xxxxx XXXX PLATFORM

OWASP TOP10 Penetration Test Report

Version: 1.0
Date: xx, 2022
Author: Vadim Balan OSCP, CISSP

Description

Title	OWASP TOP10 Penetration Test Report
Version	1.0
State	Final
Author	Vadim BALAN (FeelIT Services)

Evolutions

Version	Date	Author	Verification	Evolution
1.0	August 24, 2022	Vadim BALAN	Quality service	Final

Recipients

Name	Company
xxxxx	Xxxx LLC
xxxxx	Xxxx LLC
xxxxx	Xxxx LLC
xxxxx	Xxxx LLC

Classification

	Forbidden	Controlled	Restricted	Free	Validity
External		X			Unlimited
Internal			X		Unlimited

CONTENTS

STATEMENT OF CONFIDENTIALITY

This document contains confidential information and has been provided to Xxxx LLC as a deliverable of the agreed statement of work between FeelIT Services and Xxxx. The sole purpose of this document is to provide Xxxx with the results of the penetration testing exercise. Xxxx may at its discretion distribute this report to other third parties who have a need to know, provided they are under confidentiality obligations to Xxxx. Each recipient agrees that, prior to reading this document, it shall not distribute or use the information contained herein and any other information regarding Xxxx for any purpose other than those stated. This document, and any other Xxxx related information provided, shall remain the sole property of Xxxx and may not be copied, reproduced, or distributed without the prior written consent of Xxxx.

1. EXECUTIVE SUMMARY

FeelIT Services was engaged by Xxxx to perform a Grey Box penetration test according to OWASP TOP10 Web Application Security Risks on their external facing web application – xxxx Xxxxxxxx Platform. The penetration testing was performed between ..

As result of executing the external penetration test, the Penetration Tester identified a few security concerns that may have an isolated impact on the privacy of customer data. The medium and low severity issues related to authentication and authorization controls should be addressed to improve the security of the application.

The external penetration test didn't identify any security vulnerabilities with high-risk impact on Xxxx application or customer data.

It should be noted that the penetration test did not include advanced persistent threat types of attacks which combined with social engineering techniques either on Xxxx staff or Company's customers, may increase the attack surface that a real word attacker could have.

The detailed results and conclusions of the application and network penetration tests are described in Chapter 4. OWASP TOP10 Test results.

Evaluation of OWASP TOP 10 Web Application Security Risks

Name	Details	Evaluation
A01-Broken Access Control	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, modify other users' data, change access rights, etc.	↘ Insufficient
A02-Cryptographic Failures	Many web applications and APIs have issues related to cryptography which exposes sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes.	↗ Requires improvements
A03-Injection	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.	↑ Good
A04-Insecure Design	Insecure design is a broad category representing different weaknesses, expressed as "missing or ineffective control design." An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.	↑ Good
A05-Security Misconfiguration	Security misconfiguration is the commonly seen issue. This is 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.	↗ Requires improvements
A06-Vulnerable and Outdated Components	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application.	↗ Requires improvements
A07-Identification and Authentication Failures	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.	↗ Requires improvements
A08-Software and Data Integrity Failures	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).	↑ Good
A09-Security Logging and Monitoring Failures	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.	↑ Good
A10-Server-Side Request Forgery	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).	↑ Good

↑ Good | ↗ Required improvements | ↘ Insufficient | ↓ Critical

2. SCOPE

The penetration test was performed on Xxxx Xxxxxxxx web application. To assist the Penetration Tester during the assignment, Xxxx representatives provided technical information of the infrastructure, diagrams and API documentation.

The Penetration Tester performed application layer and network layer tests against the external targets listed below.

Network Targets

Nr.	IP / Network address	Network mask	Int./Ext.	Comments
1.	staging.xxxxxxxx.com	N/A	Ext.	Xxxx stage web application server

Application Targets

Nr.	URL / Application	Port / Protocol	Function	Int./Ext.	Comments
1.	staging.xxxxxxxx.com	https	Xxxx Xxxxxxxx Platform - Xxxx main web application.	Ext.	Application pen-testing was performed on staging environment.

3. SUMMARY OF FINDINGS

Identified security issues for staging.xxxxxxxx.com:

Severity	Vulnerability	OWASP TOP 10 Category
	**4.1.1. Unauthorized access to …**	A01:2021-Broken Access Control
	**4.2.1. Web server supports TLS 1.0 and TLS 1.1 https connections**	A02:2021-Cryptographic Failures
	**4.4.1. [description of the issue]**	A04:2021- Insecure Design
	**4.6.1. Client-side application contains outdated JS libraries with known vulnerabilities**	A06:2021- Using Components with Known Vulnerabilities
	**4.6.2. Vulnerable version of Apache Tomcat**	A06:2021- Using Components with Known Vulnerabilities
	**4.7.1. User session is not terminated at logout**	A07:2021-Identification and Authentication Failures
	**4.7.2. Session cookie without secure flag set**	A07:2021-Identification and Authentication Failures

4. OWASP TOP10 TEST RESULTS

4.1. A01:2021-Broken Access Control

A01:2021- Broken Access Control	Overall evaluation	↘ **Insufficient**
Attack vectors	Security weakness	Impact
Exploitation of access control is a core skill of attackers. Access control is detectable using manual means, or possibly through automation for the absence of access controls in certain frameworks.	Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers. Manual testing is the best way to detect missing or ineffective access control, including HTTP method (GET vs PUT, etc), controller, direct object references, etc.	The technical impact is attackers acting as users or administrators, or users using privileged functions, or creating, accessing, updating or deleting every record.
Testing procedures		Test results
▪ Violation of the principle of least privilege or deny by default, where access should only be granted for capabilities, roles, or users, but is available to anyone. ▪ Bypassing access control checks by modifying the URL (parameter tampering or force browsing), internal application state, or the HTML page, or by using an attack tool modifying API requests. ▪ Permitting viewing or editing someone else's account, by providing its unique identifier (insecure direct object references) ▪ Accessing API with missing access controls for POST, PUT and DELETE. ▪ Elevation of privilege. Acting as a user without being logged in or acting as an admin when logged in as a user. ▪ CORS misconfiguration allows API access from unauthorized/untrusted origins. ▪ Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user.		[description of test results]

IDENTIFIED SECURITY VULNERABILITIES		
Description	Severity/Impact	Recommendations
4.1.1. Unauthorized access to .. [description]	 **Medium** [description]	[description]

4.2. A02:2021-Cryptographic Failures

A02:2021-Cryptographic Failures	Overall evaluation	↗ **Requires improvements**
Attack vectors	Security weakness	Impact
Rather than directly attacking crypto, attackers steal keys, execute man-in-the-middle attacks, or steal clear text data off the server, while in transit, or from the user's client, e.g. browser. A manual attack is generally required. Previously retrieved password databases could be brute forced.	The most common flaw is simply not encrypting sensitive data. When crypto is employed, weak key generation and management, and weak algorithm, protocol and cipher usage is common, particularly for weak password hashing storage techniques. For data in transit, server-side weaknesses are mainly easy to detect, but hard for data at rest.	Failure frequently compromises all data that should have been protected. Typically, this information includes sensitive personal information (PII) data such as personal data, credentials and credit cards, which often require protection as defined by laws or regulations such as the local privacy laws.
Testing procedures		Test results
▪ Manual analysis of identified data in transit and at rest and identifying weaknesses that may xxxx to the compromise of its confidentiality. ▪ Identifying any data transmitted in clear text by usage of unencrypted protocols such as HTTP, SMTP, FTP;		▪ [description of test results]

▪ Testing for old or weak cryptographic algorithms used either by default or in older code. Testing the web server https configuration with ssllabs.com. Testing if encryption is enforced, e.g. are any user agent (browser) security directives or headers missing; ▪ Testing for default crypto keys in use, weak crypto keys generated or re-used; ▪ Testing if the user agent verifies if the received server certificate is valid.	

IDENTIFIED SECURITY VULNERABILITIES		
Description	Severity/Impact	Recommendations
4.2.1. Web server supports TLS 1.0 and TLS 1.1 https connections [description]	 **Low** [description]	[description]

4.3. A03:2021-Injection

A03:2021-Injection	Overall evaluation	↑ **Good**
Attack vectors	Security weakness	Impact

Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter.	Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers. Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.	Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes xxxx to complete host takeover.

Testing procedures	Test results
▪ Testing if the user-supplied data for validation, filtering, or sanitization by the web application on the client-side and server-side; ▪ Testing if dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter; ▪ Testing with Burp Suite Pro for common injections like SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL); ▪ All the injection type tests were performed on testing environment.	▪ [description of test results]

IDENTIFIED SECURITY VULNERABILITIES

No Injection type vulnerabilities were identified for the tested scope.

4.4. A04:2021-Insecure Design

A04:2021- Insecure Design	Overall evaluation	↑ **Good**
Attack vectors	Security weakness	Impact
Attack vectors can vary for each case and will have at a basis some business logic and insecure application design weakness findings.	An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks	Insecure design may xxxx to complete or partial security compromise and may cause unauthorized data access, data loss or data corruption, host takeover.

Testing procedures	Test results
[description of testing procedures]	○ [description of test results].

IDENTIFIED SECURITY VULNERABILITIES		
Description	Severity/Impact	Recommendations
4.4.1. [description of the issue]	 **Low** [description]	[description]

4.5. A05:2021-Security Misconfiguration

A5:2021- Security Misconfiguration	Overall evaluation	↗ **Requires improvements**
Attack vectors	Security weakness	Impact
Attackers will often attempt to exploit unpatched flaws or access default accounts, unused pages, unprotected files and directories, etc. to gain unauthorized access or knowledge of the system.	Security misconfiguration can happen at any level of an application stack, including the network services, platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage.	Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise.

Testing procedures	Test results
▪ Testing for missing appropriate security hardening across any part of the application stack, or improperly configured permissions for services;	[description of test results]

▪ Testing if unnecessary features are enabled or installed (e.g. unnecessary ports, services, pages, accounts, or privileges); ▪ Testing if default accounts and their passwords still enabled and unchanged; ▪ Testing if error handling reveals stack traces or other overly informative error messages to users; ▪ Testing the upgraded systems, latest security features are disabled or not configured securely; ▪ Scanning the security settings in the application servers, application frameworks, libraries, etc. not set to secure values; ▪ Testing if the server does not send security headers or directives or they are not set to secure values;	

IDENTIFIED SECURITY VULNERABILITIES
No Security Misconfiguration type vulnerabilities were identified for the tested scope.

4.6. A06:2021-Using Components with Known Vulnerabilities

A06:2021- Using Components with Known Vulnerabilities	Overall evaluation	↗ **Requires improvements**
Attack vectors	Security weakness	Impact
While it is easy to find already-written exploits for many known vulnerabilities, other vulnerabilities require concentrated effort to develop a custom exploit.	Prevalence of this issue is very widespread. Component-heavy development patterns can xxxx to development teams not even understanding which components they use in their application or API, much less keeping them up to date.	While some known vulnerabilities xxxx to only minor impacts, some of the largest breaches to date have relied on exploiting known vulnerabilities in components. Depending on the assets you are protecting, perhaps this risk should be at the top of the list.
Testing procedures		Test results

▪ Analyzing the Web Application fingerprints to identify the versions and software components of the web application infrastructure (both client-side and server-side). This includes components directly used as well as nested dependencies; ▪ Checking if software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.	▪ [description of test results]

IDENTIFIED SECURITY VULNERABILITIES		
Description	Severity/Impact	Recommendations
4.6.1. Client-side application contains outdated JS libraries with known vulnerabilities [description]	 **Low** [description]	[description]
4.6.2. Vulnerable version of Apache Tomcat [description]	 **Low** [description]	[description]

4.7. A07:2021-Identification and Authentication Failures

A07:2021-Identification and Authentication Failures	Overall evaluation	↗ **Requires improvements**

Attack vectors	Security weakness	Impact
Attackers have access to hundreds of millions of valid username and password combinations for credential stuffing, default administrative account lists, automated brute force, and dictionary attack tools. Session management attacks are well understood, particularly in relation to unexpired session tokens.	The prevalence of broken authentication is widespread due to the design and implementation of most identity and access controls. Session management is the bedrock of authentication and access controls and is present in all stateful applications. Attackers can detect broken authentication using manual means and exploit them using automated tools with password lists and dictionary attacks.	Attackers have to gain access to only a few accounts, or just one admin account to compromise the system. This may allow social security fraud, and identity theft, or disclose legally protected highly sensitive information.

Testing procedures	Test results
▪ Testing the applications for permitting brute force or other automated attacks on the authentication interfaces; ▪ Manual Testing for default, weak, or well-known passwords, such as "Password1" or "admin/admin"; ▪ Testing for weak or ineffective credential recovery; ▪ Testing for use of plain text, encrypted, or weakly hashed passwords; missing or ineffective multi-factor authentication; exposed Session IDs in the URL; rotation of Session IDs after successful login; ▪ Testing the properly invalidation of Session IDs: User sessions or authentication tokens properly invalidation during logout or a period of inactivity; ▪ Testing the security of the server-side built-in session manager that should generate new random session IDs with high entropy after login and if necessary, after specific user actions (like password change); ▪ Testing for bypassing session management schema, cookies attributes, session fixation attacks, cross site request forgery and session puzzling vulnerabilities;	▪ [description of test results]

IDENTIFIED SECURITY VULNERABILITIES

Description	Severity/Impact	Recommendations
4.7.1. User session is not terminated at logout [description]	 **Low** [description]	[description]
4.7.2. Session cookie without secure flag set [description]	 **Low** [description]	[description]

4.8. A08:2021-Software and Data Integrity Failures

A08-Software and Data Integrity Failures	Overall evaluation	↑ **Good**
Attack vectors	Security weakness	Impact
Attackers could potentially upload their own updates to be distributed and run on all installations. Another example is where objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization. An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise.	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). Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application.	Unauthorized access to sensitive information, account takeover or complete host takeover, caused by malicious code, backdoors nested into the application.

Testing procedures	Test results
[description of testing procedures]	▪ [description of testing procedures]
IDENTIFIED SECURITY VULNERABILITIES	
No Server Side Request Forgery type vulnerabilities were identified for the tested scope.	

4.9. A09:2021-Insufficient Logging & Monitoring

A09:2021- Insufficient Logging & Monitoring	Overall evaluation	↑ **Good**
Attack vectors	Security weakness	Impact
Exploitation of insufficient logging and monitoring is the bedrock of nearly every major incident. Attackers rely on the lack of monitoring and timely response to achieve their goals without being detected.	One strategy for determining if you have sufficient monitoring is to examine the logs following penetration testing. The testers' actions should be recorded sufficiently to understand what damages they may have inflicted.	Most successful attacks start with vulnerability probing. Allowing such probes to continue can raise the likelihood of successful exploit to nearly 100%.
Testing procedures		Test results
[description of testing procedures]		▪ [description of testing procedures]
IDENTIFIED SECURITY VULNERABILITIES		

No issues related to insufficient logging and monitoring were identified for the tested scope.

4.10. A10:2021-Server-Side Request Forgery

A10:2021-Server-Side Request Forgery	Overall evaluation	↑ **Good**
Attack vectors	Security weakness	Impact
Attacker may craft a special malicious application request to collect additional technical information about server infrastructure and internal services, to expose sensitive date, to access metadata storage of cloud services, to achieve Remote Code Execution or Denial of Service.	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).	Unauthorized actions on the resources that a normally not achievable from the public, with objective of account or host takeover, denial of service, data compromise.
Testing procedures		Test results
▪ Automated Scans for the presence of SSRF issues; ▪ Manually review the scan results and performing manual tests for the flagged and suspicious results;		▪ [description]
IDENTIFIED SECURITY VULNERABILITIES		
No Server-Side Request Forgery type vulnerabilities were identified for the tested scope.		

5. PENETRATION TESTING RESULTS

During this assignment, the Penetration Tester successfully executed a couple of vulnerability exploitations. Some important technical details and proof-of-concepts for exploitation of discovered vulnerabilities are presented in this section to facilitate better technical understanding and quicker fixing of discovered vulnerabilities.

vuln #4.1.1. Unauthorized access to …

[exhaustive description and additional technical details on the exploited vulnerability]

Vuln #4.4.1. …

[exhaustive description and additional technical details on the exploited vulnerability]

APPENDIX A – PENETRATION TESTING METHODOLOGY

The Consultant conducted the penetration testing using the methodology based on OWASP Web Security Testing Guide to test for the OWASP Top10 Web Application Security Risk. The description of the testing procedures is presented in Chapter 4. OWASP TOP10 test results.

The following excerpt provides a high-level description of the phases the Consultant went through while testing Xxxx environment.

Phase 1 – Scanning

The test will typically begin with reconnaissance using several different types of scans with the overarching goal to learn more about the target application and find opportunities to access and interact with the target application and underlying system. These scans include:

- **Application Version scanning** - Attempting to determine both the version and type of services available.
- **Application Vulnerability scanning** - The most crucial part of the scanning process, since it attempts to identify whether the target systems are affected by one of thousands potential vulnerabilities, which could include misconfiguration, insecure development or an unpatched service.

Phase 2 - Exploitation

Within this phase, the tester uses the gathered knowledge of the target system and application to demonstrate how the presence of vulnerabilities may be used by malicious individuals to penetrate the environment. Often the tester will chain several types of exploit together with a goal of breaking through layers of defenses, circumventing existing security measures, and gaining control of the target system by elevating their level of access. Typically, the process incorporates the following steps:

- **Existing proof of concept code / tools** - The tester searches proof of concept code within their repository or from publicly available sources to test the identified vulnerability.
- **Custom code / tool development** - Under some circumstances it may be more expedient and cost effective for the tester to develop their own tools to exploit identified vulnerabilities.
- **Testing** - Custom code is tested to ensure that there are no adverse side-effects from using it, as well as ensuring that the code is effective. This process may xxxx to further customization of the concept code / tool.
- **Exploitation** - The code / tool is used against the target system in an attempt to compromise the target system and subvert the existing security controls.
- **Verification** - Establishing whether the vulnerability is exploitable and may negatively impact the security of the target system.

The tools used during penetration testing included:

Tool	Function
Kali Linux	Operating System with opensource tools
Nessus Professional	Vulnerability Scanner
Nikto	HTTP vulnerability scanner
Burp Suite Professional	HTTP Protocol Analysis. Web application vulnerability scanning and exploitation
Metasploit	Vulnerability scanning and exploitation
custom scripts and code	scripts coded by the Consultant to improve the analysis

APPENDIX B – CONSTULTANT'S PROFILE

Name: Vadim Balan

Role: Responsible for performing OWASP TOP10 security testing and document the Penetration Test Report.

Qualifications:

OSCP (Offensive Security Certified Professional)
Certificate ID: OS-101-031264 (Issued: 12 September 2017)

CISSP (Certified Information Systems Security Professional)
Certificate ID: 409710 (Certified since 2014)

TOGAF (The Open Group Architecture Framework) Certified
Certificate ID: 86836 (Issued: 22 July 2014)

Experience: Over 13 years of experience in information security consulting and IT audit services. Over 7 years of experience in Penetration Testing and Information Security Risk Assessments.

50+ executed penetration testing assignments for systems and infrastructures of various complexity and technologies.

Contacts: Email: xxxxxxxxxxxxxx.com