

Upload Scan Report

Project Name	Upload
Scan Start	Friday, August 24, 2018 11:36:13 AM
Preset	Checkmarx Default
Scan Time	00h:02m:01s
Lines Of Code Scanned	216
Files Scanned	2
Report Creation Time	Friday, August 24, 2018 11:43:30 AM
Online Results	http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17
Team	CxServer
Checkmarx Version	8.7.0 HF1
Scan Type	Incremental
Source Origin	LocalPath
Density	5/100 (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](#)

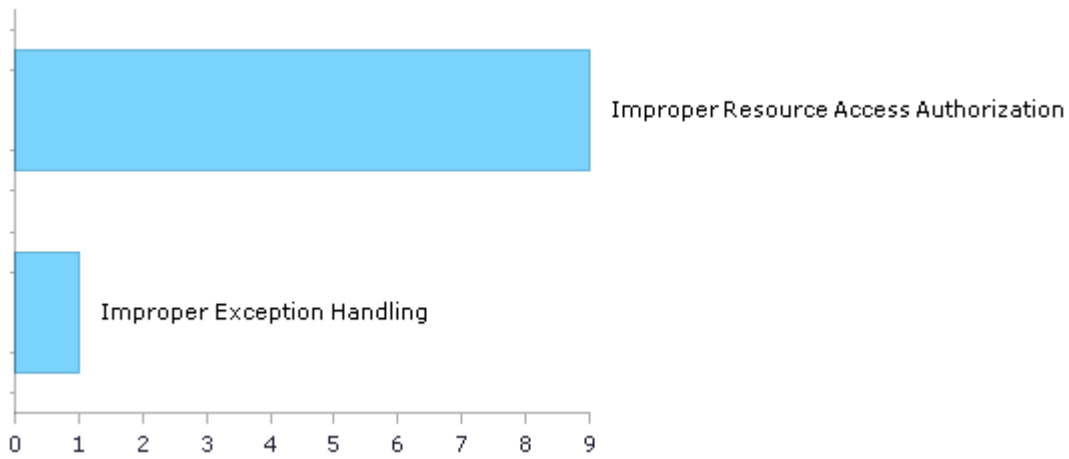
Result Summary



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	0	0
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	0	0
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	0	0
A6-Security Misconfiguration *	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
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	0	0
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	0	0
A5-Security Misconfiguration *	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	0	0
A6-Sensitive Data Exposure*	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	0	0
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	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 - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection	0	0
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*	1	0
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.2) - 6.5.8 - Improper access control*	0	0
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.	0	0
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.	9	0
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.	0	0
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.	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 - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)	0	0
AC-3 Access Enforcement (P1)*	9	0
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)*	0	0
SC-4 Information in Shared Resources (P1)	0	0
SC-5 Denial of Service Protection (P1)*	1	0
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	0	0
SI-11 Error Handling (P2)*	0	0
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 wasn't 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

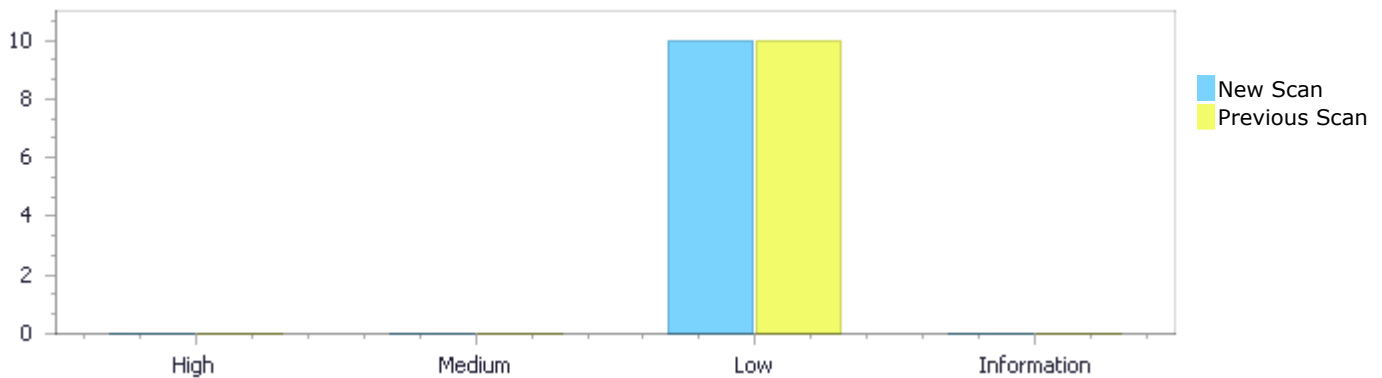
Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status

Compared to project scan from 8/24/2018 11:36 AM

	High	Medium	Low	Information	Total
New Issues	0	0	0	0	0
Recurrent Issues	0	0	10	0	10
Total	0	0	10	0	10

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	0	10	0	10
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	0	0	10	0	10

Result Summary

Vulnerability Type	Occurrences	Severity
Improper Resource Access Authorization	9	Low
Improper Exception Handling	1	Low

Scan Results Details

Improper Resource Access Authorization

Query Path:

Java\Cx\Java Low Visibility\Improper Resource Access Authorization Version:1

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	http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=2
Status	Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	84	84
Object	query	query

Code Snippet

File Name JdbcOwnerRepositoryImpl.java
Method public Collection<Owner> findByLastName(String lastName) throws
DataAccessException {

```
....  
84.         List<Owner> owners = this.namedParameterJdbcTemplate.query(
```

Improper Resource Access Authorization\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=3
Status	Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	103	103
Object	queryForObject	queryForObject

Code Snippet

File Name JdbcOwnerRepositoryImpl.java
Method public Owner findById(int id) throws DataAccessException {

```
.....
103.                owner =
this.namedParameterJdbcTemplate.queryForObject (
```

Improper Resource Access Authorization\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=4
Status	Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	123	123
Object	query	query

Code Snippet

File Name JdbcOwnerRepositoryImpl.java
Method public void loadPetsAndVisits(final Owner owner) {

```
.....
123.                final List<JdbcPet> pets =
this.namedParameterJdbcTemplate.query(
```

Improper Resource Access Authorization\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=5
Status	Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	164	164
Object	query	query

Code Snippet

File Name JdbcOwnerRepositoryImpl.java
Method public Collection<PetType> getPetTypes() throws DataAccessException {

```
.....
164.                return this.namedParameterJdbcTemplate.query(
```

Improper Resource Access Authorization\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?sc

Status [anid=1000043&projectid=17&pathid=6](#)
Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	183	183
Object	query	query

Code Snippet

File Name JdbcOwnerRepositoryImpl.java

Method public Collection<Owner> findAll() throws DataAccessException {

```
....  
183.             List<Owner> owners =  
this.namedParameterJdbcTemplate.query(
```

Improper Resource Access Authorization\Path 6:

Severity Low

Result State To Verify

Online Results <http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=7>

Status Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	156	156
Object	update	update

Code Snippet

File Name JdbcOwnerRepositoryImpl.java

Method public void save(Owner owner) throws DataAccessException {

```
....  
156.             this.namedParameterJdbcTemplate.update (
```

Improper Resource Access Authorization\Path 7:

Severity Low

Result State To Verify

Online Results <http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=8>

Status Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	208	208
Object	update	update

Code Snippet**File Name** JdbcOwnerRepositoryImpl.java**Method** public void delete(Owner owner) throws DataAccessException {

```
....  
208.                this.namedParameterJdbcTemplate.update("DELETE  
FROM visits WHERE id=:id", visit_params);
```

Improper Resource Access Authorization\Path 8:**Severity** Low**Result State** To Verify**Online Results** <http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=9>**Status** Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	210	210
Object	update	update

Code Snippet**File Name** JdbcOwnerRepositoryImpl.java**Method** public void delete(Owner owner) throws DataAccessException {

```
....  
210.                this.namedParameterJdbcTemplate.update("DELETE FROM  
pets WHERE id=:id", pet_params);
```

Improper Resource Access Authorization\Path 9:**Severity** Low**Result State** To Verify**Online Results** <http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=10>**Status** Recurrent

	Source	Destination
File	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	212	212
Object	update	update

Code Snippet**File Name** JdbcOwnerRepositoryImpl.java**Method** public void delete(Owner owner) throws DataAccessException {

```
....  
212.                this.namedParameterJdbcTemplate.update("DELETE FROM owners  
WHERE id=:id", owner_params);
```


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	http://cxsample.eastus.cloudapp.azure.com/CxWebClient/ViewerMain.aspx?scanid=1000043&projectid=17&pathid=1
Status	Recurrent

Method loadPetsAndVisits at line 115 of JdbcOwnerRepositoryImpl.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	JdbcOwnerRepositoryImpl.java	JdbcOwnerRepositoryImpl.java
Line	123	123
Object	query	query

Code Snippet

File Name JdbcOwnerRepositoryImpl.java
Method public void loadPetsAndVisits(final Owner owner) {

```
....
123.         final List<JdbcPet> pets =
this.namedParameterJdbcTemplate.query(
```

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
}
```



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 defaults 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](#)

Scanned Languages

Language	Hash Number	Change Date
Java	3039917160133120	5/16/2018
Common	0105849645654507	5/16/2018