Malicious Flow

UDURRANI

Malicious Flow

Every malicious code has a specific flow. The flow could be divided in multiple stages. During this flow the payload could use legitimate tools e.g. powershell, wscript, cmd etc. This is mostly done by using the following function calls.

- ☑ CreateProcess()
- ☑ exec()
- ☑ system()
- ☑ ShellExecute()

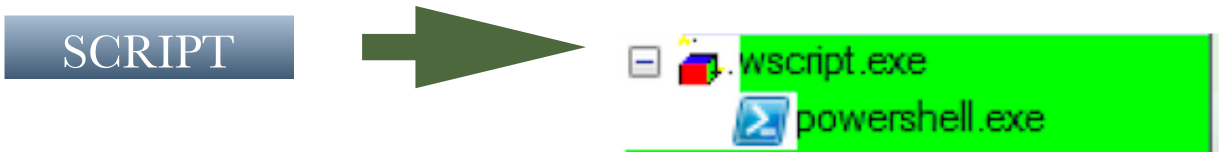
Malware could use either a script or an executable to accomplish this task. In case of a script, the flow or the child process will be initiated by the script engine. In case of an executable, the executable itself will initiate the child process. Let's suppose the malicious code is a VB script with the following line:

```
Set foo = WScript.CreateObject ("WScript.Shell") : foo.run "cmd.exe /c Powershell
```

In the above example, VB script is trying to initiate powershell. Who spawns the powershell in this case? VB script?

Not really!

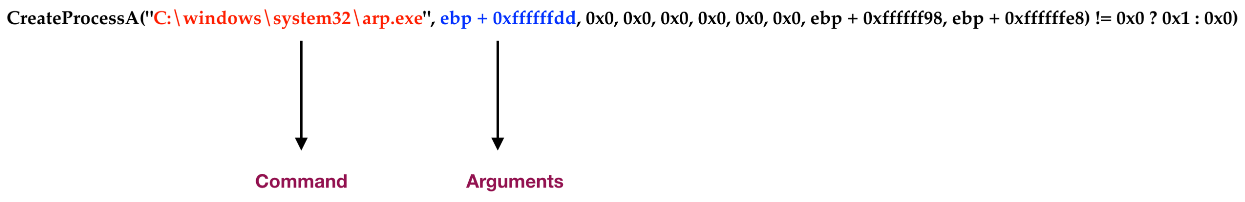
VB script is just a data file that is processed by WSCRIPT engine. Thats why WSCRIPT.exe will be responsible for this flow i.e. **WSCRIPT.exe** —> **POWERSHELL.exe**. The VB script is interpreted by the framework. So **foo.run** will eventually call **CreateProcess**()



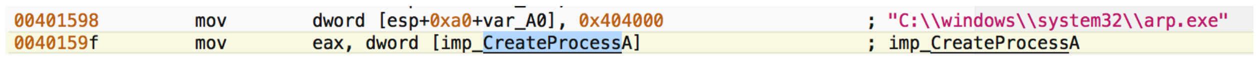
What about the executable?

When it comes to an executable, it will initiate the flow directly. E.g a malicious executable is trying to run **arp -a** command to look at the arp table. The code can use **CreateProcess**() directly to run the command. Let's look at this flow.

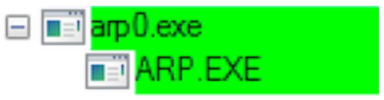
The executable will call the following function.



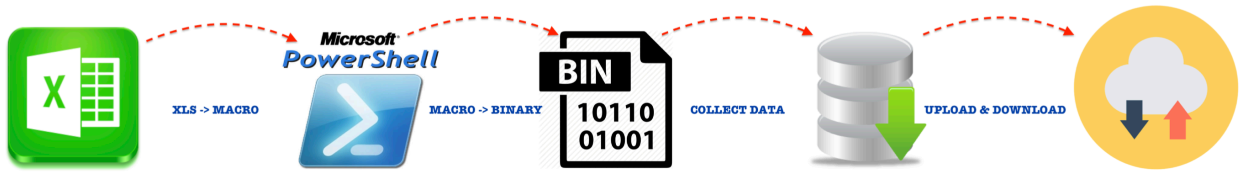
```
CreateProcessA("C:\windows\system32\arp.exe", ebp + 0xffffffdd, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, ebp + 0xffffff98, ebp + 0xffffffe8) != 0x0 ? 0x1 : 0x0)
```



```
00401598        mov       dword [esp+0xa0+var_A0], 0x404000                ; "C:\\windows\\system32\\arp.exe"
0040159f        mov       eax, dword [imp_CreateProcessA]                  ; imp_CreateProcessA
```

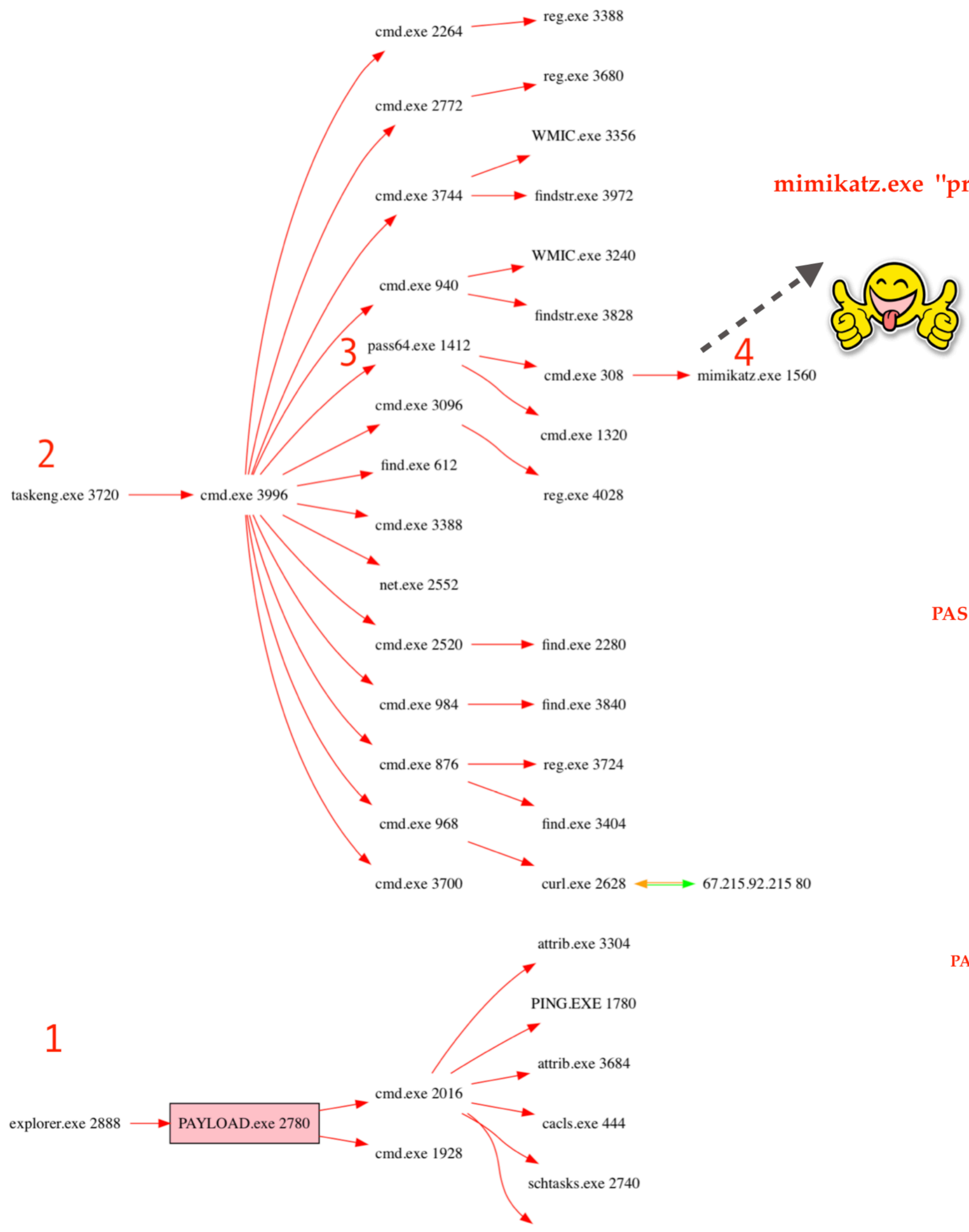


Its very important to understand the malicious flow. Remember, not all the pieces of this flow are malicious. Of course, the intent is evil but its using legitimate functionality to get there. E.g. if we look at the following flow. Excel and powershell are the helper stages. The real bad guy is a binary dropped or downloaded in the 3rd stage



Let’s look at some more flows:

Credential theft flow:



mimikatz.exe "privilege::debug" "log" "sekurlsa::logonpasswords" "exit"

PASS64.exe is a packed file that contains the following

```
pass64
├── mimidrv.sys
├── mimikatz.exe
├── mimilib.dll
└── pass.bat
```

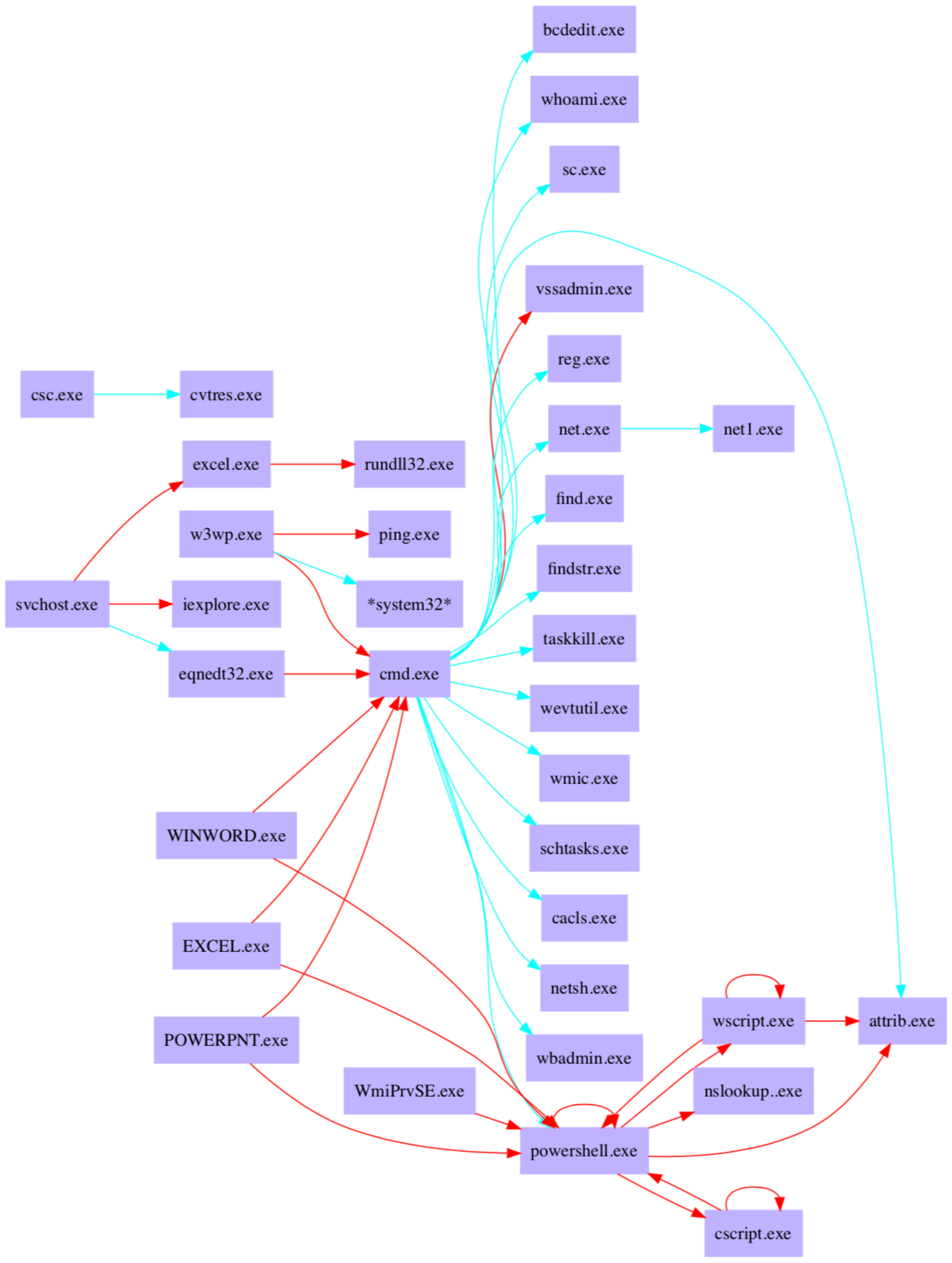
PASS.bat runs the mimkatz command and later delete all the files

Flow for ServerDestroyer payload

FLOW: **http://udurrani.com/0fff/server_ransomware_flow.pdf**

COMMAND(S) **http://udurrani.com/0fff/server_ransomware.pdf**

Here are some of the flows that must be either prevented or detected. The flows with red arrows can be prevented. Flows with cyan arrows can run in notification mode.



In some cases, the attacker can write extra code and not use a **system**() or **createProcess**() functionality. This technique is rear, but if used, can bypass multiple security layers.

Here is a binary that runs arp -a command by using arp.exe command (Password: foo)

http://udurrani.com/0fff/a1.zip

Here is a binary that runs the same functionality without using the arp.exe command (Password: foo)

http://udurrani.com/0fff/a2.zip

Some links that shows malicious flows.

GREENBUG:	**http://udurrani.com/0fff/gbvt.pdf**
MACRO (RAT):	**http://udurrani.com/0fff/msword_to_backdoor.pdf**
EMOTET:	**http://udurrani.com/0fff/EMOTET_OBFUSCATION.pdf**
WANACRYPT:	**http://udurrani.com/0fff/all1.pdf**