Handling	2	Low
Improper Resource Shutdown or Release	2	Low

Information Exposure Through an Error Message	2	Low
Stored Open Redirect	2	Low
Heuristic 2nd Order SQL Injection	1	Low
Password In Comment	1	Low
Serializable Class Containing Sensitive Data	1	Low
Stored Absolute Path Traversal	1	Low
Insufficient Logging of Exceptions	20	Information

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java	1
ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java	1
ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/webservices/WSClientBase.java	1

Scan Results Details

Heap Inspection

Query Path:
Java\Cx\Java Medium Threat\Heap Inspection Version:3

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	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=9
Status	New

Method password; at line 26 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.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	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java
Line	26	26
Object	password	password

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java
Method	private String password;

```
....
26.    private String password;
```

Heap Inspection\Path 2:

Severity	Medium
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=10
Status	New

Method null; at line 33 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java defines passwordApplication, which is designated to contain user passwords. However, while plaintext passwords are later assigned to passwordApplication, this variable is never cleared from memory.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java
Line	33	33
Object	passwordApplication	passwordApplication

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java
Method	private String passwordApplication = null;

```
....
33.    private String passwordApplication = null;
```

Heap Inspection\Path 3:

Severity	Medium
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=11
Status	New

Method passwordApplication; at line 14 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/webservices/WSClientBase.java defines passwordApplication, which is designated to contain user passwords. However, while plaintext passwords are later assigned to passwordApplication, this variable is never cleared from memory.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/webservices/WSClientBase.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/webservices/WSClientBase.java
Line	14	14
Object	passwordApplication	passwordApplication

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/webservices/WSClientBase.java
Method	private String passwordApplication;

```
....
14.    private String passwordApplication;
```

Improper Resource Access Authorization

Query Path:
Java\Cx\Java Low Visibility\Improper Resource Access Authorization Version:4

Categories

FISMA 2014: Identification And Authentication
NIST SP 800-53: AC-3 Access Enforcement (P1)

Description

Improper Resource Access Authorization\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=12
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	63	63
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getCodeDomaine() {

```
....
63.          return
this.properties.getProperty("servicesrid.code_domaine");
```

Improper Resource Access Authorization\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=13
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	72	72
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getUrlAuthentification() {

```
....
72.          return
this.properties.getProperty("servicesrid.web_authentification_url");
```

Improper Resource Access Authorization\Path 3:

Severity	Low

Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=14
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	81	81
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getWebServicesUrl() {

```
....
81.         return
this.properties.getProperty("servicesrid.web_services_url");
```

Improper Resource Access Authorization\Path 4:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=15
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	90	90
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getWebServicesUtilisateur() {

```
....
90.         return
this.properties.getProperty("servicesrid.web_services_utilisateur");
```

Improper Resource Access Authorization\Path 5:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=16

Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	99	99
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getWebServicesMotDePasse() {

```
....
99.          return
this.properties.getProperty("servicesrid.web_services_motdepasse");
```

Improper Resource Access Authorization\Path 6:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=17
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	108	108
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getLogOnUrl() {

```
....
108.            return
this.properties.getProperty("servicesrid.log_on_url");
```

Improper Resource Access Authorization\Path 7:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=18
Status	New

	Source	Destination

File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	112	112
Object	getProperty	getProperty

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method public ModeAuthentification getModeAuthentificationRequis() {

```
....
112.                return
ModeAuthentification.fromString(this.properties.getProperty("servicesrid
.mode_requis"));
```

Improper Resource Access Authorization\Path 8:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=19
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	121	121
Object	getProperty	getProperty

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method public String getJetonUrl() {

```
....
121.                return
this.properties.getProperty("servicesrid.jeton_url");
```

Improper Resource Access Authorization\Path 9:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=20
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/Servic	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/Servic

	esRidConfig.java	esRidConfig.java
Line	130	130
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getLogOffUrl() {

```
....
130.                return
this.properties.getProperty("servicesrid.log_off_url");
```

Improper Resource Access Authorization\Path 10:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=21
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	139	139
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getLogAbsoluteFilePath() {

```
....
139.                return
this.properties.getProperty("servicesrid.log_absolute_file_path");
```

Improper Resource Access Authorization\Path 11:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=22
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	148	148

Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getLogLevel() {

```
....
148.                    return
this.properties.getProperty("servicesrid.log_level");
```

Stored HTTP Response Splitting

Query Path:
Java\Cx\Java Stored\Stored HTTP Response Splitting Version:0

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	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=2
Status	New

Method getUrlAuthentification at line 71 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java gets user input from the getProperty 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 deconnecterSso at line 75 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	72	92
Object	getProperty	setHeader

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getUrlAuthentification() {

```
....
72.             return
this.properties.getProperty("servicesrid.web_authentification_url");
```

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method	public void deconnecterSso(final String redirectUri) {

```
....
92.                 response.setHeader("Location", sb.toString());
```

Stored HTTP Response Splitting\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=3
Status	New

Method getUrlAuthentification at line 71 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java gets user input from the getProperty 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 redirigerPageLogOn at line 101 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	72	145
Object	getProperty	setHeader

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getUrlAuthentification() {

```
....
72.             return
this.properties.getProperty("servicesrid.web_authentification_url");
```

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method	public void redirigerPageLogOn(final String returnUri) {

```
....
145.                    response.setHeader("Location", urlLogin);
```

Stored HTTP Response Splitting\Path 3:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=4
Status	New

Method getUrlAuthentification at line 71 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java gets user input from the getProperty 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 deconnecterSso at line 75 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	72	90
Object	getProperty	sendRedirect

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getUrlAuthentification() {

```
....
72.         return
this.properties.getProperty("servicesrid.web_authentification_url");
```

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method	public void deconnecterSso(final String redirectUri) {

```
....
90.                 response.sendRedirect(sb.toString());
```

Stored HTTP Response Splitting\Path 4:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=5
Status	New

Method getUrlAuthentification at line 71 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java gets user input from the getProperty 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 redirigerPageLogOn at line 101 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	72	143
Object	getProperty	sendRedirect

Code Snippet	
File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getUrlAuthentification() {

```
....
72.          return
this.properties.getProperty("servicesrid.web_authentification_url");
```

▼

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method	public void redirigerPageLogOn(final String returnUri) {

```
....
143.                         response.sendRedirect(urlLogin);
```

Stored Open Redirect

Query Path:
Java\Cx\Java Stored\Stored Open Redirect Version:0

Categories

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

Stored Open Redirect\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=6
Status	New

A possible open redirect has been found at line 75 in ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java file. This might lead to an untrusted site which mainly used for phishing.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	72	90
Object	getProperty	sendRedirect

Code Snippet	
File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getUrlAuthentification() {

```
....
72.          return
this.properties.getProperty("servicesrid.web_authentification_url");
```

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method	public void deconnecterSso(final String redirectUri) {

```
....
90.                    response.sendRedirect(sb.toString());
```

Stored Open Redirect\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=7
Status	New

A possible open redirect has been found at line 101 in ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java file. This might lead to an untrusted site which mainly used for phishing.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	72	143
Object	getProperty	sendRedirect

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getUrlAuthentification() {

```
....
72.          return
this.properties.getProperty("servicesrid.web_authentification_url");
```

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method	public void redirigerPageLogOn(final String returnUri) {

```
....
143.                         response.sendRedirect(urlLogin);
```

Improper Resource Shutdown or Release

Query Path:
Java\Cx\Java Low Visibility\Improper Resource Shutdown or Release Version:4

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	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=45
Status	New

The application's ServicesRidConfig method in ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java defines and initializes the FileInputStream 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	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	26	33
Object	FileInputStream	input

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	private ServicesRidConfig() throws IOException {

```
....
26.                 input = new FileInputStream(conf);
....
33.           properties.load(input);
```

Improper Resource Shutdown or Release\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=46
Status	New

The application's ServicesRidConfig method in ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java defines and initializes the getResourceAsStream 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	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java

Line	28	33
Object	getResourceAsStream	input

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	private ServicesRidConfig() throws IOException {

```
....
28.                 input = 
getClass().getResourceAsStream("/servicesrid.properties");
....
33.            properties.load(input);
```

Improper Exception Handling

Query Path:
Java\Cx\Java Low Visibility\Improper Exception Handling Version:0

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	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=47
Status	New

Method ServicesRidConfig at line 22 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java 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	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	23	23
Object	getProperty	getProperty

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	private ServicesRidConfig() throws IOException {

```
....
23.            String conf = System.getProperty("service.rid.config");
```

Improper Exception Handling\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=48
Status	New

Method ServicesRidConfig at line 22 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java 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	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	33	33
Object	load	load

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	private ServicesRidConfig() throws IOException {

```
....
33.         properties.load(input);
```

Information Exposure Through an Error Message

Query Path:
Java\Cx\Java Low Visibility\Information Exposure Through an Error Message Version:2

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	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=49
Status	New

Method getInstance, at line 41 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to printStackTrace, in method getInstance of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java, line 41.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/	ServicesRidJava/src/src/main/java/com/

	sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	50	51
Object	e	printStackTrace

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method public final static ServicesRidConfig getInstance() {

```
....
50.           } catch (IOException e) {
51.                  e.printStackTrace();
```

Information Exposure Through an Error Message\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=50
Status	New

Method handle, at line 121 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java, handles an Exception or runtime Error e. During the exception handling code, the application exposes the exception details to printStackTrace, in method handle of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java, line 121.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java
Line	127	128
Object	e	printStackTrace

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java
Method public ClientResponse handle(ClientRequest cr) {

```
....
127.                  } catch (IOException e) {
128.                         e.printStackTrace();
```

Serializable Class Containing Sensitive Data

Query Path:
Java\Cx\Java Low Visibility\Serializable Class Containing Sensitive Data Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description
Serializable Class Containing Sensitive Data\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=1
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java
Line	25	16
Object	password	UtilisateurRid

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java
Method @JsonProperty("Password")

```
....
25.    @JsonProperty("Password")
```

File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java
Method @XmlRootElement

```
....
16.  @XmlRootElement
```

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	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=8
Status	New

Method ServicesRidConfig at line 22 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java gets dynamic data from the getProperty element. This element’s value then flows through the code and is eventually used in a file path for local disk access in ServicesRidConfig at line 22 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java. This may cause a Path Traversal vulnerability.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	23	26
Object	getProperty	FileInputStream

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	private ServicesRidConfig() throws IOException {

```
....
23.         String conf = System.getProperty("service.rid.config");
....
26.               input = new FileInputStream(conf);
```

Password In Comment

Query Path:
Java\Cx\Java Low Visibility\Password In Comment Version:4

Categories

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

Password In Comment\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=23
Status	New

The application contains passwords embedded in source code comments, such as password at line 40 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java, which can easily be viewed by users.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java
Line	40	40
Object	password	password

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/domaine/UtilisateurRid.java

Method	* @param password : Mot de passe

```
....
40.     * @param password : Mot de passe
```

Heuristic 2nd Order SQL Injection

Query Path:
Java\Cx\Java Heuristic\Heuristic 2nd 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

Heuristic 2nd Order SQL Injection\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=44
Status	New

The application's chargerParProfil method executes an SQL query with req, at line 145 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/webservices/WSPopulationsClient.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.

This apparent database access is seemingly encapsulated in an external component or API.

Thus, it seems that the attacker might be able to write arbitrary data to the database, which is then retrieved by the application with getProperty in getCodeDomaine method, at line 62 of ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java. This data then flows through the code to the external API, where it might be used directly in the SQL query, apparently without sanitization.This query might then be submitted to a database server for execution.

This may enable a Second-Order SQL Injection attack, based on heuristic analysis.

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/webservices/WSPopulationsClient.java
Line	63	164
Object	getProperty	req

Code Snippet	
File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public String getCodeDomaine() {

```
....
63.         return
this.properties.getProperty("servicesrid.code_domaine");
```

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/webservices/WSPopulationsClient.java
Method	public PersonneRid chargerParProfil(String codeDomaine, String codeProfil, String identifiantRid) throws WSException {

```
....
164.              RestReponse<PersonneRid> rep =
client.executeRequest(req);
```

Insufficient Logging of Exceptions

Query Path:
Java\Cx\Java Best Coding Practice\Insufficient Logging of Exceptions Version:1
Description

Insufficient Logging of Exceptions\Path 1:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=24
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Line	50	50
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/configuration/ServicesRidConfig.java
Method	public final static ServicesRidConfig getInstance() {

```
....
50.         } catch (IOException e) {
```

Insufficient Logging of Exceptions\Path 2:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=25
Status	New

	Source	Destination

File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java
Line	107	107
Object	catch	catch

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java
Method public RestReponse<REP_T> executeRequest(RestRequete<REQ_T> request) {

```
....
107.                } catch (Throwable e) {
```

Insufficient Logging of Exceptions\Path 3:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=26
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java
Line	127	127
Object	catch	catch

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestClient.java
Method public ClientResponse handle(ClientRequest cr) {

```
....
127.                } catch (IOException e) {
```

Insufficient Logging of Exceptions\Path 4:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=27
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Line	67	67

Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Method	public T getObject() {

```
....
67.          } catch (JsonParseException e1) {
```

Insufficient Logging of Exceptions\Path 5:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=28
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Line	69	69
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Method	public T getObject() {

```
....
69.          } catch (JsonMappingException e1) {
```

Insufficient Logging of Exceptions\Path 6:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=29
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Line	71	71
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestRepons e.java
Method	public T getObject() {

```
....
71.          } catch (IllegalArgumentException e1) {
```

Insufficient Logging of Exceptions\Path 7:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid =1010457&projectid=11112&pathid=30
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/ sncf/dsit/servicesrid/helpers/RestRepons e.java	ServicesRidJava/src/src/main/java/com/ sncf/dsit/servicesrid/helpers/RestRepons e.java
Line	73	73
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestRepons e.java
Method	public T getObject() {

```
....
73.          } catch (IOException e1) {
```

Insufficient Logging of Exceptions\Path 8:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid =1010457&projectid=11112&pathid=31
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/ sncf/dsit/servicesrid/helpers/RestRepons e.java	ServicesRidJava/src/src/main/java/com/ sncf/dsit/servicesrid/helpers/RestRepons e.java
Line	112	112
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestRepons e.java
Method	public List<T> getList() {

```
....
112.                } catch (JsonParseException e) {
```

Insufficient Logging of Exceptions\Path 9:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=32
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Line	114	114
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Method	public List<T> getList() {

```
....
114.                } catch (IllegalArgumentException e) {
```

Insufficient Logging of Exceptions\Path 10:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=33
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Line	116	116
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/helpers/RestReponse.java
Method	public List<T> getList() {

```
....
116.                } catch (IOException e) {
```

Insufficient Logging of Exceptions\Path 11:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=34
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	85	85
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method	public void deconnecterSso(final String redirectUri) {

```
....
85.          } catch(UnsupportedEncodingException uee) {
```

Insufficient Logging of Exceptions\Path 12:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=35
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	91	91
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method	public void deconnecterSso(final String redirectUri) {

```
....
91.          } catch (IOException ioe) {
```

Insufficient Logging of Exceptions\Path 13:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=36
Status	New

	Source	Destination

File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	135	135
Object	catch	catch

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method public void redirigerPageLogOn(final String returnUri) {

```
....
135.                catch (UnsupportedEncodingException uee) {
```

Insufficient Logging of Exceptions\Path 14:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=37
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	138	138
Object	catch	catch

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Method public void redirigerPageLogOn(final String returnUri) {

```
....
138.                catch (UriBuilderException ube) {
```

Insufficient Logging of Exceptions\Path 15:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=38
Status	New

	Source	Destination
File	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java	ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java
Line	144	144
Object	catch	catch

Code Snippet
File Name ServicesRidJava/src/src/main/java/com/sncf/dsit/servicesrid/SsoManager.java

Method	public void redirigerPageLogOn(final String returnUri) {

```
....
144.                } catch (IOException ioe) {
```

Insufficient Logging of Exceptions\Path 16:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=39
Status	New

	Source	Destination
File	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSDomainesClientTest.java	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSDomainesClientTest.java
Line	43	43
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSDomainesClientTest.java
Method	public void testChargerTous() {

```
....
43.            } catch (WSException e) {
```

Insufficient Logging of Exceptions\Path 17:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=40
Status	New

	Source	Destination
File	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSJetonsClientTest.java	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSJetonsClientTest.java
Line	48	48
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSJetonsClientTest.java
Method	public void testCreationVerificationJeton() {

```
....
48.            catch(WSException e) {
```

Insufficient Logging of Exceptions\Path 18:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=41
Status	New

	Source	Destination
File	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSJetonsClientTest.java	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSJetonsClientTest.java
Line	63	63
Object	catch	catch

Code Snippet
File Name ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSJetonsClientTest.java
Method public void testVerifierJetonInexistant() {

```
....
63.          catch(WSException e) {
```

Insufficient Logging of Exceptions\Path 19:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid=1010457&projectid=11112&pathid=42
Status	New

	Source	Destination
File	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSPersonnesClientTest.java	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSPersonnesClientTest.java
Line	45	45
Object	catch	catch

Code Snippet
File Name ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSPersonnesClientTest.java
Method public void testRecuperationPersonne() {

```
....
45.          catch(WSException e) {
```

Insufficient Logging of Exceptions\Path 20:

Severity	Information
Result State	To Verify
Online Results	https://checksecucode.apps.eul.sncf.fr/CxWebClient/ViewerMain.aspx?scanid

	=1010457&projectid=11112&pathid=43
Status	New

	Source	Destination
File	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSPopulationsClientTest.java	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSPopulationsClientTest.java
Line	46	46
Object	catch	catch

Code Snippet

File Name	ServicesRidJava/src/src/test/java/com/sncf/dsit/servicesrid/webservices/WSPopulationsClientTest.java
Method	public void testChargerParProfil() {

```
....
46.          catch(WSException e) {
```

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.
   }
}
```

Serializable Class Containing Sensitive Data

Weakness ID: 499 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

The code contains a class with sensitive data, but the class does not explicitly deny serialization. The data can be accessed by serializing the class through another class.

Extended Description

Serializable classes are effectively open classes since data cannot be hidden in them. Classes that do not explicitly deny serialization can be serialized by any other class, which can then in turn use the data stored inside it.

Time of Introduction

- Implementation

Applicable Platforms

Languages

Java

Common Consequences

Scope	Effect
Confidentiality	an attacker can write out the class to a byte stream, then extract the important data from it.

Likelihood of Exploit

High

Demonstrative Examples

Example 1

(Bad Code)

Example Language: **Java**

```
class Teacher {
private String name;
private String clas;
public Teacher(String name,String clas) {

//...
//Check the database for the name and address
this.SetName() = name;
this.Setclas() = clas;
}
}
```

Potential Mitigations

Phase: Implementation

In Java, explicitly define final writeObject() to prevent serialization. This is the recommended solution. Define the writeObject() function to throw an exception explicitly denying serialization.

Phase: Implementation

Make sure to prevent serialization of your objects.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	485	Insufficient Encapsulation	**Development Concepts (primary)699 Research Concepts (primary)1000**

CanPrecede	Weakness Class	200	Information Exposure	Development Concepts699 Research Concepts1000

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
CLASP			Information leak through serialization

Content History

Submissions

Submission Date	Submitter	Organization	Source
	CLASP		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Common Consequences, Description, Relationships, Taxonomy Mappings		
2009-07-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		

Previous Entry Names

Change Date	Previous Entry Name
2008-04-11	Information Leak through Serialization

BACK TO TOP

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.

Source Code Examples

CSharp

User input is being used in an HTTP response header, enabling an attacker to add a newline character and multiple headers

```
public class HTTPResponseSplitting
{
        public void foo(HttpRequest Request, HttpResponse Response)
     {
                string author = Request.Params["author"];
                Response.AppendHeader("Author", author);
        }
}
```

The user input is both excluded if it contains a newline character and the input is also URL encoded

```
public class HTTPResponseSplittingFixed
{
        public void foo(HttpRequest Request, HttpResponse Response)
     {
                string author = Request.Params["author"];

                if (!author.Contains('\n'))
                {
                        author = HttpUtility.UrlEncode(author);
                        Response.AppendHeader("Author", author);
                }
        }
}
```

Stored 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, arbitrary site. Users may think that they are still in the original application site. The second site may be offensive, contain malware, or, most commonly, be used for phishing.

Cause

How does it happen

The application redirects the user's browser to a URL provided in a user request, without warning users that they are being redirected outside the 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, and the user may not be aware of the redirection.

General Recommendations

How to avoid it

1. Ideally, do not allow arbitrary URLs for redirection. Instead, create a server-side 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 use it as a relative URL by prefixing it with the application site domain.
    - For URLs outside the application (if necessary), use an intermediate disclaimer page to provide users with a clear warning that they are leaving your site.

Source Code Examples

CSharp

Avoid redirecting to arbitrary URLs, instead map the parameter to a list of static URLs.

```
Response.Redirect(getUrlById(targetUrlId));
```

Java

Avoid redirecting to arbitrary URLs, instead map the parameter to a list of static URLs.

```
Response.Redirect(getUrlById(targetUrlId));
```

Apex

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

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

CSharp

Using unvalidated user input as the file name may enable the user to access arbitrary files on the server local disk

```
public class PathTraversal
{
        private void foo(TextBox textbox1)
    {
                string fileNum = textbox1.Text;
                string path = "c:\files\file" + fileNum;
                FileStream f = new FileStream(path, FileMode.Open);
                byte[] output = new byte[10];
                f.Read(output,0, 10);
        }
}
```

Potentially hazardous characters are removed from the user input before use

```
public class PathTraversalFixed
{
        private void foo(TextBox textbox1)
    {
                string fileNum = textbox1.Text.Replace("\", "").Replace("..", "");
        string path = "c:\files\file" + fileNum;
                FileStream f = new FileStream(path, FileMode.Open);
                byte[] output = new byte[10];
                f.Read(output,0, 10);
        }
}
```

Java

Using unvalidated user input as the file name may enable the user to access arbitrary files on the server local disk

```
public class Absolute_Path_Traversal {
    public static void main(String[] args) {
        Scanner userInputScanner = new Scanner(System.in);
        System.out.print("\nEnter file name: ");
        String name = userInputScanner.nextLine();
        String path = "c:\files\file" + name;
        try {
            BufferedReader reader = new BufferedReader(new FileReader(path));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Potentially hazardous characters are removed from the user input before use

```
public class Absolute_Path_Traversal_Fixed {
    public static void main(String[] args) {
        Scanner userInputScanner = new Scanner(System.in);
        System.out.print("\nEnter file name: ");
        String name = userInputScanner.nextLine();
        name = name.replace("/", "").replace("..", "");
        String path = "c:\files\file" + name;
        try {
            BufferedReader reader = new BufferedReader(new FileReader(path));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Improper Access Control (Authorization)

Weakness ID: 285 *(Weakness Class)* **Status:** Draft

Description

Description Summary

The software does not perform or incorrectly performs access control checks across all potential execution paths.

Extended Description

When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information leaks, denial of service, and arbitrary code execution.

Alternate Terms

AuthZ:	"AuthZ" is typically used as an abbreviation of "authorization" within the web application security community. It is also distinct from "AuthC," which is an abbreviation of "authentication." The use of "Auth" as an abbreviation is discouraged, since it could be used for either authentication or authorization.

Time of Introduction

- Architecture and Design
- Implementation
- Operation

Applicable Platforms

Languages

Language-independent

Technology Classes

Web-Server: *(Often)*

Database-Server: *(Often)*

Modes of Introduction

A developer may introduce authorization weaknesses because of a lack of understanding about the underlying technologies. For example, a developer may assume that attackers cannot modify certain inputs such as headers or cookies.

Authorization weaknesses may arise when a single-user application is ported to a multi-user environment.

Common Consequences

Scope	Effect
Confidentiality	An attacker could read sensitive data, either by reading the data directly from a data store that is not properly restricted, or by accessing insufficiently-protected, privileged functionality to read the data.
Integrity	An attacker could modify sensitive data, either by writing the data directly to a data store that is not properly restricted, or by accessing insufficiently-protected, privileged functionality to write the data.
Integrity	An attacker could gain privileges by modifying or reading critical data directly, or by accessing insufficiently-protected, privileged functionality.

Likelihood of Exploit

High

Detection Methods

Automated Static Analysis

Automated static analysis is useful for detecting commonly-used idioms for authorization. A tool may be able to analyze related configuration files, such as .htaccess in Apache web servers, or detect the usage of commonly-used authorization libraries.

Generally, automated static analysis tools have difficulty detecting custom authorization schemes. In addition, the software's design may include some functionality that is accessible to any user and does not require an authorization check; an automated technique that detects the absence of authorization may report false positives.

Effectiveness: Limited

Automated Dynamic Analysis

Automated dynamic analysis may find many or all possible interfaces that do not require authorization, but manual analysis is required to determine if the lack of authorization violates business logic

Manual Analysis

This weakness can be detected using tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session.

Specifically, manual static analysis is useful for evaluating the correctness of custom authorization mechanisms.

Effectiveness: Moderate

These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules. However, manual efforts might not achieve desired code coverage within limited time constraints.

Demonstrative Examples

Example 1

The following program could be part of a bulletin board system that allows users to send private messages to each other. This program intends to authenticate the user before deciding whether a private message should be displayed. Assume that LookupMessageObject() ensures that the $id argument is numeric, constructs a filename based on that id, and reads the message details from that file. Also assume that the program stores all private messages for all users in the same directory.

(Bad Code)

Example Language: **Perl**

```
sub DisplayPrivateMessage {
my($id) = @_;
my $Message = LookupMessageObject($id);
print "From: " . encodeHTML($Message->{from}) . "<br>\n";
print "Subject: " . encodeHTML($Message->{subject}) . "\n";
print "<hr>\n";
print "Body: " . encodeHTML($Message->{body}) . "\n";
}

my $q = new CGI;
# For purposes of this example, assume that CWE-309 and
# CWE-523 do not apply.
if (! AuthenticateUser($q->param('username'), $q->param('password'))) {
ExitError("invalid username or password");
}

my $id = $q->param('id');
DisplayPrivateMessage($id);
```

While the program properly exits if authentication fails, it does not ensure that the message is addressed to the user. As a result, an authenticated attacker could provide any arbitrary identifier and read private messages that were intended for other users.

One way to avoid this problem would be to ensure that the "to" field in the message object matches the username of the authenticated user.

Observed Examples

Reference	Description
CVE-2009-3168	Web application does not restrict access to admin scripts, allowing authenticated users to reset administrative passwords.

CVE-2009-2960	Web application does not restrict access to admin scripts, allowing authenticated users to modify passwords of other users.
CVE-2009-3597	Web application stores database file under the web root with insufficient access control (CWE-219), allowing direct request.
CVE-2009-2282	Terminal server does not check authorization for guest access.
CVE-2009-3230	Database server does not use appropriate privileges for certain sensitive operations.
CVE-2009-2213	Gateway uses default "Allow" configuration for its authorization settings.
CVE-2009-0034	Chain: product does not properly interpret a configuration option for a system group, allowing users to gain privileges.
CVE-2008-6123	Chain: SNMP product does not properly parse a configuration option for which hosts are allowed to connect, allowing unauthorized IP addresses to connect.
CVE-2008-5027	System monitoring software allows users to bypass authorization by creating custom forms.
CVE-2008-7109	Chain: reliance on client-side security (CWE-602) allows attackers to bypass authorization using a custom client.
CVE-2008-3424	Chain: product does not properly handle wildcards in an authorization policy list, allowing unintended access.
CVE-2009-3781	Content management system does not check access permissions for private files, allowing others to view those files.
CVE-2008-4577	ACL-based protection mechanism treats negative access rights as if they are positive, allowing bypass of intended restrictions.
CVE-2008-6548	Product does not check the ACL of a page accessed using an "include" directive, allowing attackers to read unauthorized files.
CVE-2007-2925	Default ACL list for a DNS server does not set certain ACLs, allowing unauthorized DNS queries.
CVE-2006-6679	Product relies on the X-Forwarded-For HTTP header for authorization, allowing unintended access by spoofing the header.
CVE-2005-3623	OS kernel does not check for a certain privilege before setting ACLs for files.
CVE-2005-2801	Chain: file-system code performs an incorrect comparison (CWE-697), preventing defauls ACLs from being properly applied.
CVE-2001-1155	Chain: product does not properly check the result of a reverse DNS lookup because of operator precedence (CWE-783), allowing bypass of DNS-based access restrictions.

Potential Mitigations

Phase: Architecture and Design

Divide your application into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully mapping roles with data and functionality. Use role-based access control (RBAC) to enforce the roles at the appropriate boundaries.

Note that this approach may not protect against horizontal authorization, i.e., it will not protect a user from attacking others with the same role.

Phase: Architecture and Design

Ensure that you perform access control checks related to your business logic. These checks may be different than the access control checks that you apply to more generic resources such as files, connections, processes, memory, and database records. For example, a database may restrict access for medical records to a specific database user, but each record might only be intended to be accessible to the patient and the patient's doctor.

Phase: Architecture and Design

Strategy: Libraries or Frameworks

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, consider using authorization frameworks such as the JAAS Authorization Framework and the OWASP ESAPI Access Control feature.

Phase: Architecture and Design

For web applications, make sure that the access control mechanism is enforced correctly at the server side on every page. Users should not be able to access any unauthorized functionality or information by simply requesting direct access to that page.

One way to do this is to ensure that all pages containing sensitive information are not cached, and that all such pages restrict access to requests that are accompanied by an active and authenticated session token associated with a user who has the required permissions to access that page.

Phases: System Configuration; Installation

Use the access control capabilities of your operating system and server environment and define your access control lists accordingly. Use a "default deny" policy when defining these ACLs.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	254	Security Features	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Class	284	Access Control (Authorization) Issues	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ChildOf	Category	721	OWASP Top Ten 2007 Category A10 - Failure to Restrict URL Access	**Weaknesses in OWASP Top Ten (2007) (primary)629**
ChildOf	Category	723	OWASP Top Ten 2004 Category A2 - Broken Access Control	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Category	753	2009 Top 25 - Porous Defenses	**Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)750**
ChildOf	Category	803	2010 Top 25 - Porous Defenses	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
ParentOf	Weakness Variant	219	Sensitive Data Under Web Root	**Research Concepts (primary)1000**
ParentOf	Weakness Base	551	Incorrect Behavior Order: Authorization Before Parsing and Canonicalization	**Development Concepts (primary)699** Research Concepts1000
ParentOf	Weakness Class	638	Failure to Use Complete Mediation	Research Concepts1000
ParentOf	Weakness Base	804	Guessable CAPTCHA	**Development Concepts (primary)699** **Research Concepts (primary)1000**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Missing Access Control
OWASP Top Ten 2007	A10	CWE More Specific	Failure to Restrict URL Access
OWASP Top Ten 2004	A2	CWE More Specific	Broken Access Control

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
1	Accessing Functionality Not Properly Constrained by ACLs	
13	Subverting Environment Variable Values	

17	Accessing, Modifying or Executing Executable Files
87	Forceful Browsing
39	Manipulating Opaque Client-based Data Tokens
45	Buffer Overflow via Symbolic Links
51	Poison Web Service Registry
59	Session Credential Falsification through Prediction
60	Reusing Session IDs (aka Session Replay)
77	Manipulating User-Controlled Variables
76	Manipulating Input to File System Calls
104	Cross Zone Scripting

References

NIST. "Role Based Access Control and Role Based Security". <http://csrc.nist.gov/groups/SNS/rbac/>.

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 4, "Authorization" Page 114; Chapter 6, "Determining Appropriate Access Control" Page 171. 2nd Edition. Microsoft. 2002.

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 Time of Introduction		
2008-08-15		Veracode	External
	Suggested OWASP Top Ten 2004 mapping		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Other Notes, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Common Consequences, Description, Likelihood of Exploit, Name, Other Notes, Potential Mitigations, References, Relationships		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Potential Mitigations		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Description, Related Attack Patterns		
2009-07-27	CWE Content Team	MITRE	Internal
	updated Relationships		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Type		
2009-12-28	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Demonstrative Examples, Detection Factors, Modes of Introduction, Observed Examples, Relationships		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Alternate Terms, Detection Factors, Potential Mitigations, References, Relationships		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Potential Mitigations		

Previous Entry Names

Change Date	Previous Entry Name
2009-01-12	Missing or Inconsistent Access Control

BACK TO TOP

Password In Comment

Risk

What might happen

It is often possible to retrieve and view the application source code. For web applications, it is even simpler to "`View Source`" in the user's browser. Thus, a malicious user can steal these passwords, and use them to impersonate whoever they belong to. It is not known if these are valid, current passwords, nor if they are user passwords or for backend systems, like a database.

Cause

How does it happen

A well-developed application will have it's source code well commented. Often, programmers will leave deployment information in comments, or retain debugging data that was used during development. These comments often contain secret data, such as passwords. These password comments are stored in the source code in perpetuity, and are not protected.

General Recommendations

How to avoid it

Do not store secrets, such as passwords, in source code comments.

Source Code Examples

JavaScript

Old code retained in comment with password

```
function login() {
    // send credentials to server instead of reading from database
    // constring = "Initial Catalog=mytest;User Id=sa;Pwd=mypass;";
    var creds = "username=" + txtUsername.text + "&password=" + txtPassword.text;

    var resp = sendToServer("/login", creds);
    return (resp == "success");
}
```

Passwords Cleaned from Comments

```
function login() {
    // send credentials to server instead of reading from database
    var creds = "username=" + txtUsername.text + "&password=" + txtPassword.text;

    var resp = sendToServer("/login", creds);
    return (resp == "success");
}
```

Heuristic 2nd 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.

Note that in this case, the injection appears to be in an external component, which might be implementing its own internal checks.

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 the apparent database access is encapsulated in an external component or API. Thus, the attacker is able to inject arbitrary data into the SQL query, by updating fields in the database. Later, the application reads this data from the database, and embeds it within the SQL query, as SQL commands. This query is then passed to the API or component, where it is presumably submitted to the database server.

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.

- Prefer standard data access libraries and platform APIs, instead of using opaque 3rd party drivers.

- Data validation can be performed effectively using a secure library, such as OWASP's Encoder or ESAPI libraries.
- Prefer using standard `PreparedStatement` for parameterizing the queries, or even better `CallableStatement`. Add dynamic data via the `.set*()` methods, instead of string concatenation.
- Consider using a standard ORM package, such as `Hibernate`, `myBatis`, or others, instead of a custom data wrapper.

Source Code Examples

Java
Create Pseudo-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 dataOperation = "SELECT [Contact] FROM [AppUsers] WHERE [UserName] = '" +
userNameFromDB + "' " ;

            CustomDbDriver db = new CustomDbDriver();
            contact = db.runQuery(dataOperation);
      } catch (SQLException ex) {
            handleExceptions(ex);
      }
      finally {
            closeQuietly(readNameStmt);
            closeQuietly(conn);
      }

      return contact;
}
```

Use Standard PreparedStatement API to Call Regular Stored Procedures

```
public String getActiveContact_SafeParameterizedQuery(HttpServletRequest request)
            throws ServletException, IOException {

      String userNameFromDB;
      String contact;

      String sqlStoredProc = "{call getUserDetails (?, ?)}";
```

```
        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;
   }
```

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);
      }
}
```

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).
- Inadvertent application crashes may occur.

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

CSharp

Always catch exceptions explicitly.

```
try
{
// Database access or other potentially dangerous function
}
catch (SqlException ex)
{
// Handle exception
}
catch (Exception ex)
{
// Default handler for unexpected exceptions
}
```

Java

Always catch exceptions explicitly.

```
try
{
// Database access or other potentially dangerous function
}
catch (SQLException ex)
{
// Handle exception
}
catch (Exception ex)
{
// Default handler for unexpected exceptions
}
```

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

Insufficient Logging

Weakness ID: 778 *(Weakness Base)* **Status:** Draft

Description

Description Summary

When a security-critical event occurs, the software either does not record the event or omits important details about the event when logging it.

Extended Description

When security-critical events are not logged properly, such as a failed login attempt, this can make malicious behavior more difficult to detect and may hinder forensic analysis after an attack succeeds.

Time of Introduction

- Operation

Applicable Platforms

Languages

Language-independent

Common Consequences

Scope	Effect
Accountability	If security critical information is not recorded, there will be no trail for forensic analysis and discovering the cause of problems or the source of attacks may become more difficult or impossible.

Likelihood of Exploit

Medium

Demonstrative Examples

Example 1

The example below shows a configuration for the service security audit feature in the Windows Communication Foundation (WCF).

(Bad Code)

Example Language: **XML**

```xml
<system.serviceModel>
<behaviors>
<serviceBehaviors>
<behavior name="NewBehavior">
<serviceSecurityAudit auditLogLocation="Default"
suppressAuditFailure="false"
serviceAuthorizationAuditLevel="None"
messageAuthenticationAuditLevel="None" />
...
</system.serviceModel>
```

The previous configuration file has effectively disabled the recording of security-critical events, which would force the administrator to look to other sources during debug or recovery efforts.

Logging failed authentication attempts can warn administrators of potential brute force attacks. Similarly, logging successful authentication events can provide a useful audit trail when a legitimate account is compromised. The following configuration shows appropriate settings, assuming that the site does not have excessive traffic, which could fill the logs if there are a large number of success or failure events (CWE-779).

(Good Code)

Example Language: **XML**

```
<system.serviceModel>
<behaviors>
<serviceBehaviors>
<behavior name="NewBehavior">
<serviceSecurityAudit auditLogLocation="Default"
suppressAuditFailure="false"
serviceAuthorizationAuditLevel="SuccessAndFailure"
messageAuthenticationAuditLevel="SuccessAndFailure" />
...
</system.serviceModel>
```

Observed Examples

Reference	Description
CVE-2008-4315	server does not log failed authentication attempts, making it easier for attackers to perform brute force password guessing without being detected
CVE-2008-1203	admin interface does not log failed authentication attempts, making it easier for attackers to perform brute force password guessing without being detected
CVE-2007-3730	default configuration for POP server does not log source IP or username for login attempts
CVE-2007-1225	proxy does not log requests without "http://" in the URL, allowing web surfers to access restricted web content without detection
CVE-2003-1566	web server does not log requests for a non-standard request type

Potential Mitigations

Phase: Architecture and Design

Use a centralized logging mechanism that supports multiple levels of detail. Ensure that all security-related successes and failures can be logged.

Phase: Operation

Be sure to set the level of logging appropriately in a production environment. Sufficient data should be logged to enable system administrators to detect attacks, diagnose errors, and recover from attacks. At the same time, logging too much data (CWE-779) can cause the same problems.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Base	223	Omission of Security-relevant Information	**Development Concepts (primary)699 Research Concepts (primary)1000**
ChildOf	Category	254	Security Features	Development Concepts699
ChildOf	Weakness Class	693	Protection Mechanism Failure	Research Concepts1000

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
2009-07-02			Internal CWE Team
Contributions			
Contribution Date	**Contributor**	**Organization**	**Source**
2009-07-02		Fortify Software	Content
	Provided code example and additional information for description and consequences.		

BACK TO TOP

Scanned Languages

Language	Hash Number	Change Date
Java	0125956323748501	7/21/2020
Common	0114668597102001	7/21/2020