SYSLIFTERS
Aron Molnar | aron@syslifters.com | +436609234060

Pentest Report

Sample Report: Web

Pentest for **Security Maximale GmbH**
August 29, 2022

Version 1.0

Inhalt

Vulnerability Overview

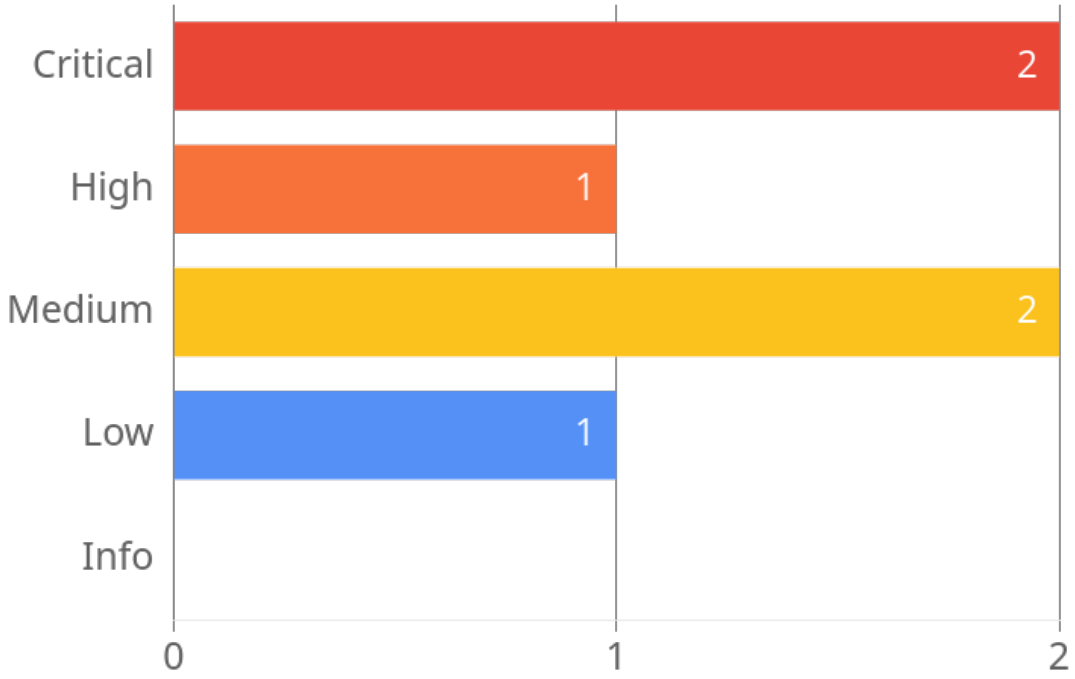

Figure 1 - Distribution of identified vulnerabilities

Finding	Severity
Takeover of user accounts by password reset	**Critical**
Stored Cross-Site Scripting (XSS)	**Critical**
Disclosure of passwords via Path Traversal	**High**
Forwarding users through Open Redirect	**Medium**
Disclosure of internal information	**Medium**
Weaknesses in session management	**Low**

Management Summary

In the course of the security audit, we were able to take over administrative accounts and customer accounts of the online store. This allowed us to view and edit customer data, including payment information and customer orders.

This was achieved in two ways: On the one hand, the function for resetting passwords was not sufficiently secure and was also available to the store's customers. This allowed customers to reset the passwords of administrators. On the other hand, malicious JavaScript functions could be infiltrated via the customers' status messages, allowing administrative user sessions to be taken over.

In addition, any files could be read from the web server using the function for selecting the language to be delivered. These included configuration files with passwords.

Other, less critical vulnerabilities, such as the redirection of external users to untrustworthy websites, the disclosure of internal information and weaknesses in session management should be addressed in a continuous improvement process.

Here is the report. What now?

It is very important to us that you work with our report and derive measures for improvement from it. Therefore, we will re-test vulnerabilities for you free of charge if they are resolved within eight weeks!

In this assessment we have identified vulnerabilities with criticality **Critical** and **High**. We recommend that these vulnerabilities be addressed as a matter of priority.

Vulnerabilities with less complex countermeasures and risk **Medium** and below should, according to our recommendation, be fixed prioritised by effort. All other vulnerabilities should either be fixed or addressed as part of a continious improvement process.

Please ensure that you deprovision all users and resources that were provisioned during the pentest as soon as they are no longer required.

Scope and duration

The scope of the pentest included the online shop system of Security Maximale GmbH at https://www.example.com.

The penetration test was carried out according to a time-box approach and covered 10 person days.

Short description

1. Takeover of user accounts by password reset (Critical: 9.9)

Affects: http://www.example.com/api/v2/users/change-password/admin
In the course of the audit, we as a customer of the online store were able to change the passwords of other customers and store administrators to any passwords we wanted. This allowed us to take over administrative user accounts and thus compromise the entire store system.

2. Stored Cross-Site Scripting (XSS) (Critical: 9.3)

Affects: http://www.example.com/api/v2/users/status
At the time of testing, the web application saved unvalidated user input and later included it into HTTP responses insecurely. It was therefore vulnerable to stored cross-site scripting attacks (XSS). This could allow attackers to compromise the user accounts of other users, including administrators.

3. Disclosure of passwords via Path Traversal (High: 7.5)

Affects: http://www.example.com/?lang=de
Users of the application were able to gain access to passwords in configuration files at the time of the test. This was achieved via a path traversal attack because the web application did not sufficiently check HTTP parameters.

4. Forwarding users through Open Redirect (Medium: 6.1)

An open redirect vulnerability allowed users to be redirected to other websites at the time of the test. To do this, users had to click on a manipulated link. The link appears trustworthy as it points to the web application, which immediately redirects the user to a third-party website.

5. Disclosure of internal information (Medium: 5.3)

Affects: http://www.example.com/.env
The web application revealed internal information that could potentially be used for further attacks. Disclosure of information, also known as information disclosure, occurs when a system unintentionally passes on internal information to its users.

6. Weaknesses in session management (Low: 3.6)

We were able to identify weaknesses in the web application's session management. The users' sessions could be used without time restrictions and therefore did not require re-authentication at any time. Persons with access to a computer system could exploit this situation if another user had not explicitly logged out of the application beforehand.

Vulnerability details

1. Takeover of user accounts by password reset

Criticality: Critical
CVSS-Score: 9.9 | CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
Affects: http://www.example.com/api/v2/users/change-password/admin

Overview

In the course of the audit, we as a customer of the online store were able to change the passwords of other customers and store administrators to any passwords we wanted. This allowed us to take over administrative user accounts and thus compromise the entire store system.

Description

Administrators of the online store should be able to reset customers' passwords. In the course of the audit, we found that this function was not restricted to administrators, but was available to all users, including registered customers.

The API endpoint `POST /api/v2/users/change-password/{username}` overwrites the password of the specified user.

Figure 2 - Overwrite user passwords via Change-Password API

The server responded with the status code 200 "OK" and changed the password of the user account "admin".

Recommendation

Customers and non-administrative users of the online store should not be allowed to use the function for changing passwords.

2. Stored Cross-Site Scripting (XSS)

Criticality: Critical
CVSS-Score: 9.3 | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:N
Affects: http://www.example.com/api/v2/users/status

Overview

At the time of testing, the web application saved unvalidated user input and later included it into HTTP responses insecurely. It was therefore vulnerable to stored cross-site scripting attacks (XSS). This could allow attackers to compromise the user accounts of other users, including administrators.

Description

During the pentest, we were able to identify a stored XSS vulnerability in the web application. Due to incorrect validation and encoding of data, we were able to infiltrate malicious scripts as a customer of the online store and store them persistently there.

Users can set a status message in the application.

Figure 3 - Specifying the status message

When clicking on the corresponding user profile and in the user overview in the backend, the status message is embedded in the HTML document. If the message is JavaScript code, it is then executed in the browser of another user.

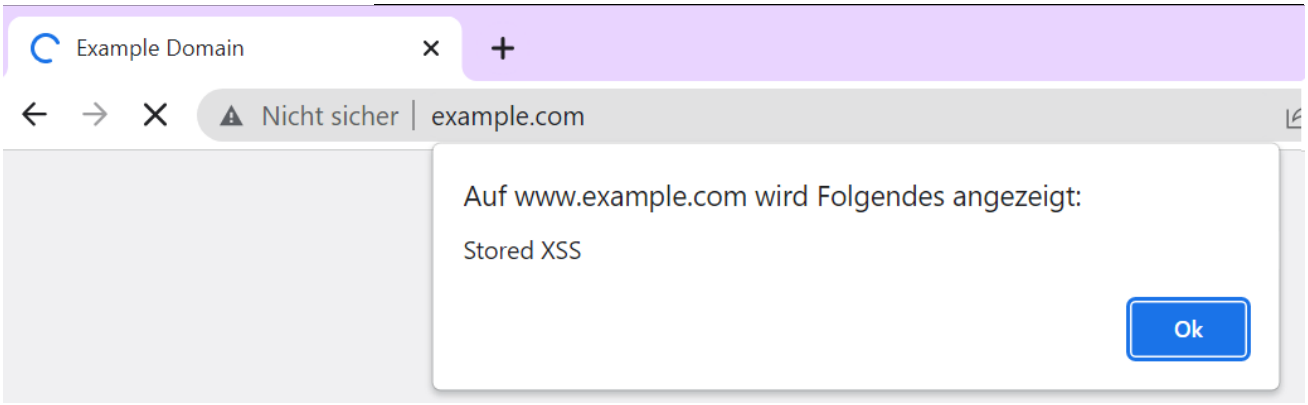

Figure 4 - Execution of JavaScript code in the backend

This could compromise the user accounts of other users, including administrators.

Cross-site scripting (XSS) is a common web security vulnerability where malicious scripts can be injected into web applications due to insufficient validation or coding of data. In XSS attacks, attackers embed JavaScript code into the content delivered by the vulnerable web application.

The aim of stored XSS attacks is to place script code on pages that are visited by other users. Simply calling up the affected subpage is sufficient for the script code to be executed in the victim's web browser.

For an attack, malicious scripts are injected into the web application by the attacker, stored and integrated into subsequent HTTP responses of the application. The malicious script is ultimately executed in the victim's web browser and can potentially access cookies, session tokens or other sensitive information.

In a successful attack, an attacker gains control over functions and data of the web application in the victim's context. If the affected user has privileged access, an attacker may be able to gain complete control over the web application.

Recommendation

- It should be ensured that all processed data is filtered as strictly as possible. It should be filtered and validated based on expected and valid inputs.
- Data should be encoded before the web application includes it in HTTP responses. Encoding should be context-sensitive, i.e. depending on where the web application inserts data in the HTML document, the appropriate encoding syntax must be taken into account.
- The HTTP headers `Content-Type` (e.g. `text/plain`) and `X-Content-Type-Options: nosniff` can be set for HTTP responses that do not contain HTML or JavaScript.
- We recommend that you also use a Content Security Policy (CSP) to control which client-side scripts are allowed and which are prohibited.
- Detailed information and assistance on preventing XSS can be found in the linked Cross-Site Scripting Prevention Cheat Sheet from OWASP.

3. Disclosure of passwords via Path Traversal

Criticality: High
CVSS-Score: 7.5 | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Affects: http://www.example.com/?lang=de

Overview

Users of the application were able to gain access to passwords in configuration files at the time of the test. This was achieved via a path traversal attack because the web application did not sufficiently check HTTP parameters.

Description

During the audit, we were able to identify a path traversal vulnerability in the web application. This allowed us to access sensitive files on the server system.

The 'lang' parameter, which allows users to select an individual language, was the name of a folder in which language information was stored and dynamically accessed.

By manipulating the parameter, we were able to read arbitrary files on the web server, including configuration files with passwords.

Figure 5 - Reading the database password of the web application

Path traversal is a web security vulnerability that allows an attacker to access files and directories of the underlying web server of a web application. In a path traversal attack, access to files and directories is restricted solely by the existing access controls of the underlying operating system.

Most web servers restrict access to a specific part of the file system, which is usually referred to as the "web root". This directory contains all files that are required for the functionality of the web application. To break out of the web root folder using a path traversal attack, attackers use special character strings in path specifications.

In its simplest form, an attacker uses the special character string "../" to change the location of the resource requested in the parameter. This character string is a relative path specification.

It refers specifically to the parent directory of the current working directory. Attackers often use alternative encodings of the "../" sequence to bypass any existing security filters. These methods include valid and invalid Unicode-encoded ("..%u2216" or "..%c0%af"), URL-encoded ("%2e%2e%2f") and double URL-encoded characters ("..%255c") of the backslash character. Advanced techniques often use additional special characters such as the dot "." to refer to the current working directory or the "%00" NULL character to bypass rudimentary end-of-file checks.

In a successful attack, an attacker can use path traversal to access any files and directories on the vulnerable system. This can include sensitive operating system files, application code or configuration files. In certain cases, it may also be possible to gain write access to files and directories through path traversal. Under these circumstances, it is possible for an attacker to gain code execution and thus complete control over the web server.

Recommendation

- The most effective way to avoid path traversal is to avoid user-defined file name and path specifications in the web application. If this is not possible, use indices instead of specific values (e.g. index: 5 corresponds to the language setting "German").
- Always validates all user input. Ensures that only valid input that is expected for the application is accepted. Potentially malicious input should be discarded.
- When normalizing path specifications, bear in mind that characters could be single or multiple encoded (e.g. URL-encoded, e.g. `%20` or `%2520` instead of a space).

4. Forwarding users through Open Redirect

Criticality: Medium
CVSS-Score: 6.1 | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

Overview

An open redirect vulnerability allowed users to be redirected to other websites at the time of the test. To do this, users had to click on a manipulated link. The link appears trustworthy as it points to the web application, which immediately redirects the user to a third-party website.

Description

For the login functionality to the online store, the `redirectURI` parameter can be used to specify which page users should be redirected to after logging in. However, any URI can be specified here.

Attackers could use this open redirect vulnerability to redirect website users to untrusted websites.

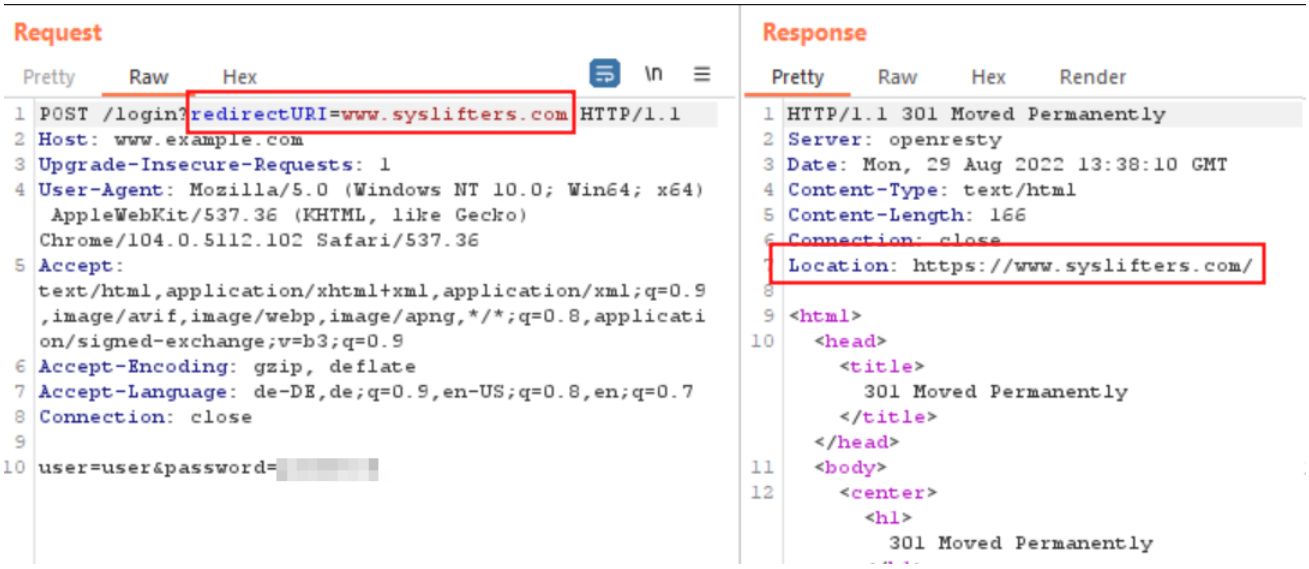

Figure 6 - Forwarding the user to www.syslifters.com

This vulnerability could be abused in the course of phishing attacks, for example.

An open redirect is a web security vulnerability that occurs when an application embeds user input into the destination of a redirect in an insecure way. A redirect occurs when a web application changes the URL that the client is currently accessing. The backend has various options for carrying out redirects, e.g. via corresponding HTTP headers (status codes 30x) or via JavaScript. An application is vulnerable to open redirects if an attacker is able to construct a URL within the application that causes a redirect to any external domain.

An open redirect vulnerability allows an attacker to use an authentic application URL in an attack that references the legitimate domain and has a valid SSL certificate. This behavior

lends more credibility to a phishing attack, making it easier to carry out social engineering attacks against users. Open redirects enable many other attacks based on them.

Recommendation

- Avoid redirects and forwarding in the web application if possible. If you need the functionality, do not allow user-defined URLs.
- If the destination of a redirect must be determined by a user, use indexes instead of full URLs (e.g. a short name, ID or token).
- Forces all redirects to external resources to first go through a page that informs the user that they are leaving your website. This page clearly indicates the destination and the user must click on a link to confirm.
- Validates all user input and ensures that only valid input expected for the application is accepted. Potentially malicious input should be discarded.
- If possible, do not accept complete URLs. URLs are difficult to validate and the URL parser could be abused depending on the technology used. If this is still necessary, make sure that only valid IP addresses or domain names are accepted.
- Uses the Allow-Listing approach if the web application only redirects to identified and trusted URLs. With this approach, a list of permitted IP addresses and domains is defined at application level. After validation of the entries and comparison with the list, the web application may only redirect to those resources that are also contained in the list. Redirects to resources outside the list are blocked with this approach.
- Detailed information and assistance on how to prevent open redirects can be found in the linked "Unvalidated Redirects and Forwards Cheat Sheet" from OWASP.

5. Disclosure of internal information

Criticality: Medium
CVSS-Score: 5.3 | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Affects: http://www.example.com/.env

Overview

The web application revealed internal information that could potentially be used for further attacks. Disclosure of information, also known as information disclosure, occurs when a system unintentionally passes on internal information to its users.

Description

During the tests, we were able to download a configuration file with internal information. This information could be used for follow-up attacks.

The `.env` file was readable in the web root of the web application. This contained information such as the application's Redis server or email gateway.

Figure 7 - Access to the file `.env`

The disclosure of this information has little impact on the security of the application. However, it could represent important key information for targeted follow-up attacks by an attacker.

Recommendation

- Configuration files should not be publicly accessible.
- The `.env` file should be moved to a location outside the web root.

6. Weaknesses in session management

Criticality: Low
CVSS-Score: 3.6 | CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:N

Overview

We were able to identify weaknesses in the web application's session management. The users' sessions could be used without time restrictions and therefore did not require re-authentication at any time. Persons with access to a computer system could exploit this situation if another user had not explicitly logged out of the application beforehand.

Description

We found that user sessions could be used without time restrictions. This could allow attackers to take over user sessions that were not explicitly logged off beforehand.

This could be possible, for example, by allowing a third party to operate a user's computer in which a session is still active. In addition, it could be possible for attackers to reuse session tokens if they become known (e.g. via log files; locally or on proxy servers, etc.).

Recommendation

- User sessions in web applications should expire automatically after a certain period of inactivity.
- Depending on the criticality of the user authorization and the application, the timeout could be between one hour and one day.

List of Changes

Version	Date	Description	Author
0.9	2022-08-26	Draft	Aron Molnar
1.0	2022-08-29	Review and Finalisation	Christoph Mahrl

Disclaimer

We cannot guarantee that all existing vulnerabilities and security risks have actually been discovered. This is due to limited time resources and limited knowledge of the pentester about the IT infrastructure, software, source code, users, etc. Extensive collaboration between the client and penetration testers increases the efficiency of the penetration test. This includes, for example, the disclosure of details of internal systems or the provisioning of test users.

This penetration test represents a snapshot at the time of testing. No future security risks can be derived from it.

Imprint

syslifters.com | **Dedicated to Pentests**
Syslifters GmbH | Eitzersthal 75 | 2013 Göllersdorf
FN 578505 v | District Court Hollabrunn