COMPASS SECURITY BLOG

Offensive Defense

Privilege escalation in Windows Domains (2/3)

AUGUST 12, 2019 / THIERRY VIACCOZ / 0 COMMENTS

Generating billions of passwords and trying every possible combination of characters, numbers and symbols isn't funny at all. It costs resources and a lot of time. But a strong password strength policy doesn't help you if your password maintenance policy is weaker than a gummy bear holding a door shut.

Not-so-secure storage device

If you happen to work in a company with more than one employee, you often encounter situations where you have to share sensitive information with someone else. Doing so by talking to each other in private is usually a risk you can take – except if you fear bugging devices under your desk – but as soon as more and more people need access to the same information, it makes sense to store them in a shared place. Unfortunately, there are many places considered safe, that are in fact not. Remember the file servers we accessed in the previous blog post using the permissions of one of your employees? Exactly.

We still have access to the authenticated connection from *penny*, so we can use it to dig a little into the file server we targeted earlier. Using proxychains and SMBMap we can quickly get an overview of the file server and start looking around for accessible drives:

```
# proxychains smbmap -u penny -d WINLAB -H 10.10.10.10
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.13
[+] Finding open SMB ports....
```

```
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  10.10.10.10:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  10.10.10.10:445  ...  OK
[+] Guest SMB session established on 10.10.10.10...
[+] IP: 10.10.10.10:445 Name: unkown
        Disk                                                    Permissions
        ----                                                    -----------
        ADMIN$                                                  NO ACCESS
        AdminShare                                              NO ACCESS
        C$                                                      NO ACCESS
        IPC$                                                    READ ONLY
        Public                                                  READ, WRITE
```

Multiple drives are available, the *Public* drive for read and even write access. This drive might contain some data available to all employees, so we investigate this further using SMBMap:

```
 # proxychains smbmap -u penny -d WINLAB -H 10.10.10.10 -r Public
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.13
[+] Finding open SMB ports....
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  10.10.10.10:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  10.10.10.10:445  ...  OK
[+] Guest SMB session established on 10.10.10.10...
[+] IP: 10.10.10.10:445 Name: unkown
        Disk                                                    Permissions
        ----                                                    -----------
        Public                                                  READ, WRITE
        ./
        dr--r--r--                0 Thu Apr 18 09:43:02 2019    .
        dr--r--r--                0 Thu Apr 18 09:43:02 2019    ..
        fr--r--r--               75 Thu Apr 18 09:40:09 2019    mount.bat
```

Bingo! Batch files to mount drives on login including plaintext or obfuscated credentials are a classic. So next we download all interesting files and look for all kind of information, but most importantly credentials:

```
# proxychains smbmap -u penny -d WINLAB -H 10.10.10.10 --download 'Public\mount.bat'
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.13

[+] Finding open SMB ports....
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  10.10.10.10:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  10.10.10.10:445  ...  OK
[+] Guest SMB session established on 10.10.10.10...
[+] Starting download: Public\mount.bat (75 bytes)
[+] File output to: /tmp/10.10.10.10-Public_mount.bat
```

The file is indeed a batch script mounting a share from the file server. The file contains hardcoded credentials for *mountuser*:

```
# cat 10.10.10.10-Public_mount.bat
@echo off
net use X: \\10.10.10.10\Public /user:mountuser jcBsUxY.JWLGk-8v
```

Executables are just fancy scripts

If you think that you can get rid of your scripts and create a compiled executable that cannot be read using a simple text editor, you probably got a false sense of security. Executables are just fancy scripts. Static data used by your application is readable, even if it’s shattered into thousands of pieces that are dynamically reassembled in one specific moment. It makes things harder because you have to know where to look, and when to look at it, but this is just a question of time until we find out.

With *mountuser*, we have a valid user account for the file server. We can use it directly without having to relay any user connections anymore. Since this user has local access to the file server and can read all the data, we have access to more data that we had with the user *penny* before:

```
 C:\admin>whoami
fileshare\mountuser
C:\admin>dir
 Volume in drive C has no label.
 Volume Serial Number is 3703-67F0

 Directory of C:\admin

03/05/2019  09:28    <DIR>          .
03/05/2019  09:28    <DIR>          ..
03/05/2019  09:32               168 Cleanup.bat
03/05/2019  09:22             5.632 RunAdm.exe
```

```
               2 File(s)          5.800 bytes
               2 Dir(s)  40.339.054.592 bytes free
```

We don’t only have access to additional drives, but also find some administrative maintenance directory with a batch file and an executable. After some investigation, we find that Cleanup.bat deletes files in C:\Temp. Unfortunately, our current user has no access to that directory. That’s when RunAdm.exe comes into play. When executed with Cleanup.bat as argument, it elevates its privileges to a local administrator and deletes all files in C:\Temp:

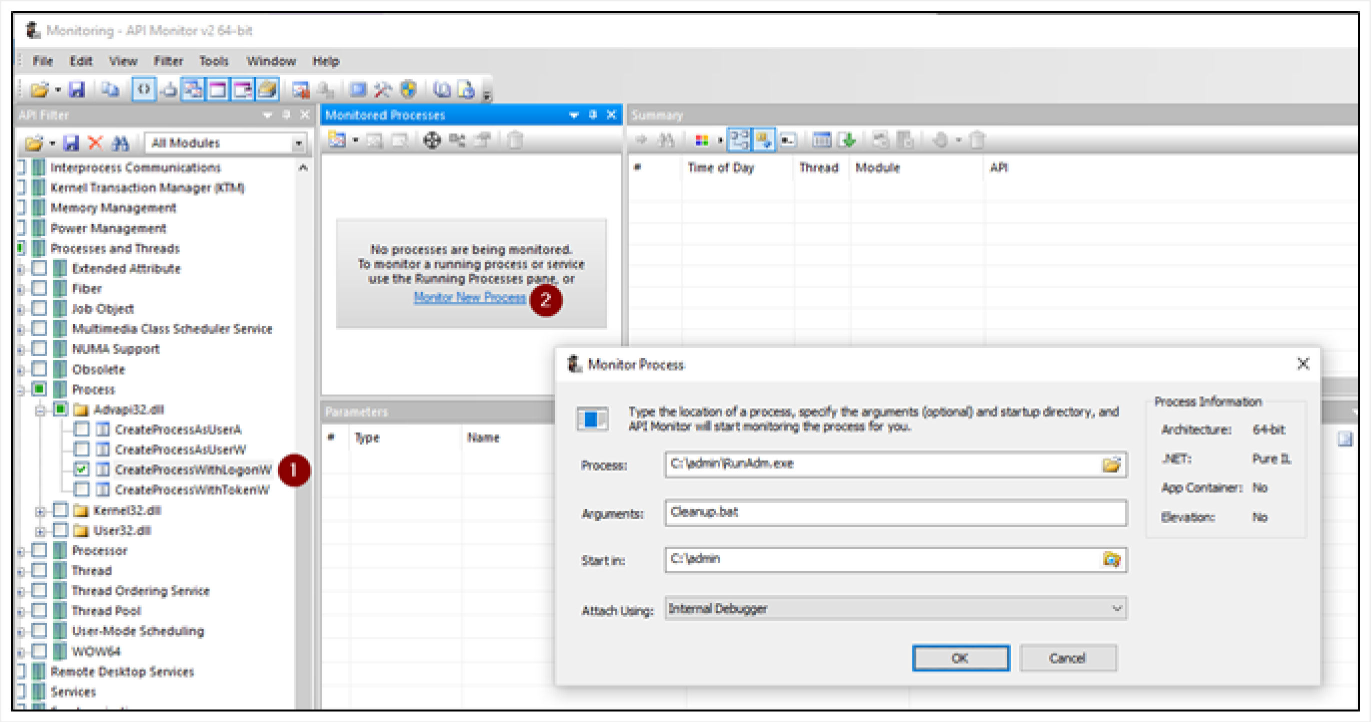
```
 C:\admin>RunAdm.exe Cleanup.bat
Execution of "C:\admin\Cleanup.bat" as the user "admin".
Start of the cleanup

Execution as the following user:
fileshare\admin

Delete all the files in C:\Temp\
```

The executable has the power to execute a script with the privileges of an administrator. The executable is obfuscated – therefore hardly readable for humans, we cannot write to the directory C:\admin, we cannot run the executable outside the directory, nor trick it into executing an executable that can be controlled by us outside of this directory:

```
C:\admin>RunAdm.exe ../pwn/pwn.exe
The file "C:\pwn\pwn.exe" is not in "C:\admin\".
```

There are multiple security measures put in place to make sure the executable only runs that specific script with administrator privileges, not more not less.

However, if it isn’t easily possible to make a static analysis of the executable because of the obfuscation, we can still analyze the executable at runtime. Using API Monitor (https://www.rohitab.com/apimonitor), we can catch the calls of different APIs. By watching *CreateProcessWithLogonW* we get notified whenever it’s called:



No processes are being monitored.
To monitor a running process or service use the Running Processes pane, or
Monitor New Process

Type the location of a process, specify the arguments (optional) and startup directory, and API Monitor will start monitoring the process for you.

Process: C:\admin\RunAdm.exe

Arguments: Cleanup.bat

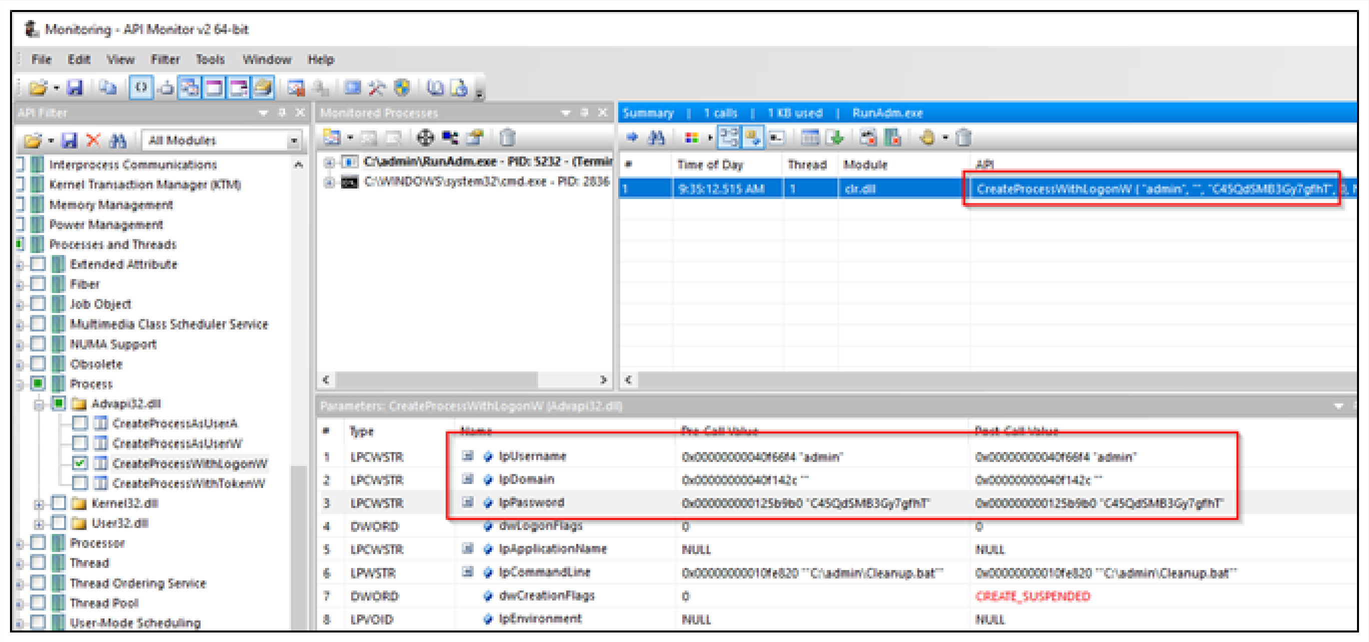
Start in: C:\admin

Attach Using: Internal Debugger

Process Information

Architecture:	64-bit
.NET:	Pure IL
App Container:	No
Elevation:	No

As expected, the method gets called with multiple parameters and – you guessed right – the cleartext credentials of the administrator:



#	Time of Day	Thread	Module	API
1	9:35:12.515 AM	1	clr.dll	CreateProcessWithLogonW ("admin", "", "C45QdSMB3Gy7gfhT", 0,

Parameters: CreateProcessWithLogonW (Advapi32.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	LPCWSTR	lpUsername	0x000000000040f66f4 "admin"	0x000000000040f66f4 "admin"
2	LPCWSTR	lpDomain	0x000000000040f142c ""	0x000000000040f142c ""
3	LPCWSTR	lpPassword	0x00000000000125b9b0 "C45QdSMB3Gy7gfhT"	0x00000000000125b9b0 "C45QdSMB3Gy7gfhT"
4	DWORD	dwLogonFlags	0	0
5	LPCWSTR	lpApplicationName	NULL	NULL
6	LPWSTR	lpCommandLine	0x000000000010fe820 ""C:\admin\Cleanup.bat""	0x000000000010fe820 ""C:\admin\Cleanup.bat""
7	DWORD	dwCreationFlags	0	CREATE_SUSPENDED
8	LPVOID	lpEnvironment	NULL	NULL

All of a sudden, we don’t only have the credentials of a local user, but also of a local administrator – and who knows on what other servers the same administrator account is configured?

Conclusion

Sensitive data has to be stored somewhere, especially when working together with other employees and keeping your network maintainable and long-living. But before you can talk about access rights, you should first understand what data you even have and what the impact of publication would be.

Especially credentials should be classified as very sensitive as they allow to elevate privileges or impersonate others. Giving “usage” rights for credentials without them being readable isn’t possible, therefore it must be avoided, even if it’s really, really hard to retrieve them.

Penetration Test

ACTIVE DIRECTORY · PENETRATION TESTING · PRIVILEGE ESCALATION

PREVIOUS POST
Privilege escalation in Windows Domains (1/3)

NEXT POST
Privilege escalation in Windows Domains (3/3)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment