|  |  |
| --- | --- |
| Serial number | {report\_number} |
| Date | {report\_date} |

**{target\_host}**

**Platform API Fuzzy Detection Report**

Version 1.2

2020/12

© 2020 Packer FuzzeR TEAM

Poc-Sir、KpLi0rn、Liucy、RachesseHS、Lupin-III

* **Copyright**

The Packer Fuzzer Inspection Report (hereinafter referred to as this report) template is copyrighted by the Packer Fuzzer development team (hereinafter referred to as the team) and is protected by law. The team has the right to modify and interpret this report template. When modifying the contents of this report template, you should retain the appropriate copyright notice. The Packer Fuzzer tool (hereinafter referred to as the tool) may not be used in any way for commercial purposes without the authorization of this team. The team will reserve the right to further pursue legal responsibility for any individual or company that violates the above statement.

* **Statements of application**

This report applies to all target systems tested with this tool, so please keep it safe and do not pass it on without the permission of the target system owner.

* **Disclaimer**

This report is automatically generated by the tool based on the user's test results, and the contents of the report do not represent the position and opinion of our team. Any direct or indirect consequences and losses caused by the dissemination or use of the testing functions provided by this tool are the responsibility of the user, and the team does not assume any responsibility for them. Please comply with the local laws and regulations of the user and the country where the target system is located when using this tool, any unauthorized testing is not allowed.

* **Revision history**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Revision date** | **Revisor** | **Modifi** | **Ratifier** |
| 1.0 | 2020/8/9 | RachesseHS | None | Poc-Sir |
| 1.2 | 2020/12/18 | RachesseHS | Add more | Poc-Sir |

**I. Summary of the report**

**Scanned platforms：{target\_host}**

Input parameter values：{target\_url}

This scan uses {scan\_type} Scan mode，use {scan\_ip} As scan IP，total time spent {scan\_time}sec。

Initiate scanning time：{start\_time}

Scan completion time：{end\_time}

The scan found {api\_num} valid API interfaces.

{js\_num} related JS files found.，respectively：

{js\_list}

A total of{vuln\_num} security vulnerabilities were identified，Of which {vuln\_h\_num} high-risk、{vuln\_m\_num} medium-risk、{vuln\_l\_num} low-risk。respectively：

{vuln\_list}

Additional Cookies Information：{extra\_cookies}

Additional transmission header information：{extra\_head}

**Security risk level of the target platform as analyzed by this tool：{sec\_lv}**

**II. Vulnerability details**

{vuln\_deta}

**III. List of APIs**

{api\_list}

**IV. Security recommendations**

{suggest\_foryou}

**V. Extra Info**

{extra\_tree}

**VI. Appendix**

* **Description of vulnerabilities**

**CORS vulnerability：**Cross-domain resource sharing can relax the browser's homology policy, which allows different websites and different servers to communicate with each other through the browser. Suppose a user logs on to vuln.com, a website with a CORS configuration, and also accesses evil.com, a link provided by the attacker. The evil.com website makes a request to vuln.com for sensitive data, and the browser's ability to receive the information depends on the configuration of vuln.com. If vuln.com is configured with the Access-Control-Allow-Origin header and is expected, then it is allowed to receive it, otherwise the browser will not receive it due to the same origin policy.

**Unauthorized Access Vulnerability:** Unauthorized access to the interface, as the name implies, can directly access and operate the corresponding business logic functions without requesting authorization. This is usually caused by a flawed or unauthenticated authentication page, improper security configuration, etc.

**Sensitive Information Disclosure Vulnerability:** Information disclosure refers to the disclosure of sensitive information in a website page or JS file. Through this sensitive information, an attacker can further compromise the server.

**Horizontal Override Vulnerability:** an override vulnerability is when an application does not strictly verify the identity permissions of the current user's operations, resulting in users being able to operate functions that are beyond their administrative privileges, thus operating some behaviors that are not available to that user. Level override can result in users between the same level having access to each other's sensitive information, such as name, phone number, contact address, personal data, order history, and so on. It may also be possible to perform a line of functions, such as delete, add, modify, etc., as other users with level override privileges.

**SQL Injection Vulnerability:** SQL injection vulnerability arises because the web application is not written for the user to submit data to the server to verify the legitimacy of the data (type, length, legitimacy of business parameters, etc.), and there is no effective special character filtering of user input data, making the user input directly into the database execution, beyond the expected results of the original design of the SQL statement, resulting in a SQL injection vulnerability.

**Weak password vulnerability:** Website management and operations personnel use very easy to remember passwords or directly adopt the system's default password due to insufficient security awareness, in order to facilitate and avoid forgetting passwords. Attackers can use this vulnerability to directly enter the application system or management system, so as to tamper with and delete the system, web pages, data, illegally access the system, user data, and may even lead to the fall of the server.

**Arbitrary file upload vulnerability:** The application system checks the legitimacy of the file type, format and content of the file uploaded by the user at the file upload function, which allows an attacker to upload a malicious Web shell script file or a file of non-expected format such as: HTML file, SHTML file, etc. At the same time, he can use the directory jump characters or control the upload. directory, uploading files directly to the web directory or any directory, which may result in the execution of arbitrary malicious script files on a remote server to gain direct access to the application system.

* **Vulnerability levels**

This report has three built-in vulnerability levels, which are: low, medium and high risk. The high-risk vulnerability types are: weak password vulnerability, arbitrary file upload vulnerability, SQLi vulnerability; the medium-risk vulnerability types are: horizontal leapfrogging vulnerability, sensitive information disclosure vulnerability, unauthorized access vulnerability; the low-risk vulnerability types are: CORS vulnerability.

The confidence level for the "low" level of detection results will automatically reduce the vulnerability of a level of harm, if it is at the lowest vulnerability level is not a downgrade. For example, a vulnerability for: SQL injection vulnerability, should be a high-risk vulnerability, but the confidence level is "low", it is automatically downgraded to the risk of vulnerability.

* **Risk level**

This report has four risk levels: no risk, low risk, medium risk, and high risk, with a scoring scale of 0, 5, 10, and 18, respectively. For example, in a certain scan, one high-risk, two medium-risk and five low-risk vulnerabilities are found, the score is calculated as 1 x 6 + 2 x 2 + 5 x 1 = 15 (points), the score is greater than 10 and less than 18, so the risk level is "medium-risk".

**（Conclusion of the report, with the following blanks）**