ATTACK SURFACE

flexible project management for attackers

Negin Nourbakhsh

Fazel Mohammad Ali Pour

31 Aug 2023

HADESS

WWW.HADESS.IO

Executive Summary

XSS to Account Takeover with Read CSRF-Token in Body:

- **Vulnerability Type:** Cross-Site Scripting (XSS)
- **Attack Vector:** Injecting malicious scripts via XSS, coupled with reading a **csrf-token** from the **<body>** tag.
- **Description:** An attacker exploits an XSS vulnerability to inject malicious scripts into user-generated content. By gaining the ability to read a **csrf-token** from the page's source code, the attacker can impersonate users and perform unauthorized actions.
- **Impact:** The attacker can bypass security measures using the **csrf-token** and execute actions on behalf of victims. This could lead to account takeover, data manipulation, and unauthorized access to sensitive functionality.

CSV Injection Leading to OS Command Execution:

- **Vulnerability Type:** CSV Injection
- **Attack Vector:** Manipulating CSV files to inject malicious formulas leading to OS command execution.
- **Description:** Attackers exploit CSV injection vulnerabilities by crafting malicious data within cells of a CSV file. When the file is opened by a spreadsheet application, the payload is interpreted as formulas, potentially leading to the execution of arbitrary commands on the host system.
- **Impact:** Successful exploitation can lead to unauthorized system access, data breaches, and even complete compromise of the host system.

01

Advisory

Abstract

Redmine is an open-source project management and issue tracking web application. It is designed to help teams and organizations manage projects, track issues, collaborate on tasks, and maintain a structured workflow. Redmine provides a variety of features, including:

1. **Issue Tracking:** Redmine allows users to create, manage, and track issues, bugs, tasks, and feature requests. Each issue can be assigned to specific users, given priorities, categorized, and tracked through its lifecycle.
2. **Project Management:** It offers tools for project planning, task scheduling, and progress tracking. Users can define milestones, create project roadmaps, and manage project-related documents.
3. **Collaboration:** Redmine enables team collaboration by providing features such as file attachments, discussions, comments, and notifications. This facilitates communication among team members working on the same project.
4. **Customization:** The application can be customized to match the specific needs of an organization. Custom fields, workflows, and issue statuses can be defined to align with the organization's processes.
5. **Time Tracking:** Redmine allows users to log time spent on various tasks and issues, which can be useful for tracking project progress and evaluating resource allocation.
6. **Integration:** Redmine supports integration with version control systems like Git and Subversion, allowing teams to link code changes to specific issues. It also supports third-party plugins to extend its functionality.
7. **User Roles and Permissions:** Different user roles can be defined, such as administrators, project managers, developers, and clients, each with varying levels of access and permissions.
8. **Reporting:** Redmine provides reporting capabilities that allow users to generate various types of reports, including issue lists, activity summaries, and progress charts.

Redmine is often used by software development teams, but its flexible nature makes it applicable to various industries and project types. It's worth noting that while Redmine is a powerful tool, its interface might require some learning for effective use. It's recommended to review its documentation and possibly provide training for team members who are new to the platform.

The attack surface of a software application refers to the set of all possible points through which an attacker could potentially exploit vulnerabilities and compromise the system's security. While Redmine is a valuable project management and issue tracking tool, like any software, it has potential attack vectors that organizations and administrators should be aware of. Here are some aspects of Redmine's attack surface to consider:

1. **Web Interface Vulnerabilities:** Redmine's web-based interface is the primary means of interaction for users. As such, vulnerabilities like Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and SQL injection could be exploited by attackers to inject malicious code or manipulate the application's behavior.
2. **Authentication and Authorization Weaknesses:** If Redmine's authentication and authorization mechanisms are improperly configured or contain vulnerabilities, unauthorized users might gain access to sensitive project data or functionality.
3. **Plugin Security:** Redmine supports third-party plugins that can enhance its functionality. However, poorly developed or outdated plugins can introduce security vulnerabilities, so organizations should carefully vet and monitor the plugins they use.
4. **File Uploads:** Redmine allows users to attach files to issues and projects. Attackers might exploit insufficient validation of uploaded files to execute malicious code or deliver malware.

02

Technical Analysis

Technical Analysis

Cross-Site Scripting (XSS) is a security vulnerability that allows attackers to inject malicious scripts into web applications, which are then executed in the context of a victim's browser. If Redmine's input fields are not properly sanitized or encoded, an attacker could inject malicious code that is subsequently executed when other users view the affected page.

Vulnerable Component: Markdown Input Fields

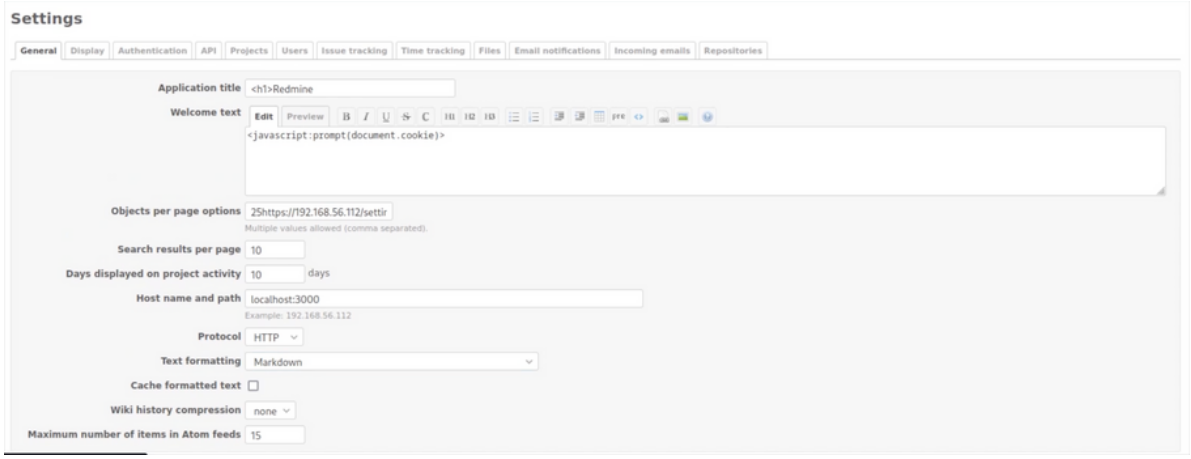
Redmine allows users to input content in Markdown format, which is then rendered as HTML. If the input fields do not properly sanitize or escape user-generated content, an attacker can inject malicious scripts that get executed when the Markdown content is rendered as HTML.
Attack Vector:
The attacker can insert a payload in a Markdown input field that will execute arbitrary JavaScript code in the context of a victim user's browser.

Payload Used:

<Javascript:prompt(document.cookie)>

Exploitation Steps:

1. The attacker inputs the payload (**<Javascript:prompt(document.cookie)>**) into a Markdown input field.
2. When another user, such as an administrator or project member, views the content containing the payload, the browser will execute the embedded JavaScript code.
3. In this case, the payload triggers the **prompt** function to display the victim's browser cookies, exposing sensitive information to the attacker.



Markdown XSS Payload Fuzzing with <Javascript:prompt(document.cookie)>

<JavaScript:prompt(document.cookie)>

Using different letter casing in the JavaScript protocol to evade basic filters.

<svg/onload=alert(document.cookie)>

Using SVG's onload attribute to trigger an alert.

<img src=x onerror=alert(document.cookie)>

Using the onerror attribute of an image tag to execute JavaScript.

Click me

Injecting JavaScript into an anchor tag's href attribute.

![Alt text](javascript:alert(document.cookie))

Attempting to execute JavaScript within an image's alt attribute.

![Alt text][1]\n\n[1]:javascript:alert(document.cookie)

Using a reference-style link to trigger JavaScript execution.

\<JaVaScRiPt:pRoMpT(document.cookie)\>

Mixing different casing and escaping to evade filters.

`javascript:alert(document.cookie)`

Using Markdown's inline code syntax to try to execute JavaScript.

<iframe src=javascript:alert(document.cookie)></iframe>

Injecting JavaScript within an <iframe> element.

Click me

Using the onclick attribute of an anchor tag to trigger an alert.

XSS Payload Execution and CSRF Token Exposure
The application is vulnerable to XSS, allowing an attacker to execute the payload **javascript:alert(document.documentElement.outerHTML)** in the context of a victim's browser. Additionally, the application exposes the **csrf-token** value in the **<body>** tag.

Attack Vector:

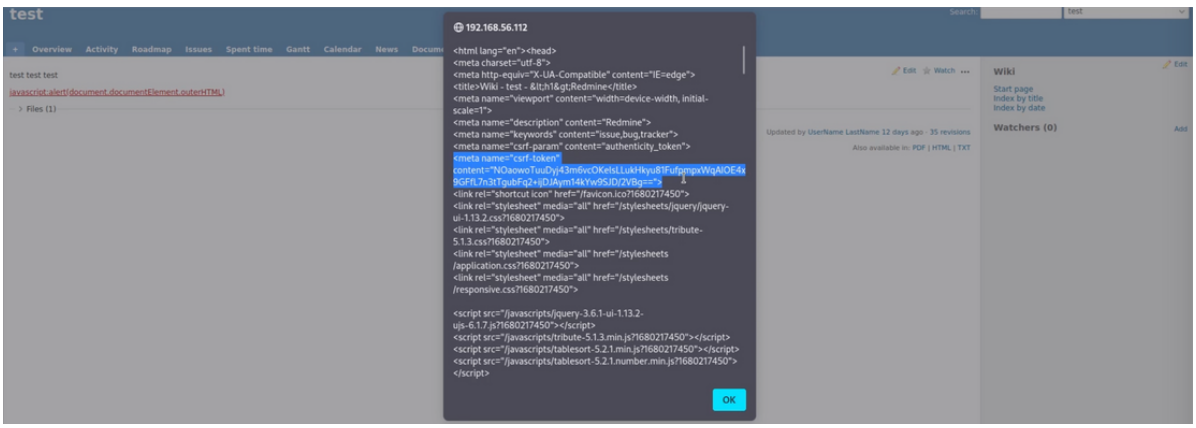
1. The attacker injects the payload as part of user-generated content, possibly in a text input field.
2. The injected payload executes within the victim's browser when they view the compromised content.
3. The payload reads the entire HTML structure using **document.documentElement.outerHTML**, including the **csrf-token**.

Exploitation Steps:

1. The attacker crafts a malicious payload containing the script **javascript:alert(document.documentElement.outerHTML)**.
2. The payload is injected into a field where the application doesn't properly sanitize input, such as a comment section.
3. When a victim views the compromised content, the payload triggers, revealing the HTML structure of the page, including the **csrf-token**.

Escalation to Account Takeover:

1. With knowledge of the victim's **csrf-token**, the attacker can craft requests that appear legitimate and include the victim's token.
2. The attacker can initiate unauthorized actions on behalf of the victim, potentially changing account settings, initiating password resets, or even taking over the account.



CSV (Comma-Separated Values) files are commonly used for storing tabular data, such as spreadsheets. CSV injection, also known as formula injection or formula injection attack, occurs when malicious data is injected into CSV files to exploit the behavior of certain spreadsheet applications. When this manipulated data is opened by a spreadsheet application, it might trigger the execution of arbitrary formulas or commands.

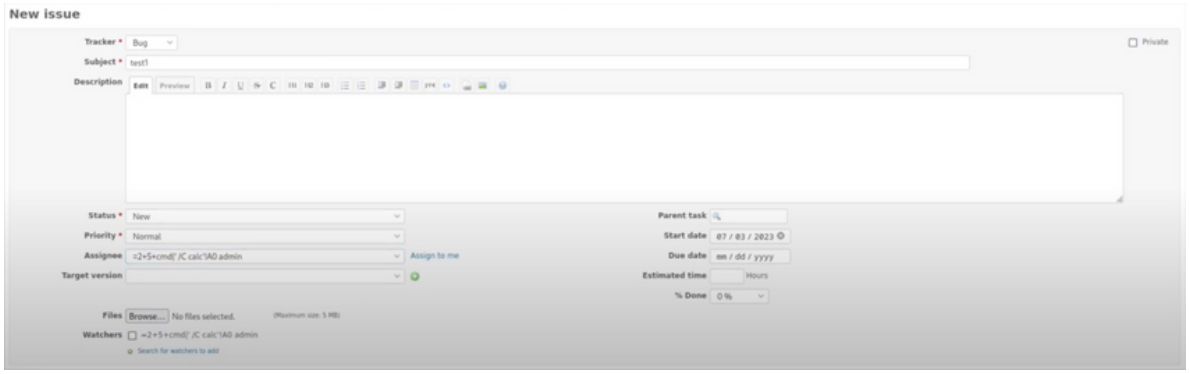
Payload Analysis:

The payload **=2+5+ cmdl* /C calc'IA0 admin** seems to be crafted for CSV injection with the intent of executing a command and possibly performing an action like opening the Windows Calculator (**calc**) or running a command with arguments.
Let's break down the payload:

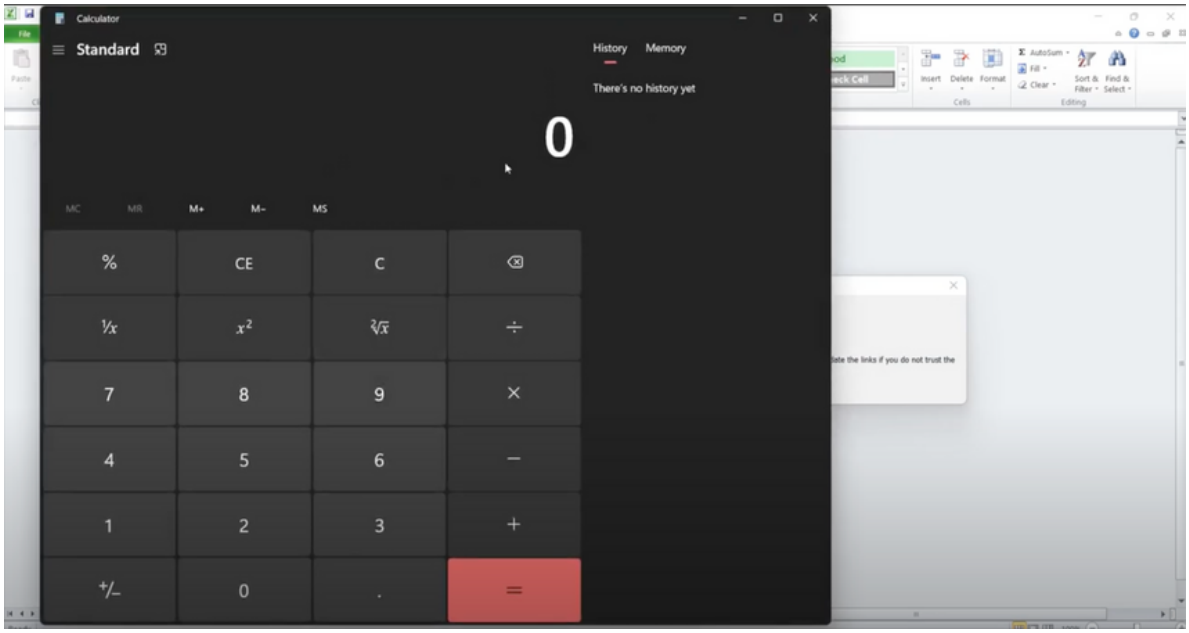
- **=2+5+**: This portion adds up the numbers 2 and 5, resulting in 7. It's likely used to ensure that the formula itself evaluates to a numeric value.
- **cmdl* /C calc'IA0 admin**: This part appears to be a command-line command intended to run a calculator (**calc**). However, there seem to be some errors or typos in the syntax:
  - **cmdl*** should likely be **cmd**, which is the command prompt executable on Windows systems.
  - **/C** is a valid option for the **cmd** command, indicating that the following string should be executed as a command.
  - **calc** is the command to open the Windows Calculator.
  - **'IA0** and **admin** seem to be unclear or incorrect parts of the payload.

Potential Impact:

If this payload is injected into a CSV file and opened by a spreadsheet application that supports formula execution, the formula might be processed, and the command could potentially be executed. However, the syntax issues in the payload might prevent it from working as intended.



The attacker's goal is to craft a payload that, when interpreted by the spreadsheet application, will lead to command execution. Here's a general sequence of steps they might follow:
a. **Choosing a Formula Entry Point:** Identify cells in the CSV file that are interpreted as formulas. For example, the attacker might target cells under columns labeled "Formula" or similar.
b. **Crafting the Formula:** The attacker starts the payload with an = character to indicate a formula. They then add commands or functions that the spreadsheet application might execute.
c. **Embedding Commands:** The attacker might use different methods to embed commands within the formula. For example, they might use the **HYPERLINK** function or other built-in functions to execute a command.
d. **Hiding Malicious Content:** The attacker often tries to make the payload blend in with legitimate data to avoid detection. They might use URL encoding or other obfuscation techniques to hide the malicious content.



03

Conclusion

In conclusion, both Cross-Site Scripting (XSS) vulnerabilities leading to account takeover with the ability to read a **csrf-token** from the **<body>** tag and CSV injection vulnerabilities that could result in OS command execution are critical security risks that need immediate attention and mitigation.

XSS to Account Takeover with CSRF Token Exposure: The combination of an XSS vulnerability allowing an attacker to inject malicious scripts, paired with the ability to read sensitive information such as a **csrf-token**, creates a significant risk for account takeover. By exploiting XSS, an attacker can compromise user accounts, impersonate victims, and potentially gain unauthorized access to sensitive data. The exposure of a **csrf-token** could facilitate unauthorized actions on behalf of victims, making the impact more severe. Organizations should prioritize input validation, output encoding, and secure handling of sensitive tokens to prevent such attacks.

CSV Injection Leading to OS Command Execution: CSV injection poses a serious threat when attackers manipulate CSV files to execute arbitrary commands on the host system. By crafting malicious formulas within cells, attackers can leverage spreadsheet application behavior to execute unintended commands. This can lead to unauthorized access, data breaches, and even full-scale system compromise. To mitigate this risk, thorough input validation, encoding of user-generated content, and user education are crucial. Preventing CSV injection vulnerabilities requires strong security practices and awareness of how spreadsheet applications interpret data.

HADESS

cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.

Website:
WWW.HADESS.IO

Email
MARKETING@HADESS.IO

Hadess Cyber Security Solutions is a renowned group of cyber security experts and white hat hackers. With a track record of discovering and responsibly reporting vulnerabilities to prominent companies such as Google, Apple, and Twitter, Hadess has earned a reputation for excellence in the field of cyber security.