

Select Medical

Summer 2023 Internship

Carson Kramer

# OWASP PENETRATION TEST

PREPARED FOR

VULNERABILITY MANAGEMENT TEAM

*August 4th, 2023*

# OWASP PENETRATION TEST

## Contents

Contents .....	2
Executive Summary .....	3
Detailed Finding Breakdown.....	<b>Error! Bookmark not defined.</b>
OWASP Category Descriptions .....	<b>Error! Bookmark not defined.</b>

# OWASP PENETRATION TEST

## Executive Summary

### Background

In the summer of 2023, Select Medical had Carson Kramer perform a penetration test of the OWASP Mutillidae II on AWS servers.

This test was designed to help Select Medical identify security gaps in their web application that could be exploited by threat actors.

Testing occurred from 8/1/23 to 8/2/23. This report contains the full results of the test, including detailed observations, OWASP Category, and recommendations for remediations.

OBSERVATION COUNT	
<b>6</b>	Critical Risk Observations
<b>1</b>	High Risk Observations
<b>12</b>	Medium Risk Observations
<b>56</b>	Low Risk Observations
<b>75</b>	<b>Total Observations</b>

TIMELINE	
07/31/2023	Kickoff
08/01/2023	Vulnerability Scan
08/04/2023	Report Delivery

The presence of a higher number of path disclosures and information disclosures compared to cross-site scripting (XSS) or SQL injection vulnerabilities can be attributed to several factors:

1. **Application Configuration:** Path disclosures and information disclosures often occur due to misconfigurations in the application or server settings. These may include directory listing enabled, verbose error messages, or inadequate access controls, leading to unintentional exposure of sensitive information.
2. **Error Handling:** Inadequate error handling practices can contribute to path and information disclosures. If error messages are not properly handled or sanitized, they may inadvertently reveal internal paths or sensitive data. Verbose error messages can provide attackers with valuable information about the system, making it easier for them to exploit vulnerabilities.
3. **Input Validation:** Cross-site scripting (XSS) and SQL injection vulnerabilities are typically the result of insufficient input validation and sanitization. If the application has strong input validation mechanisms in place, it may help reduce the risk of these types of vulnerabilities. However, path and information disclosures may still occur independently of input validation.

# OWASP PENETRATION TEST

4. **Application Design and Architecture:** The design and architecture of the application can also influence the occurrence of different types of vulnerabilities. If the application relies heavily on passing file paths, there may be a higher likelihood of unintentional path disclosures. Similarly, if the application handles and stores a significant amount of sensitive information, there may be an increased risk of information disclosures.

To address path disclosures and information disclosures, it is crucial to review and enhance your application's configuration, error handling practices, and access controls. Implementing appropriate security measures, such as disabling directory listing, sanitizing error messages, and implementing proper access controls, can help mitigate these vulnerabilities. Regular security assessments and testing are recommended to identify and address any existing vulnerabilities in your application.

## SCOPE

Testing occurred in Select Medical's AWS environment and was conducted from Qualys and Burp Suite

ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae

## FINDING

ID	TITLE	RISK	NUMBER	PROPORTION
OB-1	Cross-Site Scripting	Critical	2	3%
OB-2	SQL Injection	Critical	4	5%
OB-3	Information Disclosure	Critical -> Medium	19	25%
OB-4	Path Disclosure	Low	50	67%

## Detailed Finding Breakdown

### OB1 – Cross Site Scripting

OWASP CATEGORY	SEVERITY	NUMBER
A3 Injection	HIGH	2

#### THREAT

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

#### IMPACT

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

#### SOLUTION

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers. Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

# OWASP PENETRATION TEST

## ADDITIONAL RESOURCES

150001 Reflected Cross-Site Scripting (XSS) Vulnerabilities			
URL: http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/includes/pop-up-help-conte...			
Finding #	28442848* (932522147)	Web Application	OWASPTEST
Unique #	f3537025-ed83-4a92-8712-f79180a41eb0	Authentication	Not Used
Patch #	—		
Group	Cross-Site Scripting	Detection Date	01 Aug 2023 9:38AM GMT-0400
CWE	CWE-79	CVSS V3 Attack Vector	NETWORK
OWASP	A3 Injection		
WASC	WASC-8 CROSS-SITE SCRIPTING		
CVSS V3 Base	6.1	CVSS V3 Temporal	5.8

Figure 1 – Qualys cross-site scripting findings

The screenshot displays the Burp Suite Professional interface. The top panel shows a list of findings, with the selected finding displaying the following details:

- Finding #: 28442848\* (932522147)
- Unique #: f3537025-ed83-4a92-8712-f79180a41eb0
- Patch #: —
- Group: Cross-Site Scripting
- CWE: CWE-79
- OWASP: A3 Injection
- WASC: WASC-8 CROSS-SITE SCRIPTING
- CVSS V3 Base: 6.1
- CVSS V3 Temporal: 5.8

The bottom panel shows the HTTP request and response for the selected finding. The request is a POST to /mutillidae/index.php?page=echo.php. The response is an HTML page with the injected payload visible in the body.

Figure 2 – Burp Suite cross-site scripting example

# OWASP PENETRATION TEST

## OB2 – SQL Injection

OWASP CATEGORY	SEVERITY	NUMBER
A3 Injection	<b>HIGH</b>	4

### THREAT

SQL injection enables an attacker to modify the syntax of a SQL query in order to retrieve, corrupt, or delete data. This is accomplished by manipulating query criteria in a manner that affects the query's logic. The typical causes of this vulnerability are lack of input validation and insecure construction of the SQL query. Queries created by concatenating strings with SQL syntax and user-supplied data are prone to this vulnerability. If any part of the string concatenation can be modified, then the meaning of the query can be changed.

### IMPACT

The scope of a SQL injection exploit varies greatly. If any SQL statement can be injected into the query, then the attacker has the equivalent access of a database administrator. This access could lead to theft of data, malicious corruption of data, or deletion of data.

### SOLUTION

SQL injection vulnerabilities can be addressed in three areas: input validation, query creation, and database security. All input received from the Web client should be validated for correct content. If a value's type or content range is known beforehand, then stricter filters should be applied. For example, an email address should be in a specific format and only contain characters that make it a valid address; or numeric fields like a U.S. zip code should be limited to five digit values.

Prepared statements (also referred to as parameterized queries) provide strong protection from SQL injection. Prepared statements are precompiled SQL queries whose parameters can be modified when the query is executed. Prepared statements enforce the logic of the query and will fail if the query cannot be compiled correctly. Programming languages that support prepared statements provide specific functions for creating queries. These functions are more secure than string concatenation for assigning user-supplied data to a query.

Stored procedures are precompiled queries that reside in the database. Like prepared statements, they also enforce separation of query data and logic. SQL statements that call stored procedures should not be created via string concatenation, otherwise their security benefits are negated. SQL injection exploits can be mitigated by the use of Access Control Lists or role-based access within the database. For example, a read-only account would prevent an attacker from modifying data, but would not prevent the user from viewing unauthorized data. Table and row-based access controls potentially minimize the scope of a compromise, but they do not prevent exploits.

# OWASP PENETRATION TEST

## ADDITIONAL RESOURCES

150003 SQL Injection			
URL: http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/includes/pop-up-help-conte...			
Finding #	28442844* (932521545)	Web Application	OWASPTTEST
Unique #	713d1a50-08a7-4dad-b61b-43208040bf52	Authentication	Not Used
Patch #	-		
Group	SQL Injection	Detection Date	01 Aug 2023 9:38AM GMT-0400
CWE	CWE-89	CVSS V3 Attack Vector	NETWORK
OWASP	A3 Injection		
WASC	WASC-19 SQL INJECTION		
CVSS V3 Base	10	CVSS V3 Temporal	9

Figure 3 – Qualys SQL injection findings

#1 Request

GET http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/includes/pop-up-help-context-generator.php?pagename='%20onEvent%3DX150019208Y1\_1Z%20  
Referer: http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/  
Cookie: showhints=0; PHPSESSID=he0c9174f2r9vtd5fjd6af9o40;  
Host: ec2-3-142-74-110.us-east-2.compute.amazonaws.com  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_5) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1.1 Safari/605.1.15  
Accept: \*/\*

Click this [link](#) to try to reproduce the vulnerability using above payload. Note that clicking this link may not lead to visible results, either because the vulnerability requires context to be previously set (authentication, cookies...) or because the exploitation of the vulnerability does not lead to any visible proof.

#1 Response

comment: Response status: 200  
bel">File</td><td class="error-detail">/var/www/html/mutillidae/classes/MySQLHandler.php</td></tr><tr><td class="error-label">Message</td><td class="error-detail">/var/www/html/mutillidae/classes/MySQLHandler.php on line 230: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'onEvent=X150019208Y1\_1Z ' ORDER BY page\_help.order\_preference' at line 10 Query: &#x0d;&#x0a;&#x09;&#x09;&#x09;SELECT&#x20;CONCAT&#x20;

Export...

Figure 4 – Qualys SQL injection payload



# OWASP PENETRATION TEST

## OB3 – Information Disclosure

OWASP CATEGORY	SEVERITY	NUMBER
A5 Security Misconfiguration	<b>HIGH to Medium</b>	19

### THREAT

The login form's default action contains a link that is not submitted via HTTPS (HTTP over SSL).

Additionally, jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. The web application is using a JavaScript library that is known to contain at least one vulnerability.

A link is also functional over an insecure, HTTP connection. No redirection to HTTPS occurs. Note that this QID is reported for 200/OK responses as well as 4xx and 5xx responses.

### IMPACT

Sensitive data such as authentication credentials should be encrypted when transmitted over the network. Otherwise they are exposed to sniffing attacks.

Attackers could potentially exploit the vulnerability in the JavaScript library. The impact of a successful exploit depends on the nature of the vulnerability and how the web application makes use of the library.

Data sent over a non-HTTPS connection is unencrypted and vulnerable to network sniffing attacks that can expose sensitive or confidential information. This includes non-secure cookies and other potentially sensitive data contained in HTTP headers. Even if no sensitive data is transmitted, man-in-the-middle (MITM) attacks are possible over non-HTTPS connections. An attacker who exploits MITM can intercept and change the conversation between the client (e.g., web browser, mobile device, etc.) and the server.

### SOLUTION

Change the login form's action to submit via HTTPS.

Check the vendor's security advisories related to the vulnerable version of the library.

Ensure that all links are accessible over HTTPS only. The most secure design is for the application to listen and respond only to encrypted HTTPS requests. Alternatively, if non-HTTPS requests are accepted, the server should redirect these requests to HTTPS using a 301 or 302 response. It is also strongly recommended to use HTTP Strict Transport Security (HSTS) so that web browsers are instructed to use only HTTPS when making requests to the server.

# OWASP PENETRATION TEST

## ADDITIONAL RESOURCES

<b>150053 Login Form Is Not Submitted Via HTTPS</b>			
URL: <a href="http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/index.php?page=login.php">http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/index.php?page=login.php</a>			
Finding #	28442820* (932521533)	Web Application	OWASPTEST
Unique #	<a href="#">430f3056-254e-4f04-a261-ddd79b78283a</a>	Authentication	Not Used
Patch #	–		
Group	Information Disclosure	Detection Date	01 Aug 2023 9:38AM GMT-0400
CWE	<a href="#">CWE-523</a>	CVSS V3 Attack Vector	NETWORK
OWASP	<a href="#">A5 Security Misconfiguration</a>		
WASC	<a href="#">WASC-4 INSUFFICIENT TRANSPORT LAYER PROTECTION</a>		
CVSS V3 Base	6.5	CVSS V3 Temporal	6.2

Figure 5 – Qualys information disclosure findings

**#1 Request**

POST <http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/index.php?page=login.php>  
Host: ec2-3-142-74-110.us-east-2.compute.amazonaws.com  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_5) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1.1 Safari/605.1.15  
Accept: \*/\*  
Content-Type: application/x-www-form-urlencoded

*Click this [link](#) to try to reproduce the vulnerability using above payload. Note that clicking this link may not lead to visible results, either because the vulnerability requires context to be previously set (authentication, cookies...) or because the exploitation of the vulnerability does not lead to any visible proof.*

**#1 Response**

comment: Parent URL of Login Form is : <http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/index.php?page=login.php>  
Login Form Is Not Submitted Via HTTPS


Export...

Figure 6 – Qualys information disclosure payload

# OWASP PENETRATION TEST

## OB4 – Path Disclosure

OWASP CATEGORY	SEVERITY	NUMBER
A3 Injection	Low	50

### THREAT

The Web server presents a directory listing.

A file, directory, or directory listing was discovered on the Web server. These resources are confirmed to be present based on our logic. Some of the content on these files might have sensitive information.

### IMPACT

All file names in this directory are exposed.

The contents of this file or directory may disclose sensitive information.

### SOLUTION

The presence of a browsable directory does not necessarily imply a vulnerability. Determine if the directory listing is intended to be displayed. Verify that no files in the directory contain content that should not be served by the Web application.

It is advised to review the contents of the disclosed files. If the contents contain sensitive information, please verify that access to this file or directory is permitted. If necessary, remove it or apply access controls to it.

# OWASP PENETRATION TEST

## ADDITIONAL RESOURCES


 <b>150023 Directory Listing</b>			
URL: <a href="http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/javascript/jQuery/colorbox...">http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/javascript/jQuery/colorbox...</a>			
Finding #	28442752* (932521499)	Web Application	OWASPTTEST
Unique #	<b>f1009732-47da-4aff-a12b-1bdbae2e3882</b>	Authentication	Not Used
Patch #	-		
Group	Path Disclosure	Detection Date	01 Aug 2023 9:38AM GMT-0400
CWE	<b>CWE-548</b>	CVSS V3 Attack Vector	NETWORK
OWASP	<b>A1 Broken Access Control</b>		
WASC	<b>WASC-16 DIRECTORY INDEXING</b>		
CVSS V3 Base	5.3	CVSS V3 Temporal	5

Figure 7 – Qualys path disclosure findings

#1 Request

GET http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/javascript/jQuery/colorbox/images/ie6/  
Referer: http://ec2-3-142-74-110.us-east-2.compute.amazonaws.com/mutillidae/  
Cookie: showhints=0; PHPSESSID=he0c9174f2r9vtd5fjd6af9o40;  
Host: ec2-3-142-74-110.us-east-2.compute.amazonaws.com  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_5) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1.1  
Safari/605.1.15  
Accept: \*/\*

Click this [link](#) to try to reproduce the vulnerability using above payload. Note that clicking this link may not lead to visible results, either because the vulnerability requires context to be previously set (authentication, cookies...) or because the exploitation of the vulnerability does not lead to any visible proof.

#1 Response

comment: This directory was discovered during the crawl phase.

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html><head>  
<title>Index of /mutillidae/javascript/jQuery/colorbox/images/ie6</title>  
</head>  
<body>  
<h1>Index of /mutillidae/javascript/jQuery/colorbox/images/ie6</h1>  
<table>  
<tbody><tr><th valign="top"></th><th><a href="?C=N;O=D">Name</a></th><th><a href="?C=M;O=A">Last modified</a></th><th><a href="?C=S;O=A">Size</a></th><th><a href="?C=D;O=A">Description</a></th>  
</tr>  
<tr><th colspan=


 Export...

Figure 8 – Qualys SQL injection payload

# OWASP PENETRATION TEST

## OWASP Category Descriptions

The following table summarizes the OWASP categories and descriptions:

OWASP CATEGORY	DESCRIPTION
<b>Broken Access Control</b>	A01:2021 - Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.
<b>Cryptographic Failures</b>	A02:2021 - The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation (GDPR), or regulations, e.g., financial data protection such as PCI Data Security Standard (PCI DSS).
<b>Injection</b>	A03:2021 - Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
<b>Insecure Design</b>	A04:2021 - Insecure design is a broad category representing different weaknesses, expressed as "missing or ineffective control design." Insecure design is not the source for all other Top 10 risk categories. There is a difference between insecure design and insecure implementation. We differentiate between design flaws and implementation defects for a reason, they have different root causes and remediation. A secure design can still have implementation defects leading to vulnerabilities that may be exploited. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.
<b>Security Misconfiguration</b>	A05:2021 - Commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information.
<b>Vulnerable and Outdated Components</b>	A06:2021 - Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
<b>Identification and Authentication Failures</b>	A07:2021 - Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
<b>Software and Data Integrity Failures</b>	A08:2021 - Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system

# OWASP PENETRATION TEST

	compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application.
<b>Security Logging and Monitoring Failures</b>	A09:2021 - Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data.
<b>Server-Side Request Forgery (SSRF)</b>	A10:2021 - SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL). As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the severity of SSRF is becoming higher due to cloud services and the complexity of architectures.