|  |
| --- |
|  |
| Client January 2018:  Penetration test report |
| Date: 09th March 2018  Version: v1.0  Prepared for: Kutlymurat Mambetniyazov  Prepared by: Imya Familiya, Imya Familiya, …, CS-1901  CONFIDENTIAL |

Contents

[Statement of Confidentiality 3](#_Toc96478584)

[1. Executive Summary 3](#_Toc96478585)

[1.1 Summary of findings 4](#_Toc96478586)

[2. Project Approach 5](#_Toc96478587)

[2.1 Rules of engagement 5](#_Toc96478588)

[2.2 Penetration testing methodology 5](#_Toc96478589)

[2.2.1 Open source intelligence gathering and network scanning 5](#_Toc96478590)

[2.2.2 Network Mapping 5](#_Toc96478591)

[2.2.3 Vulnerability testing 5](#_Toc96478592)

[2.2.4 Manual verification 6](#_Toc96478593)

[2.2.5 Vulnerability Exploitation 6](#_Toc96478594)

[3. Findings and recommendations key 7](#_Toc96478595)

[4.0 Network penetration test 8](#_Toc96478596)

[4.1 Scope 8](#_Toc96478597)

[4.2 Findings breakdown 9](#_Toc96478598)

[4.3 Findings Summary 10](#_Toc96478599)

[5.0 Further information 12](#_Toc96478600)

[5.1 Critical / High severity findings 12](#_Toc96478601)

[5.2 Medium severity findings 15](#_Toc96478602)

[5.3 Low severity findings 16](#_Toc96478603)

[5.4 Informational severity findings 18](#_Toc96478604)

[6.0 Identified Services and Open Network Ports 19](#_Toc96478605)

[7.0 OWASP Top Ten 2021 20](#_Toc96478606)

[Appendix A – Pentesting process 21](#_Toc96478607)

[Appendix B – Reference documents 22](#_Toc96478608)

Statement of Confidentiality

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

# 1. Executive Summary

Penetration testing is focused on finding security vulnerabilities in a target environment that could let an attacker penetrate the network or computer systems. The goal of penetration testing is to actually compromise a target system and ultimately steal sensitive information. This typically requires tools and techniques very similar to those that an attacker would use.

The penetration test was conducted against external and internal hosts provided by Customer on the 13 Jan 2021. The assessment was conducted in a manner that simulated a malicious individual who has access to the Customer’s internal network over the Internet connection with aim to determine whether an attacker could compromise Customers’ defence.

The tests were carried out externally outside the Customer’s premise. The best practice OSSTMM (Open Source Security Testing Methodology Manual), OWASP (Open Web Application Security Project), NIST and ISACA penetration testing and auditing standards and guidelines were used. Testing was conducted against the supporting environment such as operating system. Automated and manual techniques were used to evaluate the security of the target systems.

Initially detected vulnerabilities:

## 1.1 Summary of findings

Penetration testing was conducted against external and internal environments in the networks provided by Customer on 13 Jan 2021 with the understanding that this would be the scope for the engagement.

1. Source NAT:

a. 0.0.0.0

Destination / Static NAT:

a. 0.0.0.0

2. Corporate website

[https://astanait.edu.kz](https://arx.com.ua/)

3. AXON front-end

https://murat.one

https://astanait.edu.kz

4. Integration platform

[https://astanait.edu.kz](https://arx.com.ua/)

Web-services:

[https://astanait.edu.kz](https://arx.com.ua/)

5. Web Shop

[https://astanait.edu.kz](https://arx.com.ua/)

6. Remote Access (Cisco AnyConnect Secure Mobility Client)

[https://astanait.edu.kz](https://arx.com.ua/)

7. Exchange ActiveSync/OWA

[https://astanait.edu.kz](https://arx.com.ua/)

[https://astanait.edu.kz](https://arx.com.ua/)

[https://astanait.edu.kz](https://arx.com.ua/)

8. Online Payments

[https://astanait.edu.kz](https://arx.com.ua/)

Pentester identified 2 vulnerability rated as HIGH, 8 as MEDIUM and 5 as LOW in the hosts presented for the test by Customer.

It is strongly recommended to implement remedial actions for the found vulnerabilities.

The best practice would be to conduct the regression test.

# 2. Project Approach

## 2.1 Rules of engagement

Prior to the engagement Pentester established the rules of the engagement for the assessment. These rules provided the permission to conduct testing and outlined the procedures for notification of vulnerability scanning, notification of vulnerabilities and vulnerability exploitation. The testing was performed over the Internet connection provided by customer on 13 Jan 2021.

## 2.2 Penetration testing methodology

The test was done using a combination of manual and automated tools and techniques to identify vulnerabilities within the target environment and exploit them. Denial of Service and Social Engineering attacks were deemed out of scope during this test.

Following steps were carried out during this test.

* Open source intelligence gathering;
* Network mapping;
* Vulnerability testing;
* Manual verification;
* Vulnerability exploitation

### 2.2.1 Open source intelligence gathering and network scanning

Subdomain enumeration was omitted at this stage, as the scope was limited to several hosts only. Nmap port scans were performed to gather information about enabled services. Results of the enumeration are illustrated below:

*Figure 1– Nmap scanning process*

After initial port scanning, nmap scripting engine (NSE) and nmap service identification were also used to obtain additional information about in-scope hosts:

*Figure 2 -- Nmap scanning results*

*Figure 3 -- Nmap scanning results*

*Figure 4 -- Nmap scanning results*

### 2.2.2 Network Mapping

The network mapping phase involved actively probing the designated target systems. The information obtained provided Pentester consultants with an understanding of the listening services and operating systems. Pentester used multiple Internet protocols to gather information about the target host or network. The knowledge derived from the network mapping phase was essential for an efficient vulnerability testing stage.

|  |  |
| --- | --- |
| Ping | Used to test simple system response and for the implementation of filtering. |
| Nmap | Scanning tool, which can detect listening services and operating systems. |
| Telnet, Netcat | Used to interact with services or obtain relevant information about them. |
| culr | Used for transferring data using various protocols. |
| Metasploit discovery modules | Used to identify other services |

### 2.2.3 Vulnerability testing

The objective of this phase was to identify hosts, services, and vulnerabilities in the target environment using a combination of open-source and commercial security tools. During this phase, Pentester performed host, service and vulnerability identification. The vulnerability testing phase was more intrusive than the previous phase and may have been picked up by any intrusion detection or monitoring systems located on the client network.

|  |  |
| --- | --- |
| Burp Suite Professional | Used to assess web application vulnerabilities. |
| OpenVAS | Used to assess the level of vulnerability within the system. |
| nikto | Used to assess the level of vulnerability within the system. |
| testssl.sh | Used to assess the level of vulnerability within the system |
| retire | Used to assess the level of vulnerability within the system |
| Sqlmap | Used to assess databases vulnerabilities. |

### 2.2.4 Manual verification

Pentester used manual techniques to confirm the results from the automated tools thus eliminating any false positives. As an addition to this, Pentester used manual testing techniques to identify obscure vulnerabilities. Manual verification offers significant value over the sole use of automated tools. Often, these advanced techniques can be used to determine that vulnerabilities identified through automated tools are false positives. Furthermore, this technique would usually allow Pentester to find services listening on obscure or high ports.

|  |  |
| --- | --- |
| Browser | Used to test HTTP and HTTPS connections. |
| Telnet, Netcat, Nmap | Used to interact with services or obtain relevant information about them. |
| Burp Suite Professional | Used to attempt to exploit web application vulnerability. |
| curl | Used for transferring data using various protocols. |
| sublist3r.py | Used for enumerating subdomains |
| dnsrecon | Used for brute-forcing subdomains |
| Wget | Used to download entire webpage |

### 2.2.5 Vulnerability Exploitation

Pentester seek to exploit the vulnerabilities identified. Pentester execute exploits with the sole aim of fulfilling the specific goals of the penetration test; however, Pentester do not actively exploit any vulnerability without obtaining permission from the customer.

Exploitation of certain vulnerabilities may have led to the identification of additional vulnerabilities that, in turn, may have required further exploitation to identify potential problems. However, please note that Pentester follow this iterative process only to the extent necessary to accomplish the goals of the assessment.

|  |  |
| --- | --- |
| Firefox | Internet web browser that provides additional security add-ons. |
| Various clients | Used to connect and test services that have been mapped. |
| Burp Suite | Used to attempt to exploit web application vulnerability. |
| Metasploit Community Edition v5.0 with latest update | Renowned exploitation framework which also has many auxiliary modules used for manual testing. |
| Sqlmap | Used to attempt to exploit databases vulnerability. |

# 3. Findings and recommendations key

Wherever possible, Pentester rates each finding in this document according to its business impact and each recommendation in terms of the effort required in correcting the problem. The following table describes the different rating levels.

|  |  |
| --- | --- |
| **Finding Description** | **This column provides a brief technical description of the finding in question. More detailed information or issue-related screenshots will typically be provided in a subsequent section or appendix, if necessary.** |
| **Affected Systems** | This column lists the IP Address, hostname or a description of the vulnerable system. |
| **Overall Risk Level** | This section indicates the overall risk to a system that a given finding implies. This is typically a subjective analysis of the exploit difficulty in conjunction with the exploit impact. A rating of high, medium, or low will be suggested as follows:  **Critical** - Critical vulnerabilities pose a serious threat to an organization's security, and should be fixed immediately. They may provide a total compromise of the target environment, or similar critical impacts.  **High** – The system is susceptible to a high level of risk. The issue should be addressed as quickly as possible.  **Medium** – The system is susceptible to significant level of risk. The issue should be incorporated into the system development life-cycle and addressed in due time.  **Low** – The system is mildly susceptible to exploit. The issue should be addressed based on resource and business impact considerations.  **Info** - Informational vulnerabilities have little-or-no impact to the target scope by themselves. They are included however, as they may be a risk when combined with other circumstances or technologies not currently in place. Remediation of informational items is not necessary. |
| **CVSS Score** | The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities. CVSS attempts to assign severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat. Scores are calculated based on a formula that depends on several metrics that approximate ease of exploit and the impact of exploit. Scores range from 0 to 10, with 10 being the most severe. |
| **Remediation** | This column provides a brief general or technical description of the suggested remediation path. This may include links to bug fixes or patch information. Other references or brief descriptions of typical remediation approaches are also included. |
| **Remediation status** | This column is added in case of retest or recheck, it provides a brief view on improvement implemented after initial testing:  Fixed – The vulnerability was fully removed.  Mitigated – The vulnerability is not fixed but no more exploitable (e.g. workaround).  Partially fixed – The vulnerability is only partially fixed and remains fully or partially exploitable.  Remaining – No visible improvement. |

# 4.0 Network penetration test

Pentester performed external and internal penetrations of the scope provided by Customer. The testing involved automated scanning, manual verification and careful analysis of the vulnerabilities found.

## 4.1 Scope

The following hosts within the scope of the testing:

|  |  |  |
| --- | --- | --- |
| **#** | **Host/Network** | **Internal/External** |
| 1. | 0.0.0.0/24 | Ext. |
| 2. |  | Int. |
| 3. |  | Ext. |
| 4. |  | Ext. |
| 5. |  | Ext. |
| 6. |  | Ext. |
| 7. |  | Ext. |
| 8. |  | Ext. |
| 9. |  | Ext. |
| 10 |  | Ext. |

## 4.2 Findings breakdown

## 4.3 Findings Summary

The table below contains summary of audit findings.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Risk Level | Finding | Affected system | Recommendation |
| 1 | **HIGH** | Broken access control – User is able to perform admin related actions | [https://astanait.edu.kz](https://arx.com.ua/) | Use an access control matrix to define the access control rules. Configure file access permissions according to access control rules. |
| 2 | **HIGH** | Reflected XSS vulnerability | https://murat.one | Implement XSS protection, e.g. filter user-supplied input and escape it when printing out, enforce strong stylization and whitelisting for parameters where this is possible. |
| 3 | **MEDIUM** | Bruteforcible login form | [https://astanait.edu.kz](https://arx.com.ua/) | The endpoint should be protected from bruteforcing by rate limiting, CAPTCHA, and/or blocking originating IP after several failed attempts. |
| 4 | **MEDIUM** | Outdated PHP/7.3.6 version with known vulnerabilities is used | [https://astanait.edu.kz](https://arx.com.ua/) | PHP should be up to date and should not contain any known vulnerabilities. |
| 5 | **MEDIUM** | Partner registration id sent via URL in GET request | [https://astanait.edu.kz](https://arx.com.ua/) | The transmission of sensitive partner personal identifiable information should happen via HTTP PUT/POST/ requests. |
| 6 | **MEDIUM** | Malicious redirect via host header injection | [https://astanait.edu.kz](https://arx.com.ua/) | Host header should not be trusted. |
| 7 | **MEDIUM** | Insecure opening of external links allows for reverse tabnabbing | [https://astanait.edu.kz](https://arx.com.ua/) | A standard should be applied for all possible link that could be created by user. If the link has an attribute "target=\_blank" then it should be followed with "rel=noopener noreferrer" |
| 8 | **MEDIUM** | Cookies without “Secure” or “HttpOnly” flag set | [https://astanait.edu.kz](https://arx.com.ua/) | Use “Secure” and “HttpOnly”cookie flags. |
| 9 | **MEDIUM** | Missing/misconfigured security headers | [https://astanait.edu.kz](https://arx.com.ua/) | All the recommended security headers should be implemented |
| 10 | **MEDIUM** | Phone number can be sent in automated manner | [https://astanait.edu.kz](https://arx.com.ua/) | CAPTCHA protection, timeout protection and rate limiting should be implemented |
| 11 | **LOW** | Unencrypted \_\_VIEWSTATE parameter | [https://astanait.edu.kz](https://arx.com.ua/) | To fix this issue open Web.Config and add the following line under the <system.web> element: <machineKey validation="AES"/> |
| 12 | **LOW** | Improper error handling | [https://astanait.edu.kz](https://arx.com.ua/) | No detailed exception stack trace should be presented to a user |
| 13 | **LOW** | Sensitive data exposure in server response headers | [https://astanait.edu.kz](https://arx.com.ua/) | Server and software information should be removed from HTTP responses. |
| 14 | **LOW** | Web Configuration File is Exposed | [https://astanait.edu.kz](https://arx.com.ua/) | web.config is available for anonymous users , note that currently there is no username and passwords or credentials on it but that might be changed in future and it's best practice to not be allowed for normal users |
| 15 | **LOW** | Weak SSL configuration | [https://astanait.edu.kz](https://arx.com.ua/) | Use TLS 1.2 (with approved cipher suites) or higher. Only support strong cryptographic ciphers. |

# 5.0 Further information

## 5.1 Critical / High severity findings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **HIGH** | | | **20180113-Client-001** | **[Systemic] Broken access control – User is able to perform admin related actions** |
| **Location** | | | [https://astanait.edu.kz](https://arx.com.ua/) | | |
| **Description** | | | It was found that it is possible for a sales user to create new users, view other users list, view other groups, view organizations, view vehicle list etc.  Access control, sometimes called authorization, is how a web application grants access to content and functions to some users and not others. These checks are performed after authentication, and govern what ‘authorized’ users are allowed to do. Access control sounds like a simple problem but is insidiously difficult to implement correctly. A web application’s access control model is closely tied to the content and functions that the site provides. In addition, the users may fall into a number of groups or roles with different abilities or privileges.  Developers frequently underestimate the difficulty of implementing a reliable access control mechanism. Many of these schemes were not deliberately designed, but have simply evolved along with the web site. In these cases, access control rules are inserted in various locations all over the code. As the site nears deployment, the ad hoc collection of rules becomes so unwieldy that it is almost impossible to understand. | | |
| **Evidence** | | | | | |
| All the requests below were performed using direct links from admin interface. **privileges**  Figure 5 – Screenshot of the new user creation request and response using sales user session  Figure 6 – Screenshot of the recently created user from admin interface  Figure 7 – Screenshot of changing the other user’s profiles using sales user account | | | | | |
| **Solution(s), Advice & Recommendation(s)** | | | | | |
| Use an access control matrix to define the access control rules. Without documenting the security policy, there is no definition of what it means to be secure for that site. The policy should document what types of users can access the system, and what functions and content each of these types of users should be allowed to access. The access control mechanism should be extensively tested to be sure that there is no way to bypass it. This testing requires a variety of accounts and extensive attempts to access unauthorized content or functions. Configure data access permissions according to access control rules.  LINKS:   * https://www.owasp.org/index.php/Broken\_Access\_Control | | | | | |
| **CVSS Score** | | 7.1 | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **HIGH** | | | **20180113-Client-002** | **Reflected XSS vulnerability** |
| **Location** | | | [https://astanait.edu.kz](https://arx.com.ua/) | | |
| **Description** | | | Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users. The services on the Customer’s host have multiple Reflected XSS vulnerabilities.  Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other web site. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user’s browser.  Technical Impact: Attackers can execute scripts in a victim’s browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user’s browser using malware, etc.  Business Impact: Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability. | | |
| **Evidence** | | | | | |
| Summary of xss vulnerability is contained in the following table:   |  |  |  |  | | --- | --- | --- | --- | | Request method | Host | Payload | XSS type | | POST | [https://astanait.edu.kz](https://arx.com.ua/) | <script>alert(1)</script> | Reflected | | POST | [https://astanait.edu.kz](https://arx.com.ua/) | <script>alert(1)</script> | Reflected |   Examples of screenshots of Burp Suite tool with injected script code for reflected XSS with the application’s response is provided below.  Figure 8 - Screenshot with popup alert window as example of reflected XSS  Figure 9 - Screenshot of the vulnerable request  Figure 10 - Screenshot with popup alert window as example of reflected XSS  Figure 11 - Screenshot of the vulnerable request | | | | | |
| **Solution(s), Advice & Recommendation(s)** | | | | | |
| Implement XSS protection, e.g. filter user-supplied input and escape it when printing out, enforce strong stylization and whitelisting for parameters where this is possible. Follow the next rules:   * Never insert untrusted input * Perform sanitation of input data before inserting it into the page content * Use native API and additional software whenever possible * Always use preset character encoding of the displayed page * Escape JSON values in an HTML context and read the data with JSON.parse. Ensure returned Content-Type header is application/json and not text/html. This shall instruct the browser not misunderstand the context and execute injected script. Also, x-content-type-options header should return no sniff that is used to disable content-sniffing on old versions of Internet Explorer. * Use HTTPOnly cookie flag * Implement Content Security Policy * Use an Auto-Escaping Template System * Use the X-XSS-Protection Response Header   LINKS:   * https://www.owasp.org/index.php/Cross-site\_Scripting\_(XSS) * https://www.owasp.org/index.php/XSS\_(Cross\_Site\_Scripting)\_Prevention\_Cheat\_Sheet | | | | | |
| **CVSS Score** | | 7.1 | | | |

## 5.2 Medium severity findings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MEDIUM** | | **20180113-Client-003** | **Bruteforcible login form** |
| **Location** | | [https://astanait.edu.kz](https://arx.com.ua/) | | |
| **Description** | | Since the application is not protected from bruteforcing by rate limiting or CAPTCHA, it is possible to perform a bruteforce attack, which attempts to guess the correct username and password in automated way using wordlist of common credentials. In this case, pentester used a wordlist containing 825000 entries. | | |
| **Evidence** | | | | |
| Sample evidence follows:  Figure 12 - Screenshot of performing a bruteforce attack using a common usernames and passwords wordlist with 825000 entries  Figure 13 - Screenshot of affected request and response | | | | |
| **Solution(s), Advice & Recommendation(s)** | | | | |
| The endpoint should be protected from bruteforcing by rate limiting, CAPTCHA, and/or blocking originating IP after several failed attempts.  LINKS:   * <https://www.owasp.org/index.php/Testing_for_Captcha_(OWASP-AT-008)> * <https://owasp.org/www-community/attacks/Brute_force_attack> | | | | |
| **CVSS Score** | | 6.5 | | |

## 5.3 Low severity findings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **LOW RISK** | | **20180113-Client-011** | **Unencrypted \_\_VIEWSTATE parameter** |
| **Location** | | [https://astanait.edu.kz](https://arx.com.ua/) | | |
| **Description** | | The \_\_VIEWSTATE parameter is not encrypted. To reduce the chance of someone intercepting the information stored in the ViewState, it is good design to encrypt the ViewState. To do this, set the machineKey validation type to AES. This instructs ASP.NET to encrypt the ViewState value using the Advanced Encryption Standard.  This vulnerability affects POST /activate/ HTTP/1.1 request | | |
| **Evidence** | | | | |
| Sample evidence follows:  Figure 14 – Screenshot with Burp tool that decodes the \_\_VIEWSTATE parameter value  Figure 15– Screenshot with Burp tool that shows the affected request | | | | |
| **Solution(s), Advice & Recommendation(s)** | | | | |
| Note that currently there is no username and passwords or credentials on it but that might be changed in future. To fix this issue open Web.Config and add the following line under the <system.web> element: <machineKey validation="AES"/>  LINKS:   * <https://www.valencynetworks.com/kb/how-to-encrypt-viewstates-in-dotnet.html> | | | | |
| **CVSS Score** | | 2.6 | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **LOW RISK** | | **20180113-Client-012** | **Improper error handling** |
| **Location** | | [https://astanait.edu.kz](https://arx.com.ua/) | | |
| **Description** | | Improper handling of errors can introduce a variety of security problems for a web site. The most common problem is when detailed internal error messages such as stack traces, database dumps, and error codes are displayed to the user (hacker). These messages reveal implementation details that should never be revealed. Such details can provide hackers important clues on potential flaws in the site and such messages are also disturbing to normal users. | | |
| **Evidence** | | | | |
| Sample evidence follows:  Figure 16– Screenshot of the java exception stacktrace  Figure 17– Screenshot with Burp tool that shows the affected request  Figure 18– Screenshot of the ArgumentNullException  Figure 19– Screenshot of the exception stacktrace | | | | |
| **Solution(s), Advice & Recommendation(s)** | | | | |
| No detailed exception stack trace should be presented to a user. A specific policy for how to handle errors should be documented, including the types of errors to be handled and for each, what information is going to be reported back to the user, and what information is going to be logged. All developers need to understand the policy and ensure that their code follows it.  LINKS:   * <https://owasp.org/www-community/Improper_Error_Handling> | | | | |
| **CVSS Score** | | 3.3 | | |

## 5.4 Informational severity findings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **INFO** | | | **20191004-Client-010** | **Password field with autocomplete enabled** |
| **Location** | | | https://admin | | |
| **Description** | | | Most browsers have a tool to remember user credentials that are entered into HTML forms. This function can be configured by the user and also by applications that employ user credentials. If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application. The stored credentials can be captured by an attacker who gains control over the user's computer. Further, an attacker who finds separate application vulnerability such as cross-site scripting may be able to exploit this to retrieve a user's browser-stored credentials.  If the autocomplete is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application.  The stored credentials can be captured by an attacker who gains control over the user's computer. Further, an attacker who finds separate application vulnerability such as cross-site scripting may be able to exploit this to retrieve a user's browser-stored credentials. | | |
| **Evidence** | | | | | |
| Examples of application login form with possibility of autocomplete:  Figure 20– Screenshot of application login form with possibility of password autocompleting | | | | | |
| **Solution(s), Advice & Recommendation(s)** | | | | | |
| * To prevent browsers from storing credentials entered into HTML forms, include the attribute autocomplete="off" within the FORM tag (to protect all form fields) or within the relevant INPUT tags (to protect specific individual fields); * In some cases, the browser will keep suggesting autocompletion values **even if the autocomplete attribute is set to off**. This unexpected behavior can be quite puzzling for developers. The trick to really forcing the no-autocompletion is to assign a random string to the attribute, for example: autocomplete="nope"; * If an author would like to prevent the autofilling of password fields in user management pages where a user can specify a new password for someone other than themself, autocomplete="new-password" should be specified, though support for this has not been implemented in all browsers yet; * Also, you can use jQuery disableAutoFill plugin (https://github.com/terrylinooo/jquery.disableAutoFill).   LINKS:   * https://developer.mozilla.org/en-US/docs/Web/Security/Securing\_your\_site/Turning\_off\_form\_autocompletion | | | | | |
| **CVSS Score** | | N/A | | | |

# 6.0 Identified Services and Open Network Ports

During the reconnaissance phase of the penetration testing exercise, penetration testing team performed:

* Network sweeps to identify / verify the IP address(es) of the target systems
* Network traces to discern the topology of the target network and draw a network map
* Port scanning to identify any potential openings within target machine(s) by identifying any listening ports and services

|  |  |  |
| --- | --- | --- |
| **IP address** | **Internal / External** | **Function / Purpose of the port or service** |
| 0.0.0.0 | Ext | 139/tcp filtered netbios-ssn  443/tcp open ssl/http Cisco ASA SSL VPN |
|  | Ext | 139/tcp filtered netbios-ssn  443/tcp open ssl/http Cisco ASA SSL VPN |

# 7.0 OWASP Top Ten 2021

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications that can be trusted. The goal of the Top 10 project is to raise awareness about application security by identifying some of the most critical risks facing organizations.

Note that the following table indicates the results of the original test only and does not display conclusions of any followed re-checks.

|  |  |  |
| --- | --- | --- |
| **Description** | **Status** | **Finding Index** |
| **A01:2021-Broken Access Control** | **Pass** |  |
| **A02:2021-Cryptographic Failures** | **Pass** |  |
| **A03:2021-Injection** | **Fail** | **20180113-Client-002** |
| **A04:2021-Insecure Design** | **Pass** |  |
| **A05:2021-Security Misconfiguration** | **Fail** | **20180113-Client-007 20180113-Client-008** |
| **A06:2021-Vulnerable and Outdated Components** | **Fail** | **20180113-Client-005 20180113-Client-013** |
| **A07:2021-Identification and Authentication Failures** | **Fail** | **20180113-Client-001** |
| **A08:2021-Software and Data Integrity Failures** | **Pass** |  |
| **A09:2021-Security Logging and Monitoring Failures** | **Pass** |  |
| **A10:2021-Server-Side Request Forgery** | **Pass** |  |

# Appendix A – Pentesting process

This section contains screenshots examples of pentesting process.

Figure 21– Testing using testssl tool

Figure 22– Testing for access control flaws using Authorize tool

Figure 23– Testing process with nmap

Figure 24– Fuzzing directories

Figure 25– Generating CSRF poc

Figure 26– Automated scanning using Burp Suite

Figure 27– Spidering the domain

Figure 28 – Investigating directory structure

Figure 29– Testing for XSS vulnerability

Figure 30– Checking the local storage for sensitive information

# Appendix B – Reference documents

Find below list of documents related to this service.

|  |  |
| --- | --- |
| **Document name** | **Description** |
| **CLIENT\_Vulnerability\_Regsitry\_01.01.2022.xlsx** | Extended CLIENT Vulnerability registry table |