

# Wordpress Plugins Security Analysis

40 0DAYS IN 40 DAYS



HADESS

[WWW.HADESS.IO](http://WWW.HADESS.IO)



# TABLE OF CONTENTS

Zero-Day Vulnerability

1

Vulnerabilities

2

Plugins

3



# Introduction

---

We are excited to announce the launch of our 40 Vulnerabilities in 40 Days Campaign! Our goal is to raise awareness about the importance of proactive vulnerability management and to encourage everyone to take action to secure their systems.

Starting from March 1st, we will be showcasing one vulnerability every day for 40 days, along with details on how to detect and remediate it. Our team of experts will be available to provide insights and best practices, so you can learn from real-world scenarios and understand the impact of these vulnerabilities.

We believe that knowledge is power, and by educating ourselves and others, we can help make the world a safer place. Join us and become a part of the 40 Vulnerabilities in 40 Days Campaign today!

**SECTION 1**
















# Zero-Day

A zero-day vulnerability is a security weakness in software or hardware that is unknown to the party responsible for patching or otherwise protecting the system. This vulnerability can be exploited by attackers to conduct malicious activities such as unauthorized access to sensitive data, spreading malware, or disrupting normal operations.

Zero-day vulnerabilities are particularly dangerous because they can be used by attackers before the vendor has had a chance to release a patch or a fix for the issue. Attackers can take advantage of these vulnerabilities to launch targeted attacks, which can have serious consequences, such as data theft, financial loss, or reputational damage.

**SECTION 1**

# Vulnerabilities List

<b>CVE-2022-45834</b>		<b>CSRF</b>
<b>CVE-2022-4367</b>		<b>CSRF</b>
<b>CVE-2022-4011</b>		<b>CSRF</b>
<b>CVE-2022-3941</b>		<b>CSRF</b>
<b>CVE-2022-4412</b>		<b>CSRF</b>
<b>CVE-2022-4411</b>		<b>CSRF</b>
<b>CVE-2022-4406</b>		<b>CSRF</b>
<b>CVE-2022-4405</b>		<b>CSRF</b>
<b>CVE-2022-4404</b>		<b>CSRF</b>
<b>CVE-2022-4528</b>		<b>CSRF</b>
<b>CVE-2022-4529</b>		<b>CSRF</b>
<b>CVE-2022-4530</b>		<b>CSRF</b>
<b>CVE-2022-4531</b>		<b>CSRF</b>
<b>CVE-2022-4532</b>		<b>CSRF</b>
<b>CVE-2022-4533</b>		<b>CSRF</b>

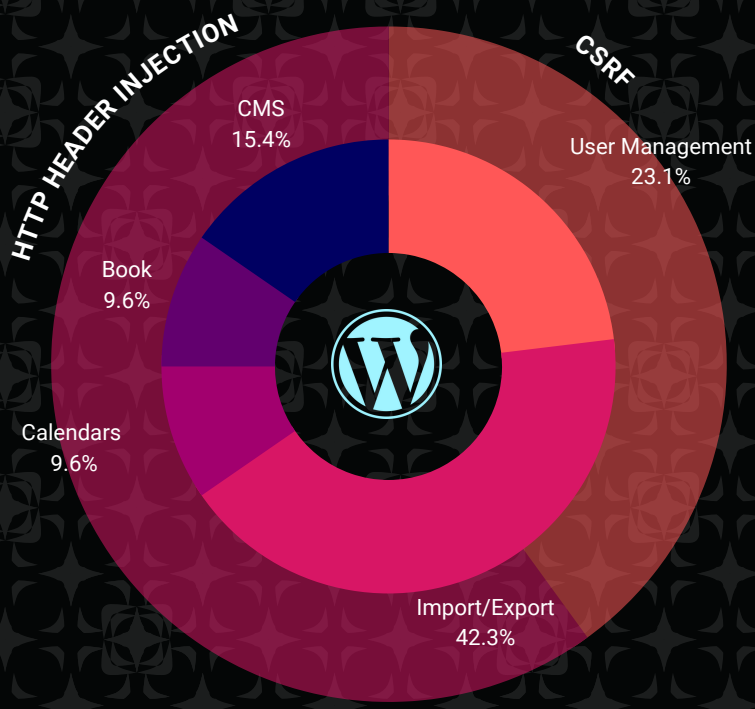


<b>CVE-2022-4534</b>		<b>CSRF</b>
<b>CVE-2022-4535</b>		<b>CSRF</b>
<b>CVE-2022-4536</b>		<b>CSRF</b>
<b>CVE-2022-4537</b>		<b>CSRF</b>
<b>CVE-2022-4538</b>		<b>CSRF</b>
<b>CVE-2022-4539</b>		<b>CSRF</b>
<b>CVE-2022-4540</b>		<b>CSRF</b>
<b>CVE-2022-4541</b>		<b>CSRF</b>
<b>CVE-2022-4550</b>		<b>CSRF</b>
<b>CVE-2022-46847</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-4423</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-4424</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-4425</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-4443</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47171</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47163</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47162</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47159</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47155</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47154</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47152</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47147</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47138</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47141</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47143</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47139</b>		<b>HTTP Header Injection</b>
<b>CVE-2022-47135</b>		<b>HTTP Header Injection</b>

# Vulnerabilities List

CVE-2022-47448	HTTP Header Injection
CVE-2022-47447	HTTP Header Injection
CVE-2022-47440	HTTP Header Injection
CVE-2022-47446	HTTP Header Injection
CVE-2022-47443	HTTP Header Injection
CVE-2022-47422	HTTP Header Injection
CVE-2022-4549	HTTP Header Injection
CVE-2022-4548	HTTP Header Injection
CVE-2022-47427	HTTP Header Injection

# Plugins Type



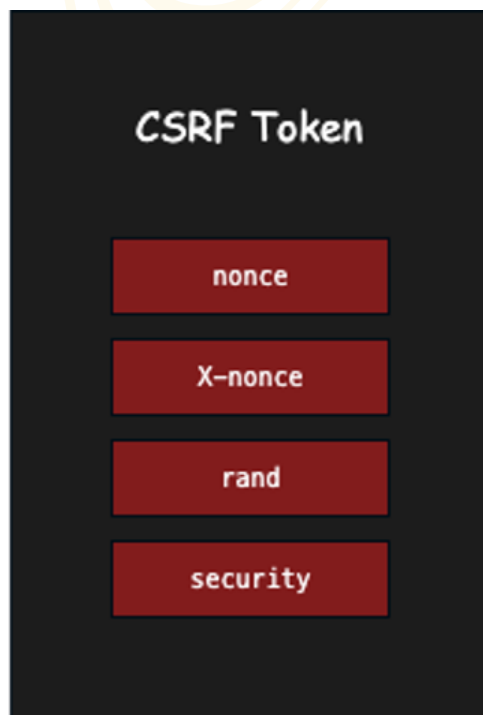






The displayed code snippet is used in WordPress and can save the information in the CSV file to the WordPress database. After filling the WordPress form with the required information, the code connects to the WordPress database and stores the information in the CSV file in the desired table of the WordPress database. This code also has a special value "update\_db" which, if checked, will update the data in the CSV file instead of creating a new record. If no records have been updated, a success message is displayed. But if there is a problem connecting with the database, an error message will be displayed.

### **CSRF Protection Parameter**



One of the useful methods to prevent CSRF vulnerability in the development of WordPress plugins is to use unique tokens. To use this method, you need to use tokens in your plugin development code that are uniquely added to the form. For example, you can use the form author token.

For example, in the code below, the form author token is added as a variable containing a random value in the form:

```
1 function add_csrf_nonce_field() {  
2     wp_nonce_field( 'csrf_nonce_action', 'csrf_nonce_name'  
3 };  
4 add_action( 'form_name_form_tag', 'add_csrf_nonce_field'
```

In the code below, the token of the form author is checked after submitting the form, and if the token is not correct, the form is considered invalid:

```
1 function verify_csrf_nonce_field() {  
2     if ( !isset( $_POST['csrf_nonce_name'] ) || !wp_verify_nonce( $_POST['csrf_nonce_name'], 'csrf_nonce_action' ) )  
3 {  
     wp_die(
```

You can also use change tokens like time token. For example, you can add a time token like this:

```
1 function add_csrf_nonce_field() {  
2     $nonce_value = time();  
3     echo '<input type="hidden" name="csrf_nonce_name" value="' . $nonce_value . '" />';  
4 }  
5 add_action( 'form_name_form_tag', 'add_csrf_nonce_field' );
```

In the following code, the time token is checked after submitting the form and if the token is older than one minute, it considers the form invalid:

```
1 function verify_csrf_nonce_field() {  
2     $submitted_nonce = isset( $_POST['csrf_nonce_name'] ) ? intval( $_POST['csrf_nonce_name'] ) :  
3 0; if ( time() - $submitted_nonce > 60 ) {  
4         wp_die( 'Invalid CSRF nonce. Please try again.' );  
5     }  
6 }  
7 add_action( 'form_name_process_action', 'verify_csrf_nonce_field');
```



You can also use WordPress framework features, such as the `wp_verify_nonce` class, to prevent CSRF. Using the `wp_verify_nonce` class in the WordPress framework is a useful and advanced method to prevent CSRF attacks.

The `wp_verify_nonce` class uses a random token that is generated on each new request and after the form is submitted, the token is compared with the token submitted in the form.

To use the `wp_verify_nonce` class in the development of WordPress plugins, you can use the following code:

```
1 function add_csrf_nonce_field() {  
2     wp_nonce_field( 'csrf_nonce_action', 'csrf_nonce_name' );  
3 }  
4 add_action( 'form_name_form_tag', 'add_csrf_nonce_field' );  
5  
6 function verify_csrf_nonce_field() {  
7     if ( !isset( $_POST['csrf_nonce_name'] ) || !wp_verify_nonce( $_POST['csrf_nonce_name'], 'csrf_nonce_action' ) )  
8     {  
9         wp_die('Invalid CSRF nonce. Please try again.');
```

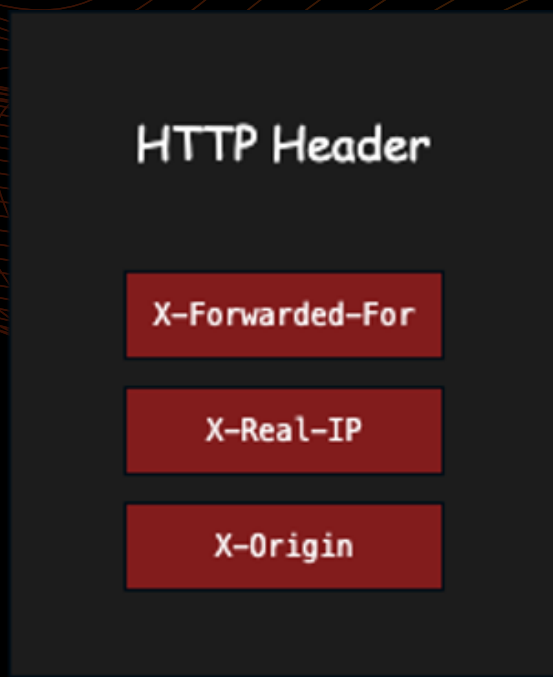
## HTTP Header Injection

```
1 POST /wp-login.php HTTP/1.1  
2 Host: localhost  
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/109.0  
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
5 Accept-Language: en-US,en;q=0.5  
6 Accept-Encoding: gzip, deflate  
7 Referer: http://localhost/wp-login.php  
8 Content-Type: application/x-www-form-urlencoded  
9 Content-Length: 103  
10 Origin: http://localhost  
11 Connection: close  
12 Cookie: amp_d59f9d=ERnkChTbaI0DMJrTas9SFN...1gk2lofbc.1gk2mu60i.0.0.0; session=b8dd9c85-204f-4c56-94f4-  
    deb76b12a9fc.0m054JetV2KWkLLS8pV2LSME5w0; TZ=-12600;  
    fm_cookie_605921e05e4e17a736ecaa4dd4099774=605921e05e4e17a736ecaa4dd4099774; PHPSESSID=fd128c23fa58597c4ee00fc022913a8a;  
    wordpress_test_cookie=WP%20Cookie%20check  
13 Upgrade-Insecure-Requests: 1  
14 Sec-Fetch-Dest: document  
15 Sec-Fetch-Mode: navigate  
16 Sec-Fetch-Site: same-origin  
17 Sec-Fetch-User: ?1  
18  
19 log=admin&pwd=admin&wp-submit=Log+In&redirect_to=http%3A%2F%2Flocalhost%2Fwp-admin%2F&testcookie=1
```



The displayed code attempts to obtain the IP address of the user requesting the page. This is done by using different variables on the server side. This code can be useful for users who use proxy or proxy server. Normally, the user's IP address is shown by default in a part of the browser's address bar, but in some cases, the user's IP address can be hidden by using a proxy or proxy server. The displayed code can obtain the user's IP address using the standard `HTTP_CLIENT_IP`, `HTTP_X_FORWARDED_FOR`, `HTTP_X_FORWARDED`, `HTTP_FORWARDED_FOR` and `HTTP_FORWARDED`. If none of these variables are available, the IP address is considered invalid and the function returns an empty value. Also, the `filter_var` function is used to check the correctness of the filtered IP address. If the IP address is not valid, the function returns empty.

## HTTP Header Injection Methods







A useful way to avoid http header injection vulnerability in WordPress plugin development is to use `wp_remote_get` and `wp_remote_post` functions. These functions use the cURL library and by default, the http header is also filtered in all the texts that are sent.

To use the `wp_remote_get` and `wp_remote_post` functions in the development of WordPress plugins, you can use the following code:

```
1 $response = wp_remote_get( 'http://example.com/api/endpoint' );-
2 if ( is_wp_error( $response ) ) {-
3     wp_die( 'Error: ' . $response->get_error_message() );-
4 }-
5 -
6 $response = wp_remote_post( 'http://example.com/api/endpoint',
7 array(
8     'param1' => 'value1',-
9     'param2' => 'value2'-
10 )-
11 ) );-
12 if ( is_wp_error( $response ) ) {-
13     wp_die( 'Error: ' . $response->get_error_message() );-
14 }
```

You can also use the `wp_http_validate_url` function to filter invalid entries.

The function checks whether the submitted URL is valid or not. If the URL is invalid, the `wp_http_validate_url` function will return an error and you will not be able to execute your code.

To use the `wp_http_validate_url` function in the development of WordPress plugins, you can use the following code:

```
1 $url = 'http://example.com';-
2 if ( !wp_http_validate_url( $url ) ) {-
3     wp_die( 'Error: Invalid URL' );-
4 }
```



You can also use the `esc_url_raw` function to control Internet access. This function filters the submitted URL and deletes it if the URL is invalid.

To use the `esc_url_raw` function in the development of WordPress plugins, you can use the following code:

```
1 $url = 'http://example.com';  
2 $url = esc_url_raw($url);
```

Cross-Site Request Forgery (CSRF) is a type of security vulnerability that occurs when a malicious website is able to trick a user's browser into sending a request to another website, without the user's knowledge or consent. This can be used to steal sensitive information or to perform unauthorized actions on behalf of the user.

1. Synchronizer token pattern: This involves adding a unique token to each form or request that is generated by the server and passed to the client. The client must then include this token in all subsequent requests to the server, which the server can then use to verify that the request was initiated by the user.
2. Same-Site Cookies: This is a flag that can be set on a cookie, which tells the browser to only send the cookie on requests to the same domain that set the cookie. This makes it more difficult for a malicious website to perform a CSRF attack, as it won't have access to the necessary cookies.
3. Referrer header check: This involves checking the value of the referrer header in each request to ensure that it was sent from the same domain. This helps to prevent CSRF attacks that use methods such as image tags or JavaScript to send requests.
4. CAPTCHA: A CAPTCHA can be used to require the user to prove that they are a human before performing a sensitive action. This makes it more difficult for a malicious website to perform a CSRF attack, as it won't be able to complete the CAPTCHA.
5. Use of Anti-CSRF libraries: Anti-CSRF libraries, such as the OWASP CSRFGuard, provide an easy-to-use solution for protecting against CSRF attacks by automatically generating and validating tokens for each request.

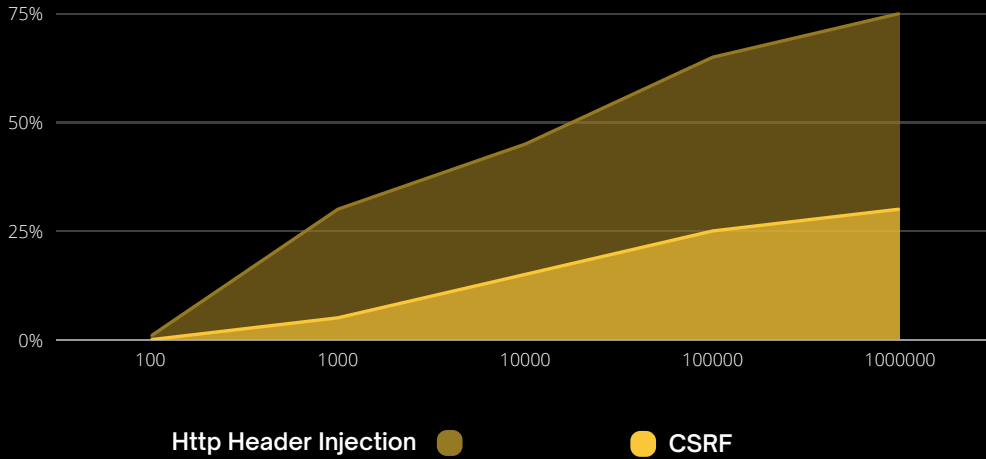
HTTP header injection is a type of security vulnerability that occurs when an attacker is able to inject malicious content into HTTP headers. This can be used to manipulate the behavior of web applications or to steal sensitive information.

To prevent HTTP header injection, it's important to follow these best practices:

1. **Input validation:** Ensure that all user-supplied data is properly validated and sanitized to prevent malicious data from being included in HTTP headers.
2. **Use secure encoding and decoding functions:** Use secure encoding and decoding functions, such as HTML entities and URL encoding, to ensure that special characters are properly handled.
3. **Limit header size:** Limit the size of HTTP headers to prevent excessive data from being included in the headers.
4. **Use a web application firewall (WAF):** A WAF can provide protection against HTTP header injection by detecting and blocking malicious requests.
5. **Keep software up to date:** Ensure that all software, including web applications and the underlying operating system, is kept up to date to address known vulnerabilities.
6. **Regularly perform security testing:** Regularly perform security testing, including penetration testing, to identify and remediate vulnerabilities in your systems.

## Vulnerabilities Type

The type of vulnerability detected in plugins by the number of plugins installed



Identified vulnerabilities were investigated separately by the amount of installation and test cases related to CSRF and HTTP Header Injection vulnerabilities.

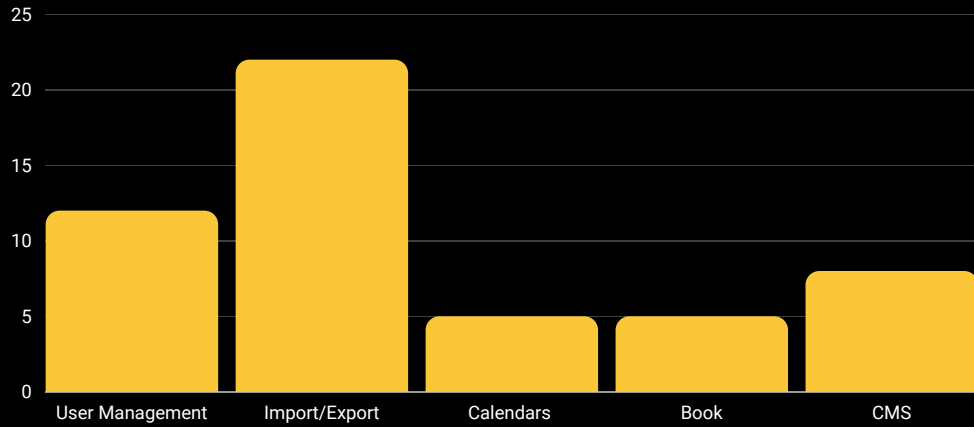
Also, the above chart is divided based on the frequency of vulnerabilities detected in about 1000 checked plugins.



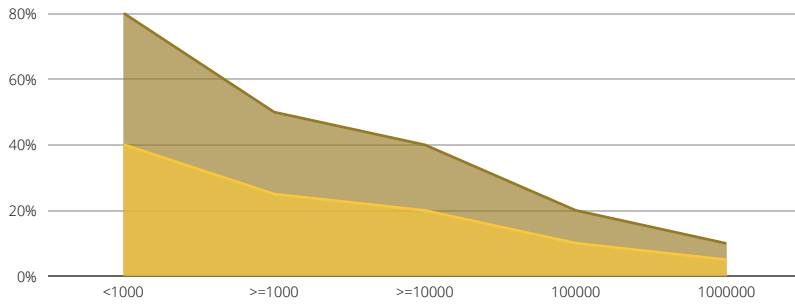


### Plugin Type

A vulnerability has been found for each type of plugin

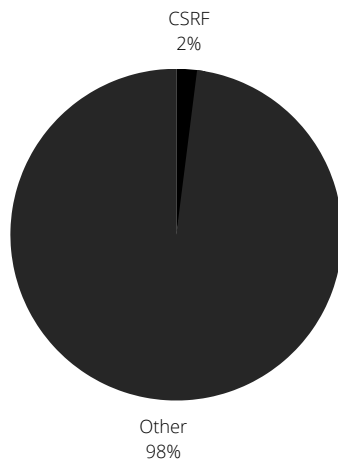


Checked plugins are divided based on the type of functionality and application of the plugin, as well as the amount of installation and the amount of vulnerabilities detected in each package.



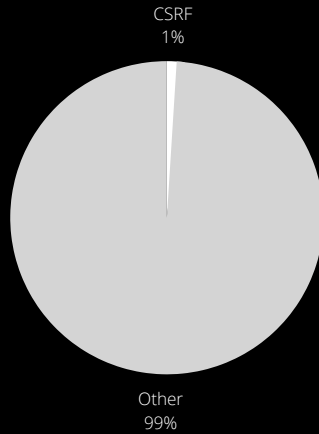
**Installation Count**

PLUGINS WITH MORE THAN 1000 INSTALLATIONS HAVE MORE THAN 10 VULNERABILITIES OUT OF MORE THAN 40 VULNERABILITIES.



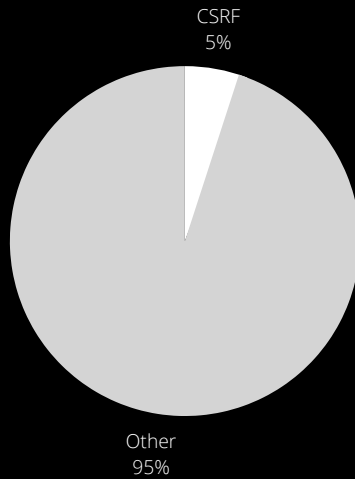
**آسیب پذیری CSRF**

OUT OF 100 PLUGINS WITH MORE THAN 1000 INSTALLS, TWO ARE VULNERABLE.



**Impact**

Any restriction on their side could be removed per vulnerability

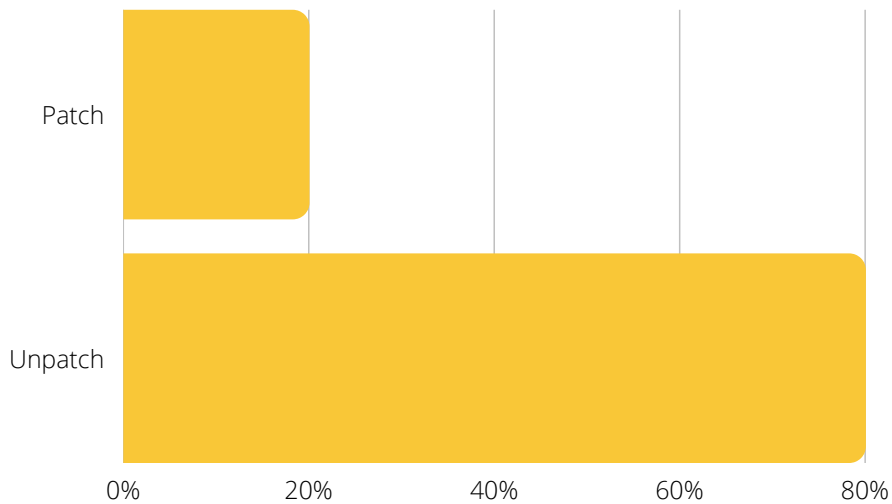


**HTTP Header Injection**

A plugin with +10000 installations was found in plugin list to be vulnerable to http header injection from 100 plugins

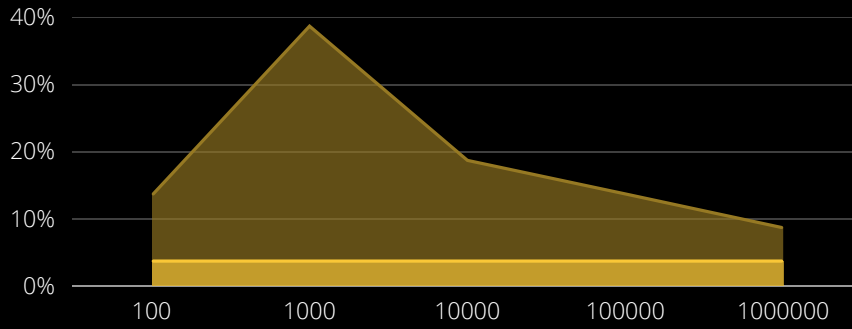
**Patch Time**

More than 80% of vulnerabilities are not patched after they are reported



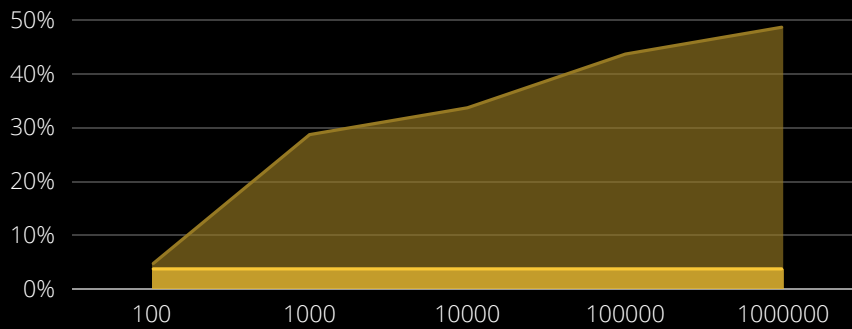
## Manually/Automatically

Vulnerabilities are found using which method



**Automatically**

80%



**Manually**

20%

## Root-Cause

No	Problem	Vulnerability
1	Wordpress CSRF Protection Misuse	CSRF
2	Lack Input Validation	CSRF
3	HTTP Header White-list	HTTP Header Injection

# About Hades

Savior of your Business to combat cyber threats

Hades performs offensive cybersecurity services through infrastructures and software that include vulnerability analysis, scenario attack planning, and implementation of custom integrated preventive projects. We organized our activities around the prevention of corporate, industrial, and laboratory cyber threats.

## Contact Us

To request additional information about Hades's services, please fill out the form below. A Hades representative will contact you shortly.

**Website:**

[www.hadess.io](http://www.hadess.io)

**Email:**

[Marketing@hadess.io](mailto:Marketing@hadess.io)

**Phone No.**

+989362181112

**Company No.**

982128427515

hadess\_security





# Hadess

## Products and Services

### → **SAST | Audit Your Products**

Identifying and helping to address hidden weaknesses in your Applications.

### → **RASP | Protect Applications and APIs Anywhere**

Identifying and helping to address hidden weaknesses in your organization's security.

### → **Penetration Testing | PROTECTION PRO**

Fully assess your organization's threat detection and response capabilities with a simulated cyber-attack.

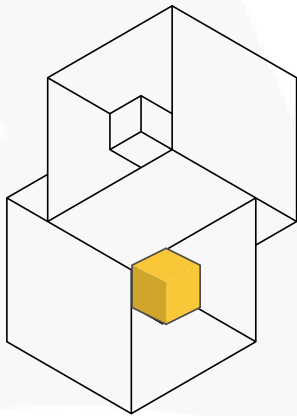
### → **Red Teaming Operation | PROTECTION PRO**

Fully assess your organization's threat detection and response capabilities with a simulated cyber-attack.

### → **ThirdEye | Attack Surface Intelligence**

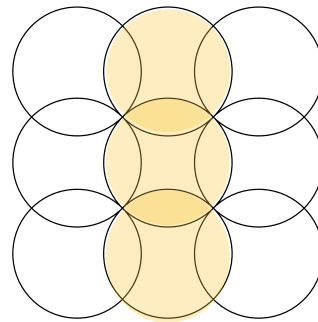
Find your company leakage and monitor attack vector.

# Penetration Testing



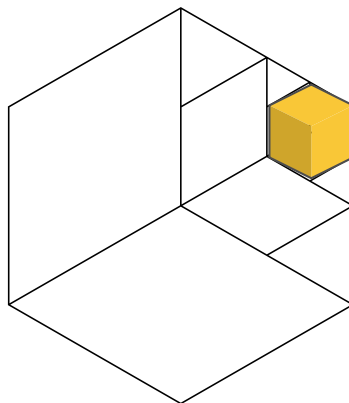
### Module-Based

Penetration testing typically involves a combination of several different testing methods and techniques, which can be grouped into different modules. The specific modules used in a penetration test will depend on the goals and scope of the test, as well as the systems and services being evaluated.



### Target-Based

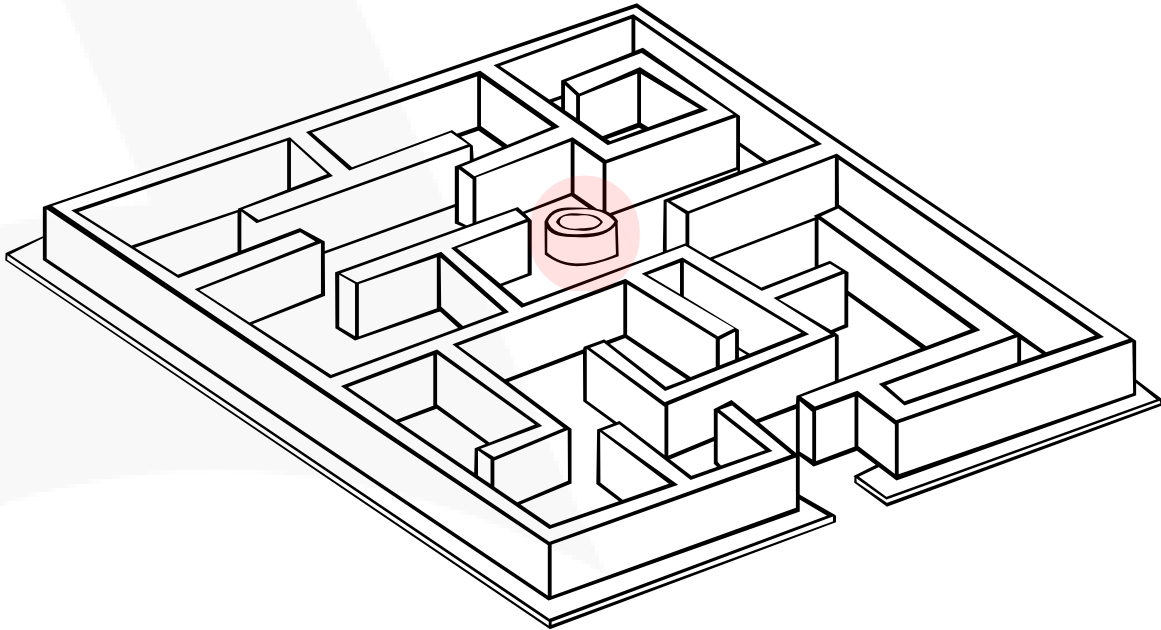
Penetration testing is a simulated cyber attack performed on a computer system, network, or web application to evaluate its security posture. The target of a penetration test can vary based on the specific needs and goals of the organization.



### Priority-Based

The priority of a vulnerability during a penetration test is determined by the potential impact of the vulnerability, if exploited, and the likelihood of it being exploited. Vulnerabilities that pose a high risk to the target systems and services are given a higher priority, while those with a lower risk are given a lower priority.

## Red Team Operation



### OSINT

OSINT (Open-Source Intelligence) is a valuable tool for red teams, as it provides them with the ability to gather information about a target in a non-intrusive manner. Red teams use OSINT to gather information about the target's infrastructure, personnel, systems, and operations. This information can be used to identify potential weaknesses and vulnerabilities that can be exploited during a penetration test or simulated attack.

### Hardening

Red team hardening is a technique used by red teams to evaluate an organization's security posture by simulating attacks and attempting to exploit vulnerabilities. The goal of red team hardening is to identify and remediate security weaknesses in an organization's systems, processes, and people, so that they are better prepared to defend against real-world attacks.

### Goal-Based

The goal of a red team exercise is to simulate an attack on an organization's systems, processes, and people to identify security weaknesses and vulnerabilities. Red teams use a variety of techniques, including penetration testing, social engineering, and physical security assessments, to test the effectiveness of an organization's defenses and to identify areas where they can be improved.

### Asset-Based

Red team asset-based testing involves simulating an attack on a specific asset or group of assets within an organization. The goal of this type of red team exercise is to identify vulnerabilities in the targeted assets and to evaluate the effectiveness of the organization's defenses in protecting those assets.

## SAST

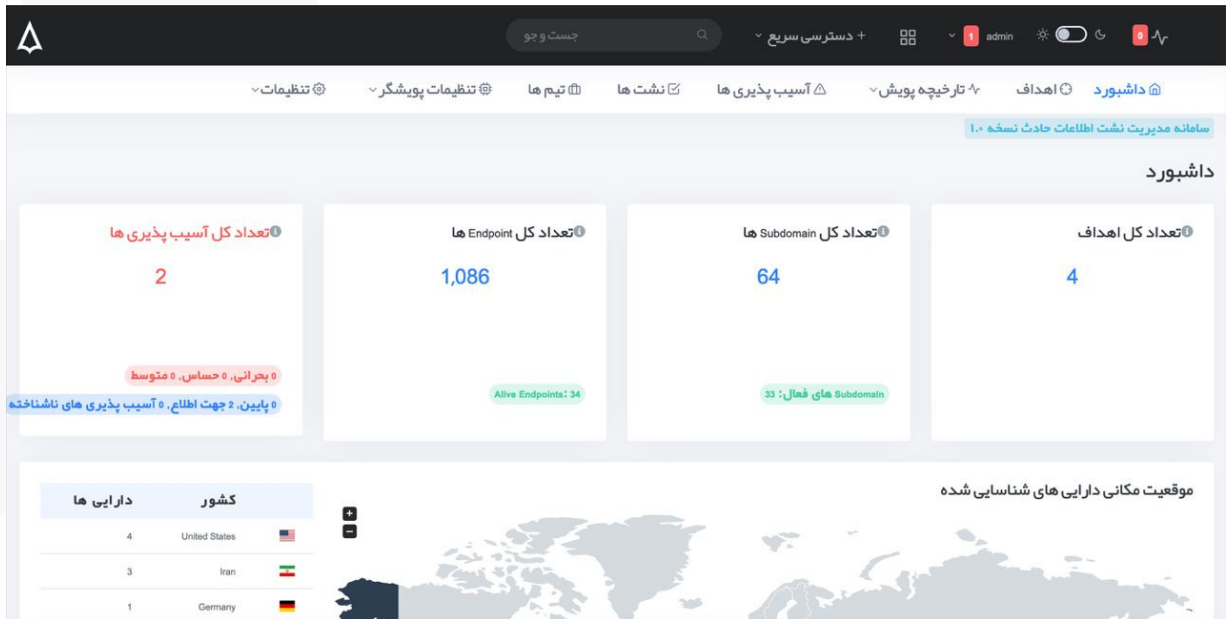
کاربر	وضعیت	درجه اهمیت	زمان	نسخه	ممنیزی	مسیر فایل	
developer2	بررسی شده (در گذشته ثبت شده)	خطا	2022-10-06 22:50	93d2ef3d	SQL Injection Userinput reaches sensitive sink. For more inform: پیام خطا: .ation, press the help icon on the left side	sql2.php sql2.php	<input type="checkbox"/>
developer2	بررسی شده (در گذشته ثبت شده)	خطا	2022-10-06 22:49	ecc8f20e	SQL Injection Userinput reaches sensitive sink. For more inform: پیام خطا: .ation, press the help icon on the left side	sql1.php sql1.php	<input type="checkbox"/>
developer2	بررسی شده (در گذشته ثبت شده)	خطا	2022-10-06 22:51	04edab7f	SQL Injection Userinput reaches sensitive sink. For more inform: پیام خطا: .ation, press the help icon on the left side	sql6.php sql6.php	<input type="checkbox"/>
developer2	بررسی شده (در گذشته ثبت شده)	خطا	2022-10-06 22:50	dc967926	SQL Injection Userinput reaches sensitive sink. For more inform: پیام خطا: .ation, press the help icon on the left side	sql3.php sql3.php	<input type="checkbox"/>
developer2	بررسی شده (در گذشته ثبت شده)	خطا	2022-10-06 22:51	124a852a	SQL Injection Userinput reaches sensitive sink. For more inform: پیام خطا: .ation, press the help icon on the left side	sql4.php sql4.php	<input type="checkbox"/>

SAST (Static Application Security Testing) is a type of security testing that involves analyzing the source code of an application, without actually executing the code, to identify potential security vulnerabilities. SAST is performed early in the software development lifecycle, before the application is deployed, and is typically integrated into the development process as part of a DevSecOps approach.



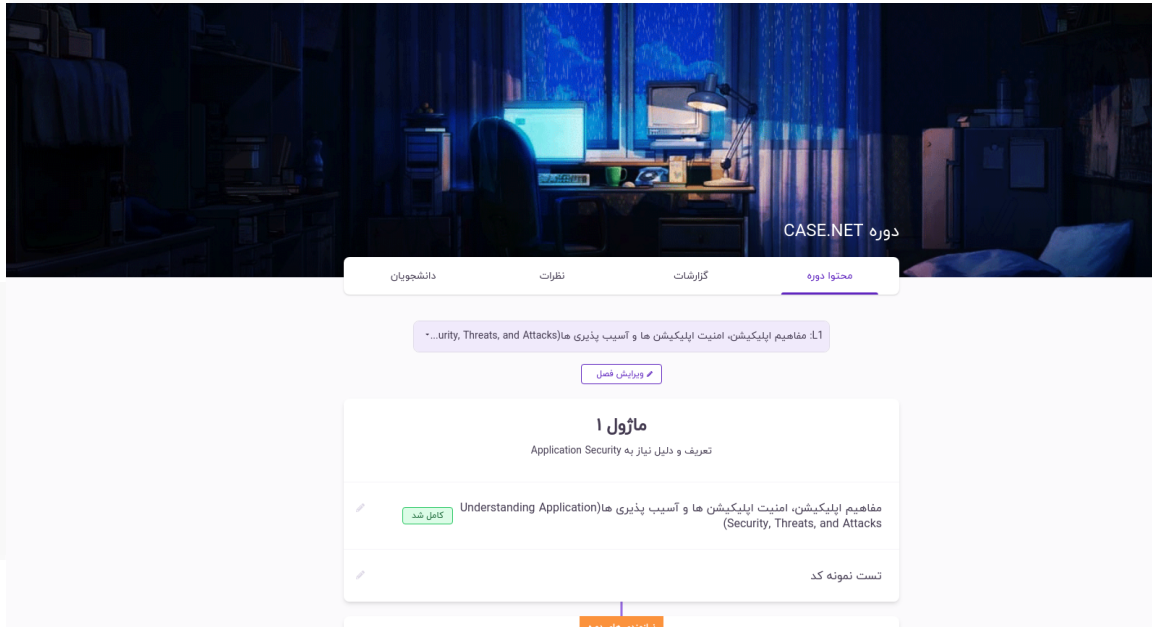
# HADESS

# ASM



Attack surface management (ASM) is a security practice that involves reducing the attack surface of an organization's systems and services, making them less vulnerable to cyber attacks. The attack surface refers to the total number of potential entry points for an attacker, including network interfaces, applications, services, and other elements of an organization's technology infrastructure.

# Secure Coding



Secure coding is a software development practice that involves writing code that is free from vulnerabilities and that follows best practices for security. The goal of secure coding is to prevent security issues and vulnerabilities from being introduced into an application during the development process.

