

Relocation Bonus

Attacking the Windows Loader Makes Analysts Switch Careers

Introduction

Introduction

Nick Cano

Introduction

Nick Cano

- 25 years old

Introduction

Nick Cano

- 25 years old
- Senior Security Architect at Cylance

Introduction

Nick Cano

- 25 years old
- Senior Security Architect at Cylance
- Author of *Game Hacking: Developing Autonomous Bots for Online Games*

Introduction

Nick Cano

- 25 years old
- Senior Security Architect at Cylance
- Author of *Game Hacking: Developing Autonomous Bots for Online Games*
- Pluralsight Instructor, *Modern C++ Secure Coding Practices: Const Correctness*

Introduction

Nick Cano

- 25 years old
- Senior Security Architect at Cylance
- Author of *Game Hacking: Developing Autonomous Bots for Online Games*
- Pluralsight Instructor, *Modern C++ Secure Coding Practices: Const Correctness*

Relocation Bonus

Introduction

Nick Cano

- 25 years old
- Senior Security Architect at Cylance
- Author of *Game Hacking: Developing Autonomous Bots for Online Games*
- Pluralsight Instructor, *Modern C++ Secure Coding Practices: Const Correctness*

Relocation Bonus

- A look into the Windows Portable Executable (PE) header and how it can be weaponized to *destroy* parsers, disassemblers, and other tools

Introduction

Nick Cano

- 25 years old
- Senior Security Architect at Cylance
- Author of *Game Hacking: Developing Autonomous Bots for Online Games*
- Pluralsight Instructor, *Modern C++ Secure Coding Practices: Const Correctness*

Relocation Bonus

- A look into the Windows Portable Executable (PE) header and how it can be weaponized to *destroy* parsers, disassemblers, and other tools
- A PE rebuilder that takes any 32bit PE then obfuscates and rebuilds it using the attack

Why Attack Relocations?

Why Attack Relocations?

The screenshot shows a debugger interface with assembly code on the left and a message window on the right.

Assembly Code:

```
00219E05 E8 8692F2FF CALL Tibia.00143010
00219E0A 68 452A6900 PUSH Tibia.00692448
00219E0F 8D65 40FFFFFF LER EDX,DWORD PTR SS:[EBP-C0]
00219E15 8B 3A2D2B00 PUSH EDX
00219E19 CC INT3
00219E1C CC INT3
00219E1D CC INT3
00219E1E CC INT3
00219E1F CC INT3
00219E20 55 PUSH EBP
00219E21 8BEC MOV EBP,ESP
00219E22 6A F0 PUSH F0
00219E25 8D45 F4 PUSH EAX
00219E26 64A1 00000000 MOU EDX,DWORD PTR FS:[0]
00219E29 50 PUSH EAX
00219E2A 81EC 00010000 SUB ESP,100
00219E2B A1 00000000 MOU EDX,DWORD PTR DS:[6983800]
00219E2C 83C4 XOR EDX,EDX
00219E2E 8945 EC MOU DWORD PTR SS:[EBP-14],EDX
00219E2F 53 PUSH EBX
00219E30 8BEC MOV EBP,ESP
00219E31 53 PUSH EBX
00219E32 50 PUSH EDI
00219E33 57 PUSH EDI
00219E34 50 PUSH EAX
00219E35 8D45 F4 LER EDX,DWORD PTR SS:[EBP-C1]
00219E38 64A1 00000000 MOU EDX,DWORD PTR FS:[0],EDX
00219E39 8B 3A2D2B00 MOU EDX,DWORD PTR SS:[EBP-10],ESP
00219E3A 8B69 MOU EDI,ECX
00219E3D C745 FC 00000000 MOU ESI,Tibia.00858029
00219E40 BE 29808598 MOU ESI,Tibia.00858029
00219E43 8D65 80FFFFFF MOU DWORD PTR SS:[EBP-F8],ESI
00219E44 83C8 08 CMP EDI,8
00219E45 80F87 C0000000 JRP Tibia.00219E04
00219EE5 FF2480 F0012100 JRP DWORD PTR DS:[EDI*4+21A1F0]
00219E55 BE A0248700 MOU ESI,Tibia.00872440
00219E56 8B 68FFFFFF MOU EDX,DWORD PTR DS:[EBP-F8],ESI
00219E59 8B 3A2D2B00 MOU EDX,DWORD PTR SS:[EBP-10],ESP
00219E5A EB 10 JRP SHORT Tibia.00219E17
00219E67 BE 80238700 MOU ESI,Tibia.00872380
00219E6C 8965 08FFFFFF MOU DWORD PTR SS:[EBP-F8],ESI
00219E6D 8B4248700 MOU ESI,Tibia.00872494
00219E70 50 PUSH EAX
00219E71 8D40 BC LER ECX,DWORD PTR SS:[EBP-44]
00219E72 E8 400BF2FF CALL Tibia.00130A60
00219E75 8D45 FC 01 MOU EITE,DWORD PTR SS:[EBP-4],1
00219E76 83C8 08 CMP EDI,8
00219E77 74 0A JE SHORT Tibia.00219E33
00219E79 89FF 01 CMW EDI,1
00219E7C B9 5C7F7000 MOV ECX,Tibia.00707F5C
00219E7D 8D45 00 MOU EIP,DWORD PTR DS:[EBP-10]
00219E7E 89 847F7000 MOU ECX,Tibia.00707F54
00219E7F 80E1 MOU EDX,DWORD PTR DS:[ECX]
00219E80 8945 DB MOU DWORD PTR SS:[EBP-29],EDX
00219E81 8945 DC MOU EDX,DWORD PTR DS:[EBP-28],ECX
00219E82 8945 DC MOU ECX,DWORD PTR DS:[EBP-41],EDX
00219E83 8B41 08 MOU EDX,DWORD PTR DS:[ECX+8]
00219E84 8945 E0 MOU DWORD PTR SS:[EBP-20],EDX
00219E85 8B41 0C MOU EDX,DWORD PTR DS:[ECX+C]
00219E86 8945 E4 MOU ECX,DWORD PTR DS:[EBP-11],ECX
00219E87 8B41 10 MOU EDX,DWORD PTR DS:[EBP-10]
```

Message Window (Right):

- ASCII "You are using an outdated client which is no longer supp
- ASCII "Error"
- ASCII "You are using an outdated client which is no longer supp
- ASCII "Information"

Why Attack Relocations?



```
00219E05 E8 8692F2FF CALL Tibia.00143110
00219E8A 68 482A6900 PUSH Tibia.00692448
00219E8F 8D85 40FFFFFF LER EDX,DWORD PTR SS:[EBP-C0]
00219E95 E9 00000000 PUSH EBX
00219E98 CC 3A2D2B00 CALL Tibia.004CCBD5
00219E9C CC INT3
00219E9D CC INT3
00219E9E CC INT3
00219E9F CC INT3
00219EA0 55 PUSH EBP
00219EA1 8BEC MOV EBP,ESP
00219EA2 6A FF PUSH ECX
00219EA3 8D45 F4 PUSH EBP
00219EA4 64A3 00000000 MOU EDX,DWORD PTR FS:[0]
00219EA8 50 PUSH EAX
00219E81 81EC 00010000 SUB ESP,100
00219E82 A1 00000000 MOU EDX,DWORD PTR DS:[6983800]
00219E86 8945 EC MOU DIWORD PTR SS:[EBP-14],EAX
00219E8C 53 PUSH EBX
00219E8D 56 PUSH ECX
00219E8E 57 PUSH EDI
00219E8F 58 PUSH EAX
00219E95 8D45 F4 LER EDX,DWORD PTR SS:[EBP-C1]
00219E98 64A3 00000000 MOU DIWORD PTR FS:[0],EDX
00219E99 8965 F0 MOU EDX,DWORD PTR SS:[EBP-10],ESP
00219E9A 8B99 EDI,ECX
00219E9D C745 FC 00000000 MOU DIWORD PTR SS:[EBP-41],0
00219E9A BE 29808500 MOU ESI,Tibia.00858029
00219E9F 8D85 08FFFFFF MOU DIWORD PTR SS:[EBP-F8],ESI
00219E98 8945 EC CMP EDI,3
00219E88 0F87 C0000000 JRP Tibia.00219F84
00219EE8 FF2480 F0012100 JRP DIWORD PTR DS:[EDI*4+21A1F0]
00219E95 BE A0248700 MOU ESI,Tibia.00872440
00219E98 8B99 68FFFFFF MOU EDX,Tibia.00872448
00219E99 88 88252700 MOU EAX,Tibia.00872598
00219E9A EB 10 JMP SHORT Tibia.00219F17
00219E97 BE 80238700 MOU ESI,Tibia.00872380
00219F0C 8965 F0 MOU DIWORD PTR SS:[EBP-F8],ESI
00219F01 8945 EC MOU ECX,Tibia.00872494
00219F17 59 PUSH EBX
00219F18 8D40 BC LEA ECX,DWORD PTR SS:[EBP-44]
00219F1B E8 40082FF2 CALL Tibia.00143110
00219F28 8D45 FC 01 MOU DIWORD PTR SS:[EBP-41],1
00219F29 8945 DB MOU EDI,1
00219F2A 8945 DC MOU DIWORD PTR DS:[EBP-14]
00219F43 8941 08 MOU EDX,DWORD PTR DS:[EBP-8]
00219F46 8945 EC MOU DIWORD PTR DS:[EBP-10]
00219F49 8941 0C MOU EDX,DWORD PTR DS:[EBP-12]
00219F4C 8945 E4 MOU DIWORD PTR DS:[EBP-14]
00219F53 8941 10 MOU EDX,DWORD PTR DS:[EBP-16]
```

```
00219E05 E8 8692F2FF CALL Tibia.00143110
00219E8A 68 482A6900 PUSH Tibia.00692448
00219E8F 8D85 40FFFFFF LER EDX,DWORD PTR SS:[EBP-C0]
00219E95 E9 00000000 PUSH EBX
00219E98 CC 3A2D2B00 CALL Tibia.004CCBD5
00219E9C CC INT3
00219E9D CC INT3
00219E9E CC INT3
00219E9F CC INT3
00219EA0 55 PUSH EBP
00219EA1 8BEC MOV EBP,ESP
00219EA2 6A FF PUSH ECX
00219EA3 8D45 F4 PUSH EBP
00219EA4 64A3 00000000 MOU EDX,DWORD PTR FS:[0]
00219EA8 50 PUSH EAX
00219E81 81EC 00010000 SUB ESP,100
00219E82 A1 00000000 MOU EDX,DWORD PTR DS:[6983800]
00219E86 8945 EC MOU DIWORD PTR SS:[EBP-14],EAX
00219E8C 53 PUSH EBX
00219E8D 56 PUSH ECX
00219E8E 57 PUSH EDI
00219E8F 58 PUSH EAX
00219E95 8D45 F4 LER EDX,DWORD PTR SS:[EBP-C1]
00219E98 64A3 00000000 MOU DIWORD PTR FS:[0],EDX
00219E99 8965 F0 MOU EDX,DWORD PTR SS:[EBP-10],ESP
00219E9A 8B99 EDI,ECX
00219E9D C745 FC 00000000 MOU DIWORD PTR SS:[EBP-41],0
00219E9A BE 29808500 MOU ESI,Tibia.00858029
00219E9F 8D85 08FFFFFF MOU DIWORD PTR SS:[EBP-F8],ESI
00219E98 8945 EC CMP EDI,3
00219E88 0F87 C0000000 JRP Tibia.00219F84
00219EE8 FF2480 F0012100 JRP DIWORD PTR DS:[EDI*4+21A1F0]
00219E95 BE 00248700 MOU ESI,Tibia.00872440
00219E98 8965 08FFFFFF MOU DIWORD PTR SS:[EBP-F8],ESI
00219E99 88 88252700 MOU EAX,Tibia.00872598
00219E9A EB 10 JMP SHORT Tibia.00219F17
00219F07 BE 80238700 MOU ESI,Tibia.00872380
00219F0C 8965 08FFFFFF MOU DIWORD PTR SS:[EBP-F8],ESI
00219F12 B8 3A2D2B00 MOU EDX,DWORD PTR DS:[EBP-24]
00219F17 50 PUSH EBX
00219F18 8D40 BC LEA ECX,DWORD PTR SS:[EBP-44]
00219F1B E8 00082FF2 CALL Tibia.00143110
00219F28 8D45 FC 01 MOU BYTE PTR SS:[EBP-41],1
00219F29 89FF 03 CMP EDI,3
00219F27 74 0A JE SHORT Tibia.00219F33
00219F29 83FF 01 CMP EDI,1
00219F2C 8D40 BC LEA ECX,DWORD PTR DS:[EBP-28]
00219F33 75 0A MOU ECX,Tibia.00707F5C
00219F38 8945 EC MOU EDX,DWORD PTR DS:[EBX]
00219F40 8945 D8 MOU EDX,DWORD PTR DS:[EBP-28],EDX
00219F40 8941 04 MOU EDX,DWORD PTR DS:[EBX+4]
```

ASCII "You are using an outdated client which is no longer supp
ASCII "Error"
ASCII "You are using an outdated client which is no longer supp
ASCII "Information"

Why Attack Relocations?

```
00219E65 E8 692F2FF CALL Tibia.00143110
00219E6A 48 8D5690B PUSH EDI,EDOWDWORD PTR SS:[EBP-C0]
00219E69 8D93 40FFFFFF LEA EDI,EDOWDWORD PTR DS:[EBP-0]
00219E95 E9CALL Tibia.004CCB05
00219E96 E8 3A2D2B000 CALL Tibia.004CCB05
00219E97 CC INT3
00219E98 CC INT3
00219E99 CC INT3
00219E9A CC INT3
00219E9B CC INT3
00219E9C E8 PUSH EBP
00219E9D MOU EBP,ESP
00219E9E 5A FF PUSH -1
00219E9F 48 8D3540000000 PUSH Tibia.006435FA
00219EA1 54 A1 00000000 MOU EDI,EDOWDWORD PTR FS:[0]
00219EA0 E9CALL Tibia.004CCB05
00219EB1 81EC 00010000 SUB ESP,100
00219EB2 A1 80836980 MOU EDX,EDOWDWORD PTR DS:[6983880]
00219EB3 8D93 40FFFFFF XOR EDX,EDX
00219EB4 8D93 40FFFFFF MOU EDWDWORD PTR SS:[EBP-141],EAX
00219EC1 53 PUSH EBX
00219EC2 56 PUSH ESI
00219EC3 57 PUSH EDI
00219EC4 58 PUSH EDX
00219EC5 8D45 F4 LER EDX,EDOWDWORD PTR SS:[EBP-C]
00219EC8 54 A3 00000000 MOU EDWDWORD PTR FS:[0],EAX
00219EC9 8D93 F0 MOU EDWDWORD PTR SS:[EBP-10],ESP
00219ED0 B8 01 00000000 MOU EDI,ECX
00219ED3 C745 FC 00000000 MOU EDWDWORD PTR SS:[EBP-41,0
00219ED4 BE 29085500 MOU ESI,Tibia.00858029
00219ED5 B9 0E 00FFFFFF MOU ECX,EDOWDWORD PTR SS:[EBP-F81],ESI
00219ED6 C9 FIDI_8
```

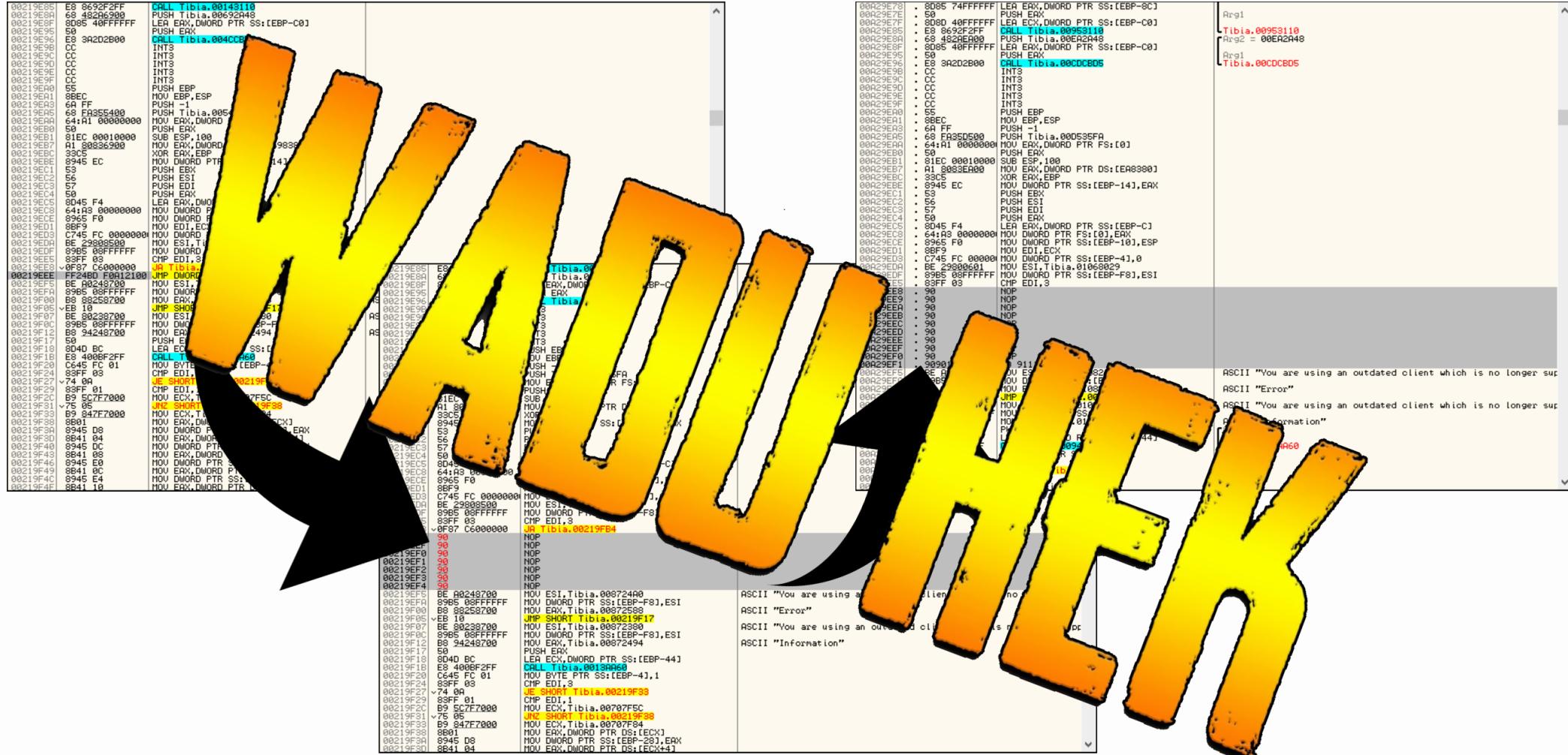
```
0002E979 805 74FFFFF LER EXK,DMDR PTR SS:[EBP-8C]
0002E97A 80D 40FFFFFF LER EXK,DMDR PTR SS:[EBP-C8]
0002E97B 80D 40FFFFFF LER EXK,DMDR PTR SS:[EBP-C8]
0002E97C 80D 40FFFFFF CALL Tibia._00053110
0002E97D 68 E892FC0000 PUSH Tibia._00082448
0002E97E 805 40FFFFFF LER EXK,DMDR PTR SS:[EBP-C8]
0002E97F 59 F0 POP ECX
0002E97G EB 00 JMP ECX
0002E97H E8 3A202000 CALL Tibia._000DCB05
0002E97I CC INT3
0002E97J CC INT3
0002E97K CC INT3
0002E97L CC INT3
0002E97M CC INT3
0002E97N CC INT3
0002E97O 83 EC PUSH EBX
0002E97P 8BEC MOU EDX,EBP
0002E97Q 6A FF PUSH -1
0002E97R 6A F9 FA35D500 PUSH Tibia._000535FA
0002E97S 6A F1 00000000 MOU EDX,DMDR PTR FS:[0]
0002E97T 50 F0 POP AL
0002E97U B1 EC 00010000 MOU EDX,SPL,100
0002E97V A1 8083CEA00 MOU EXK,DMDR PTR DS:[EA8380]
0002E97W 805 40FFFFFF LER EXK,DMDR PTR SS:[EBP-141]
0002E97X 8045 EC MOU EDX,EBX
0002E97Y 53 F0 PUSH ECX
0002E97Z 66 F9 MOU EDX,ESI
0002E97A 50 F0 POP EDI
0002E97B 58 F0 PUSH EDX
0002E97C 8045 F4 LER EXK,DMDR PTR SS:[EBP-C]
0002E97D 6A F1 00000000 MOU EDX,DMDR PTR FS:[0],ERX
0002E97E 6A F9 00000000 MOU EDX,DMDR PTR SS:[EBP-141],ESP
0002E97F 8BEC MOU EDI,ECX
0002E97G C745 FC 00000 MOU DMDR PTR SS:[EBP-41,0]
0002E97H BE 23000061 MOU EDI,ESI_01068929
0002E97I 805 40FFFFFF LER EXK,DMDR PTR SS:[EBP-F8],ESI
0002E97J 50 F0 POP EDI
0002E97K 3BEE 03 CDH MOU EDI,ESI
```

an outdated client which is no longer supp
an outdated client which is no longer supp

```
gl  
bia.00953110  
g2 = 00EA2A48  
gl  
bia.00CDCB05  
  
CII "You are using an outdated client which is no longer sup  
CII "Error"  
CII "You are using an outdated client which is no longer sup  
CII "Information"  
gl  
bia.0094AA60
```

As a result, the first step in the process of creating a new model is to identify the relevant variables that are likely to influence the outcome. This can be done through a variety of methods, such as reviewing existing literature, conducting surveys or interviews, and consulting with experts in the field. Once the variables have been identified, they can be used to develop a conceptual model that describes the relationships between them. This model can then be tested using statistical methods to determine if it accurately represents the real-world data. If the model is found to be accurate, it can be used to make predictions about future events or to inform policy decisions. However, if the model is found to be inaccurate, it may need to be revised or replaced.

Why Attack Relocations?

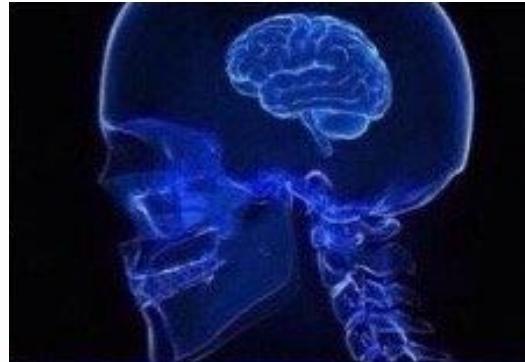


Relocation Bonus - How Did I Get Here?

Why Attack Relocations?

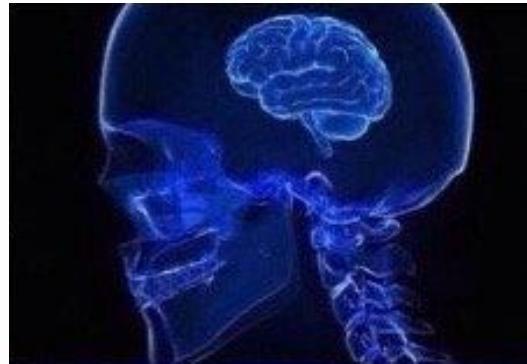
Why Attack Relocations?

its broken for no
reason

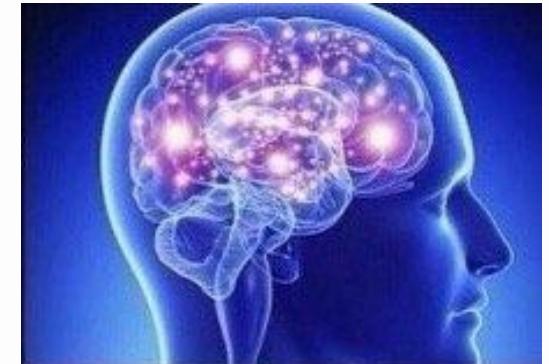


Why Attack Relocations?

its broken for no reason

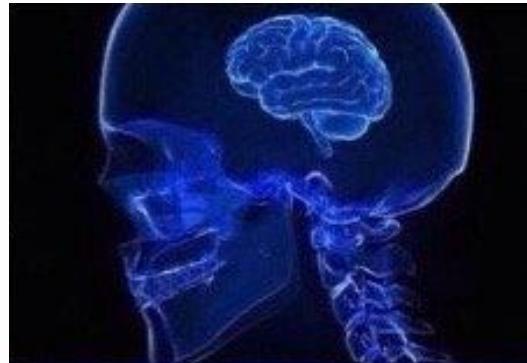


relocations corrupted the patched code



Why Attack Relocations?

its broken for no reason



relocations corrupted the patched code

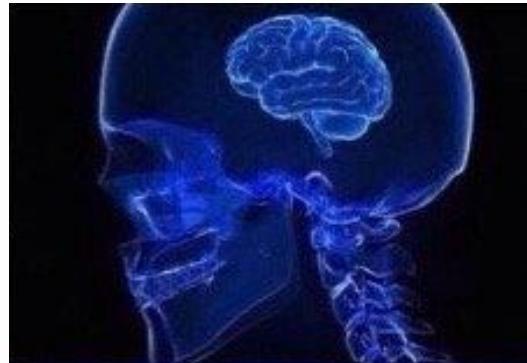


don't patch code that is relocated



Why Attack Relocations?

its broken for no reason



relocations corrupted the patched code



don't patch code that is relocated



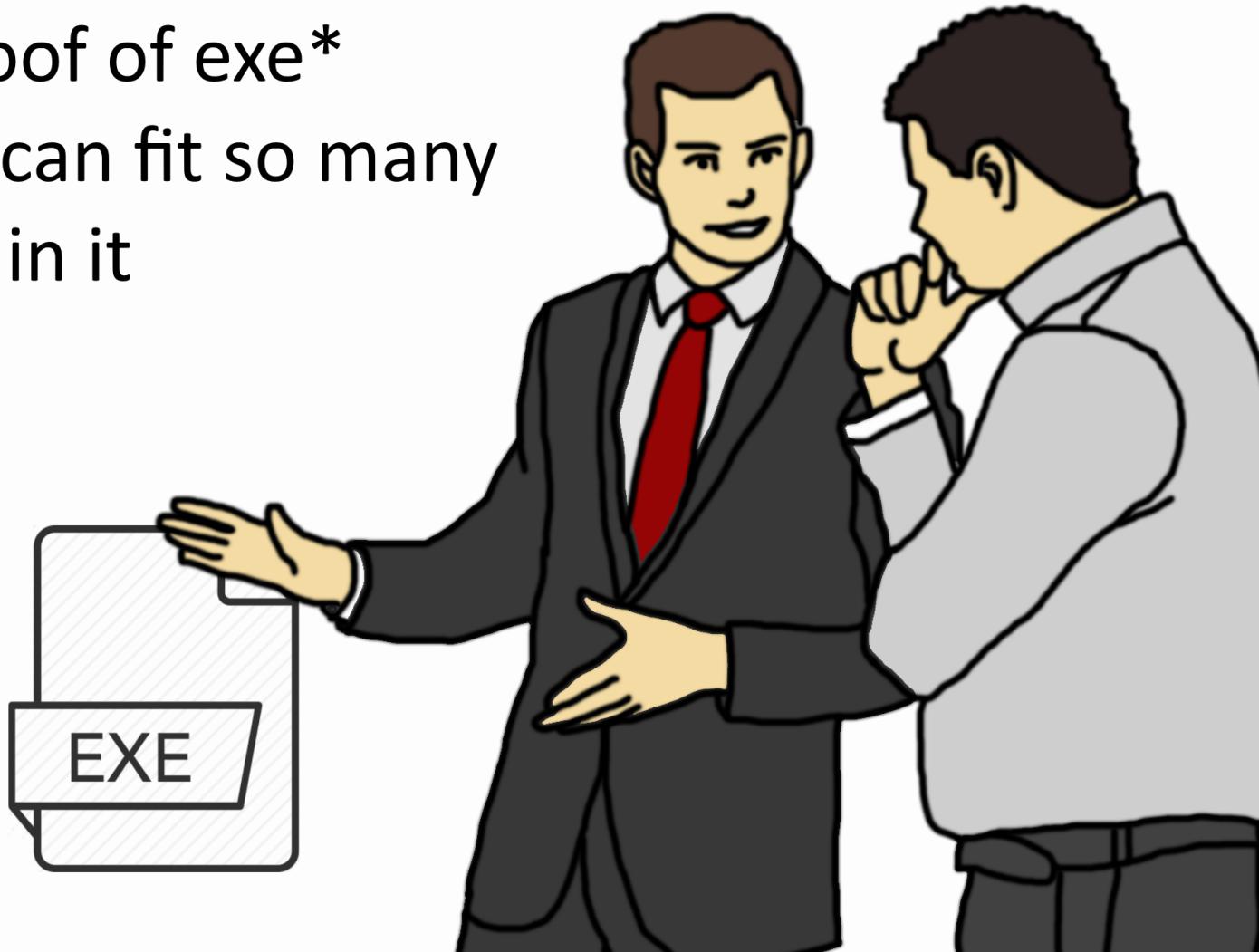
relocations can be weaponized to hide my arsenal in the bowels of the machine



Mission Statement

Mission Statement

Me: *slaps roof of exe*
this bad boy can fit so many
reloc entries in it



What Are Relocations?

What Are Relocations?

Relocations exist to enable dynamic mapping, specifically ASLR

What Are Relocations?

Relocations exist to enable dynamic mapping, specifically ASLR

0x012E0000 Base

0133A00C	. AF9F3301	DD Example.01339FAF
0133A010	. D29F3301	DD Example.01339FD2
0133A014	. 00A03301	DD Example.0133A000
0133A018	. 00A03301	DD Example.0133A000
0133A01C	. 00A03301	DD Example.0133A000
0133A020	. 00A03301	DD Example.0133A000
0133A024	. 00A03301	DD Example.0133A000
0133A028	. 00A03301	DD Example.0133A000
0133A02C	. 00A03301	DD Example.0133A000
0133A030	. 00A03301	DD Example.0133A000
0133A034	. 00A03301	DD Example.0133A000
0133A038	. 00A03301	DD Example.0133A000
0133A03C	. 00A03301	DD Example.0133A000
0133A040	. 00A03301	DD Example.0133A000
0133A044	. 00A03301	DD Example.0133A000
0133A048	. 00A03301	DD Example.0133A000
0133A04C	. 00A03301	DD Example.0133A000
0133A050	. 00A03301	DD Example.0133A000
0133A054	. 00A03301	DD Example.0133A000
0133A058	. 00A03301	DD Example.0133A000

0x01120000 Base

0117A00C	. AF9F1701	DD Example.01179FAF
0117A010	. D29F1701	DD Example.01179FD2
0117A014	. 00A01701	DD Example.0117A000
0117A018	. 00A01701	DD Example.0117A000
0117A01C	. 00A01701	DD Example.0117A000
0117A020	. 00A01701	DD Example.0117A000
0117A024	. 00A01701	DD Example.0117A000
0117A028	. 00A01701	DD Example.0117A000
0117A02C	. 00A01701	DD Example.0117A000
0117A030	. 00A01701	DD Example.0117A000
0117A034	. 00A01701	DD Example.0117A000
0117A038	. 00A01701	DD Example.0117A000
0117A03C	. 00A01701	DD Example.0117A000
0117A040	. 00A01701	DD Example.0117A000
0117A044	. 00A01701	DD Example.0117A000
0117A048	. 00A01701	DD Example.0117A000
0117A04C	. 00A01701	DD Example.0117A000
0117A050	. 00A01701	DD Example.0117A000
0117A054	. 00A01701	DD Example.0117A000
0117A058	. 00A01701	DD Example.0117A000

What Are Relocations?

Relocations exist to enable dynamic mapping, specifically ASLR

0x012E0000 Base

0133A00C	. AF9F3301	DD Example.01339FAF
0133A010	. D29F3301	DD Example.01339FD2
0133A014	. 00A03301	DD Example.0133A000
0133A018	. 00A03301	DD Example.0133A000
0133A01C	. 00A03301	DD Example.0133A000
0133A020	. 00A03301	DD Example.0133A000
0133A024	. 00A03301	DD Example.0133A000
0133A028	. 00A03301	DD Example.0133A000
0133A02C	. 00A03301	DD Example.0133A000
0133A030	. 00A03301	DD Example.0133A000
0133A034	. 00A03301	DD Example.0133A000
0133A038	. 00A03301	DD Example.0133A000
0133A03C	. 00A03301	DD Example.0133A000
0133A040	. 00A03301	DD Example.0133A000
0133A044	. 00A03301	DD Example.0133A000
0133A048	. 00A03301	DD Example.0133A000
0133A04C	. 00A03301	DD Example.0133A000
0133A050	. 00A03301	DD Example.0133A000
0133A054	. 00A03301	DD Example.0133A000
0133A058	. 00A03301	DD Example.0133A000

0x01120000 Base

0117A00C	. AF9F1701	DD Example.01179FAF
0117A010	. D29F1701	DD Example.01179FD2
0117A014	. 00A01701	DD Example.0117A000
0117A018	. 00A01701	DD Example.0117A000
0117A01C	. 00A01701	DD Example.0117A000
0117A020	. 00A01701	DD Example.0117A000
0117A024	. 00A01701	DD Example.0117A000
0117A028	. 00A01701	DD Example.0117A000
0117A02C	. 00A01701	DD Example.0117A000
0117A030	. 00A01701	DD Example.0117A000
0117A034	. 00A01701	DD Example.0117A000
0117A038	. 00A01701	DD Example.0117A000
0117A03C	. 00A01701	DD Example.0117A000
0117A040	. 00A01701	DD Example.0117A000
0117A044	. 00A01701	DD Example.0117A000
0117A048	. 00A01701	DD Example.0117A000
0117A04C	. 00A01701	DD Example.0117A000
0117A050	. 00A01701	DD Example.0117A000
0117A054	. 00A01701	DD Example.0117A000
0117A058	. 00A01701	DD Example.0117A000

What Are Relocations?

Relocations exist to enable dynamic mapping, specifically ASLR

0x012E0000 Base

0133A00C	. AF9F3301	DD Example.01339FAF
0133A010	. D29F3301	DD Example.01339FD2
0133A014	. 00A03301	DD Example.0133A000
0133A018	. 00A03301	DD Example.0133A000
0133A01C	. 00A03301	DD Example.0133A000
0133A020	. 00A03301	DD Example.0133A000
0133A024	. 00A03301	DD Example.0133A000
0133A028	. 00A03301	DD Example.0133A000
0133A02C	. 00A03301	DD Example.0133A000
0133A030	. 00A03301	DD Example.0133A000
0133A034	. 00A03301	DD Example.0133A000
0133A038	. 00A03301	DD Example.0133A000
0133A03C	. 00A03301	DD Example.0133A000
0133A040	. 00A03301	DD Example.0133A000
0133A044	. 00A03301	DD Example.0133A000
0133A048	. 00A03301	DD Example.0133A000
0133A04C	. 00A03301	DD Example.0133A000
0133A050	. 00A03301	DD Example.0133A000
0133A054	. 00A03301	DD Example.0133A000
0133A058	. 00A03301	DD Example.0133A000

0x01120000 Base

0117A00C	. AF9F1701	DD Example.01179FAF
0117A010	. D29F1701	DD Example.01179FD2
0117A014	. 00A01701	DD Example.0117A000
0117A018	. 00A01701	DD Example.0117A000
0117A01C	. 00A01701	DD Example.0117A000
0117A020	. 00A01701	DD Example.0117A000
0117A024	. 00A01701	DD Example.0117A000
0117A028	. 00A01701	DD Example.0117A000
0117A02C	. 00A01701	DD Example.0117A000
0117A030	. 00A01701	DD Example.0117A000
0117A034	. 00A01701	DD Example.0117A000
0117A038	. 00A01701	DD Example.0117A000
0117A03C	. 00A01701	DD Example.0117A000
0117A040	. 00A01701	DD Example.0117A000
0117A044	. 00A01701	DD Example.0117A000
0117A048	. 00A01701	DD Example.0117A000
0117A04C	. 00A01701	DD Example.0117A000
0117A050	. 00A01701	DD Example.0117A000
0117A054	. 00A01701	DD Example.0117A000
0117A058	. 00A01701	DD Example.0117A000

What Are Relocations?

Relocations exist to enable dynamic mapping, specifically ASLR

0x012E0000 Base

0133A00C	. AF9F3801	DD Example.01339FAF
0133A010	. D29F3301	DD Example.01339FD2
0133A014	. 00A03301	DD Example.0133A000
0133A018	. 00A03301	DD Example.0133A000
0133A01C	. 00A03301	DD Example.0133A000
0133A020	. 00A03301	DD Example.0133A000
0133A024	. 00A03301	DD Example.0133A000
0133A028	. 00A03301	DD Example.0133A000
0133A02C	. 00A03301	DD Example.0133A000
0133A030	. 00A03301	DD Example.0133A000
0133A034	. 00A03301	DD Example.0133A000
0133A038	. 00A03301	DD Example.0133A000
0133A03C	. 00A03301	DD Example.0133A000
0133A040	. 00A03301	DD Example.0133A000
0133A044	. 00A03301	DD Example.0133A000
0133A048	. 00A03301	DD Example.0133A000
0133A04C	. 00A03301	DD Example.0133A000
0133A050	. 00A03301	DD Example.0133A000
0133A054	. 00A03301	DD Example.0133A000
0133A058	. 00A03301	DD Example.0133A000

0x01120000 Base

0117A00C	. AF9F1701	DD Example.01179FAF
0117A010	. D29F1701	DD Example.01179FD2
0117A014	. 00A01701	DD Example.0117A000
0117A018	. 00A01701	DD Example.0117A000
0117A01C	. 00A01701	DD Example.0117A000
0117A020	. 00A01701	DD Example.0117A000
0117A024	. 00A01701	DD Example.0117A000
0117A028	. 00A01701	DD Example.0117A000
0117A02C	. 00A01701	DD Example.0117A000
0117A030	. 00A01701	DD Example.0117A000
0117A034	. 00A01701	DD Example.0117A000
0117A038	. 00A01701	DD Example.0117A000
0117A03C	. 00A01701	DD Example.0117A000
0117A040	. 00A01701	DD Example.0117A000
0117A044	. 00A01701	DD Example.0117A000
0117A048	. 00A01701	DD Example.0117A000
0117A04C	. 00A01701	DD Example.0117A000
0117A050	. 00A01701	DD Example.0117A000
0117A054	. 00A01701	DD Example.0117A000
0117A058	. 00A01701	DD Example.0117A000

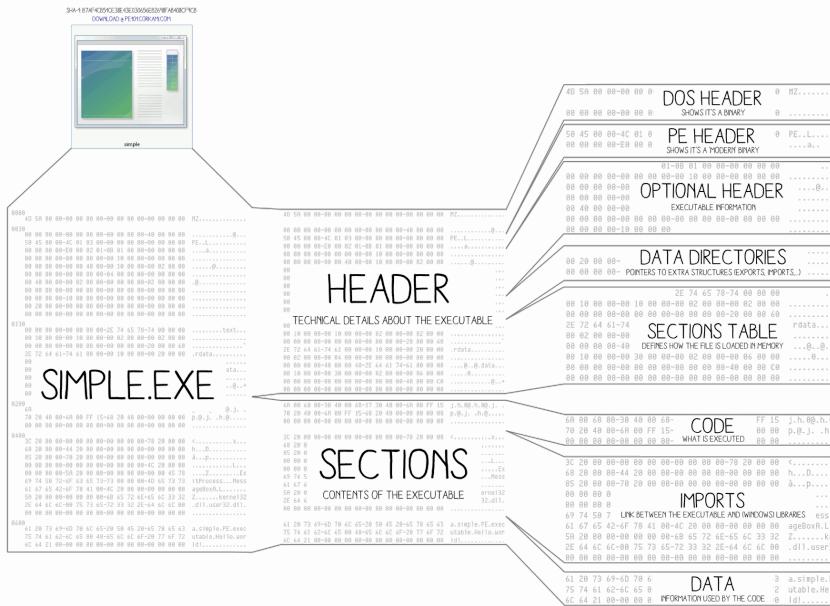
PE Header Sidebar

PE Header Sidebar

PE¹⁰¹ a windows executable walkthrough

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE



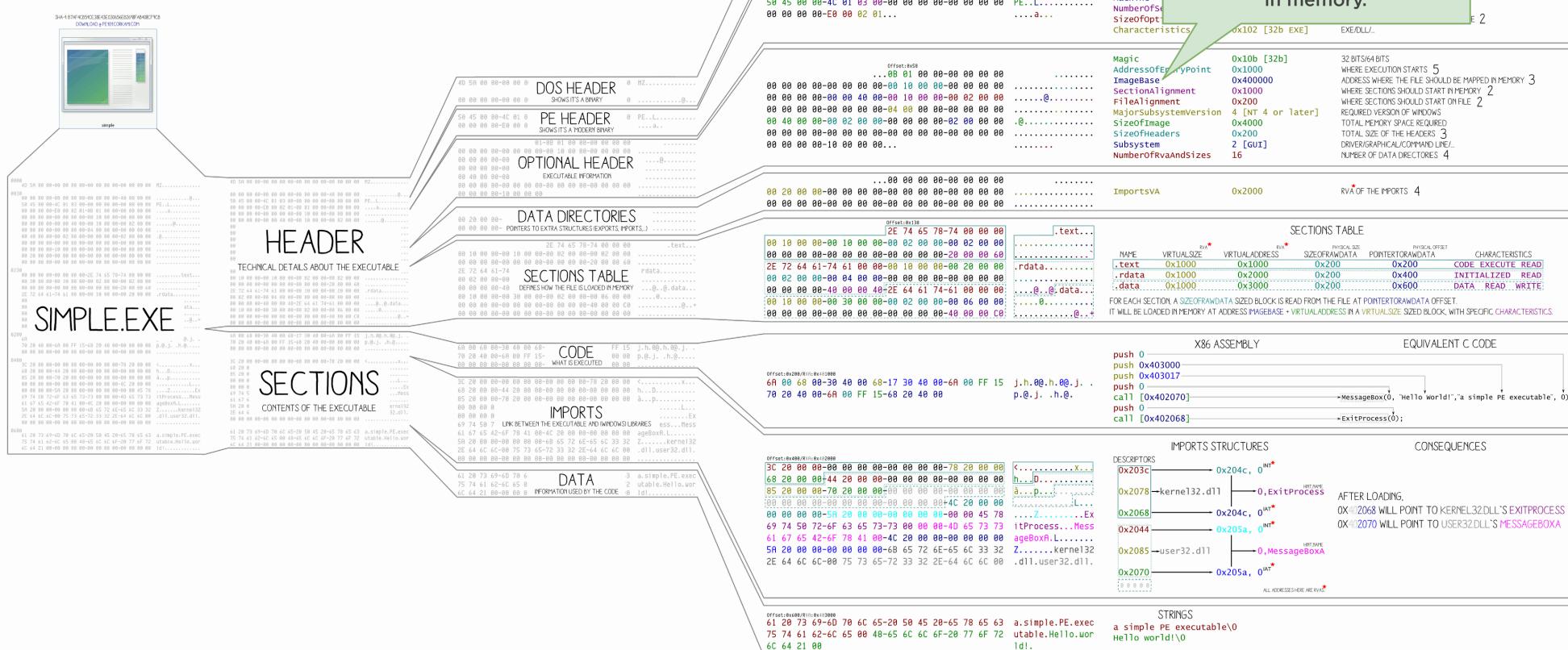
THIS IS THE WHOLE FILE, HOWEVER, MOST PE FILES CONTAIN MORE ELEMENTS.
EXPLANATIONS ARE SIMPLIFIED FOR CONCISENESS.

PE Header Sidebar

PE¹⁰¹
a windows executable walkthrough

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE

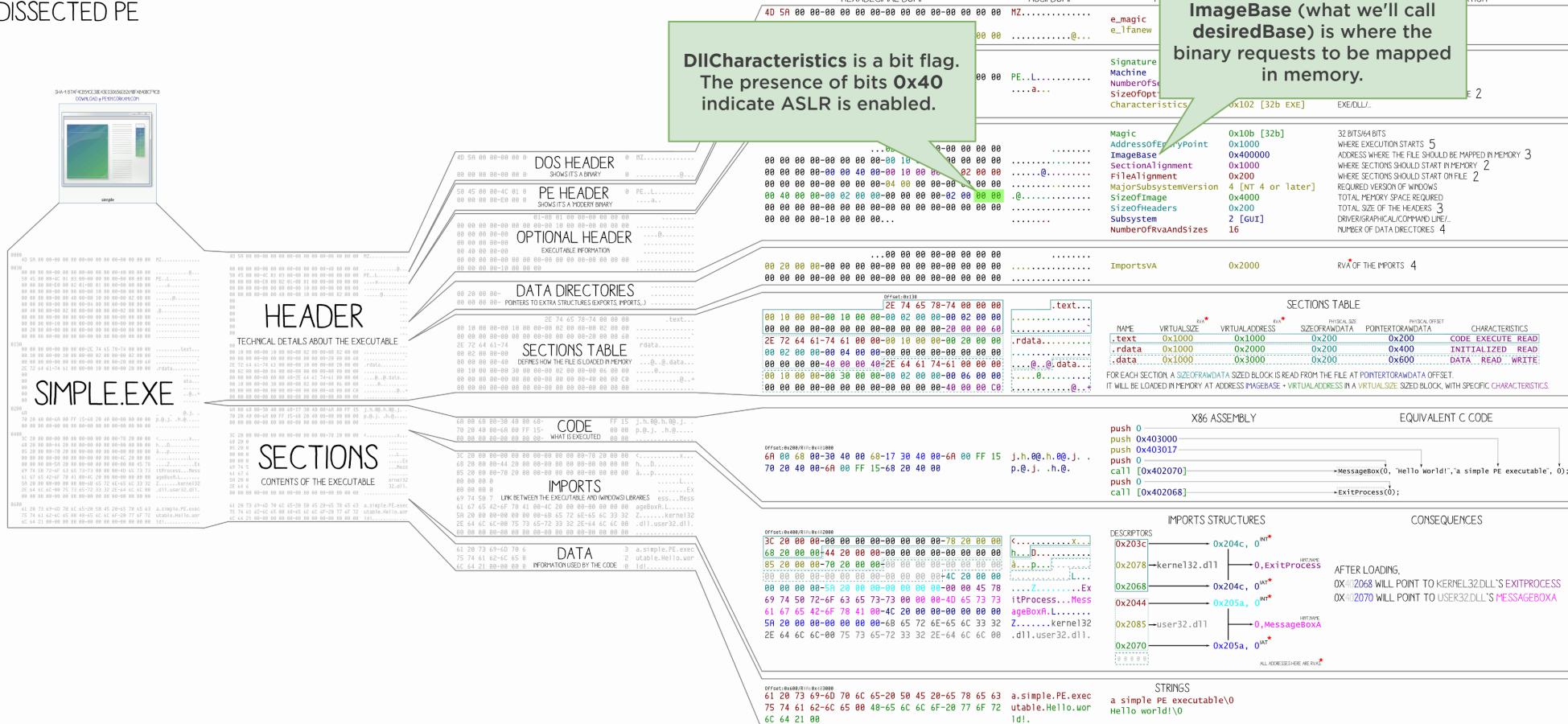


PE Header Sidebar

PE¹⁰¹
a windows executable walkthrough

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE



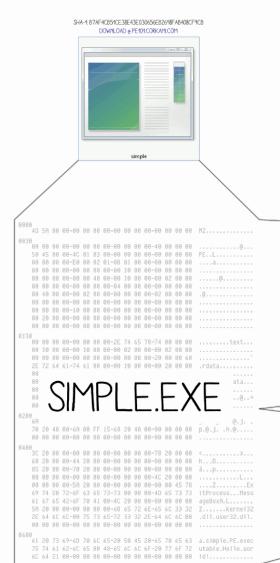
THIS IS THE WHOLE FILE. HOWEVER, MOST PE FILES CONTAIN MORE ELEMENTS.
EXPLANATIONS ARE SKIMMED FOR CONCISENESS.

PE Header Sidebar

PE¹⁰¹
a windows executable walkthrough

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE



HEADER
TECHNICAL DETAILS ABOUT THE EXECUTABLE
SECTION

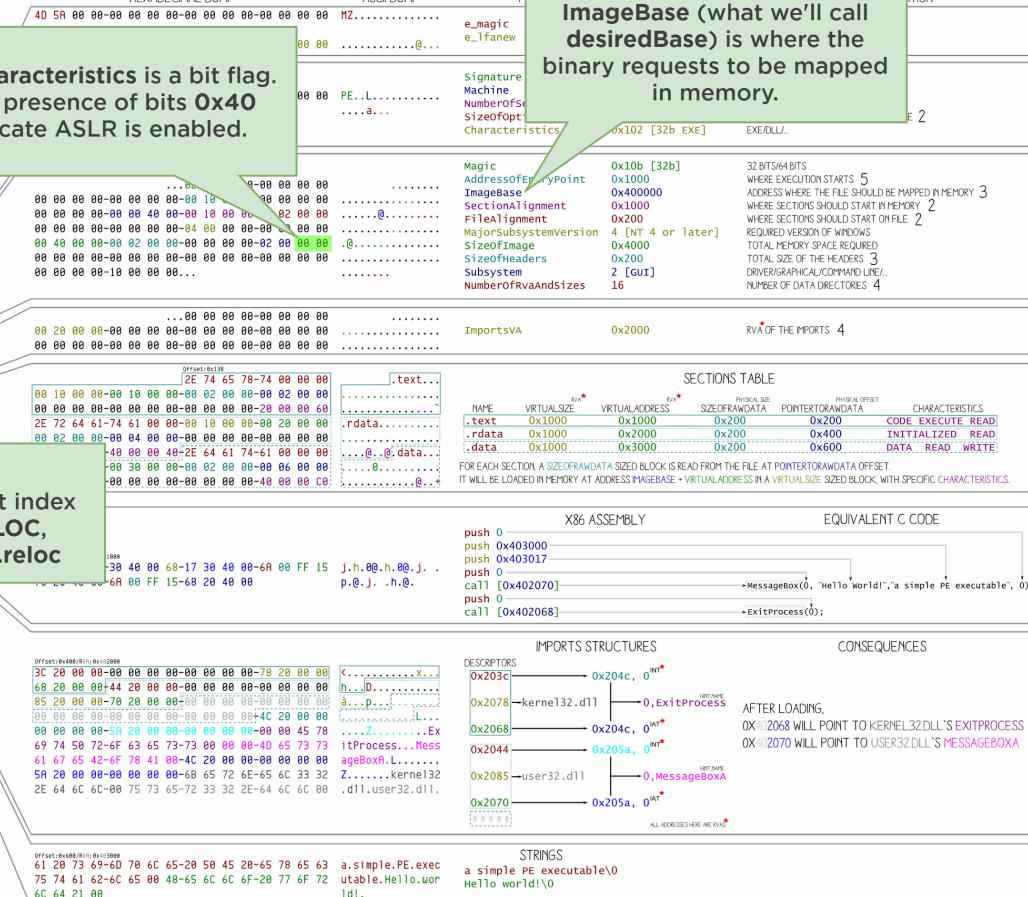
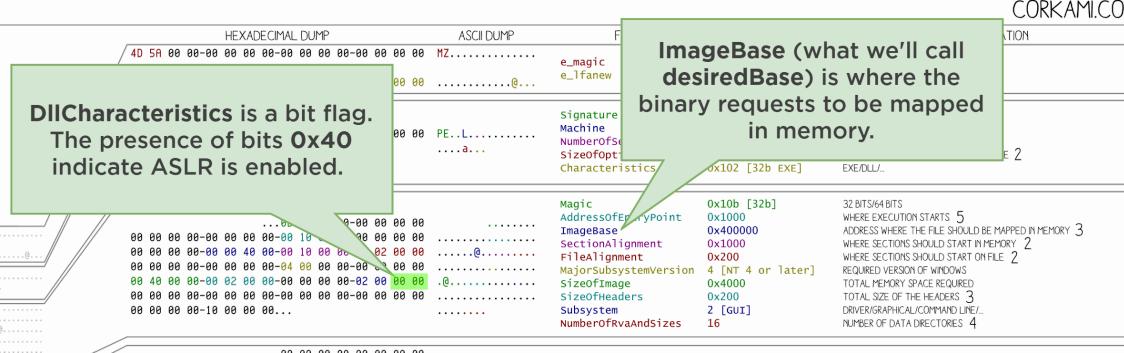
The DataDirectory array stores information about the relocations table at index IMAGE_DIRECTORY_ENTRY_BASERELOC, typically pointing into a section named .reloc

CONTENTS OF THE EXECUTABLE

IMPORTS

DATA

INFORMATION USED BY THE CODE



How Do Relocations Work?

How Do Relocations Work?

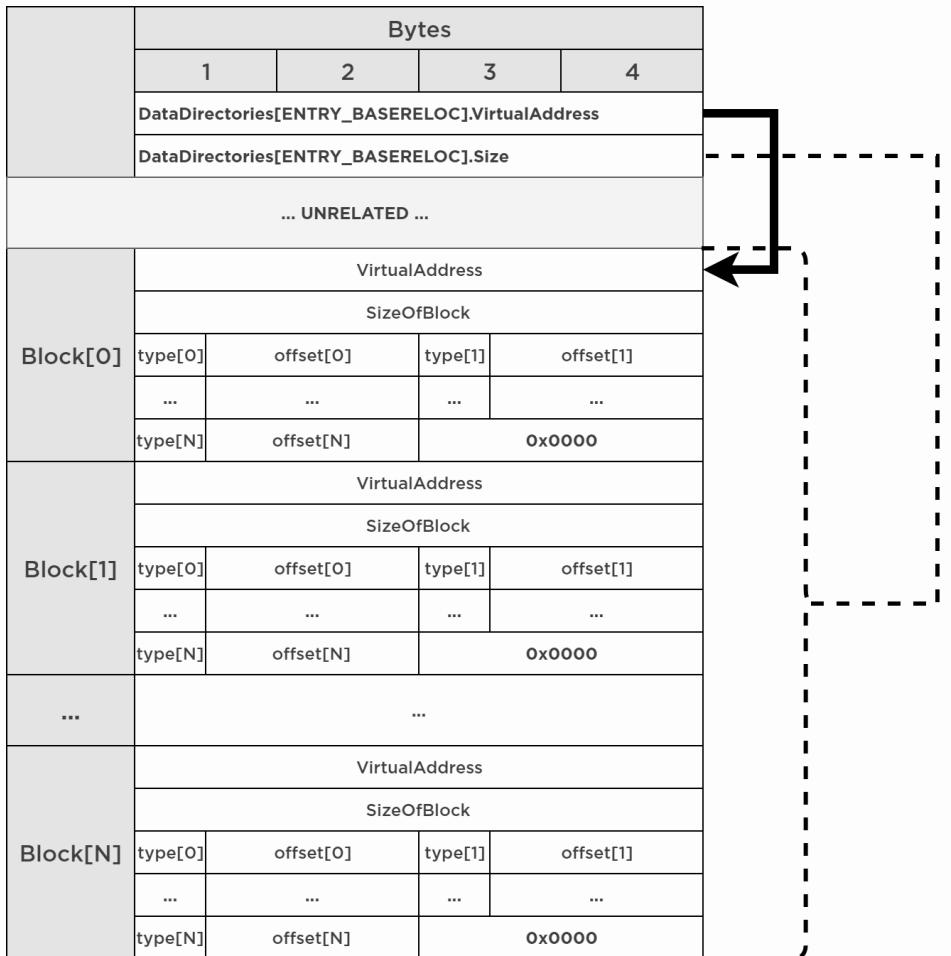
	Bytes					
	1	2	3	4		
DataDirectories[ENTRY_BASERELOC].VirtualAddress						
DataDirectories[ENTRY_BASERELOC].Size						
... UNRELATED ...						
Block[0]	VirtualAddress					
	SizeOfBlock					
	type[0]	offset[0]	type[1]	offset[1]		
		
	type[N]	offset[N]	0x0000			
Block[1]	VirtualAddress					
	SizeOfBlock					
	type[0]	offset[0]	type[1]	offset[1]		
		
	type[N]	offset[N]	0x0000			
...	...					
Block[N]	VirtualAddress					
	SizeOfBlock					
	type[0]	offset[0]	type[1]	offset[1]		
		
	type[N]	offset[N]	0x0000			

How Do Relocations Work?

	Bytes					
	1	2	3	4		
DataDirectories[ENTRY_BASERELOC].VirtualAddress						
DataDirectories[ENTRY_BASERELOC].Size						
... UNRELATED ...						
Block[0]	VirtualAddress					
	SizeOfBlock					
	type[0]	offset[0]	type[1]	offset[1]		
		
	type[N]	offset[N]	0x0000			
Block[1]	VirtualAddress					
	SizeOfBlock					
	type[0]	offset[0]	type[1]	offset[1]		
		
	type[N]	offset[N]	0x0000			
...	...					
Block[N]	VirtualAddress					
	SizeOfBlock					
	type[0]	offset[0]	type[1]	offset[1]		
		
	type[N]	offset[N]	0x0000			

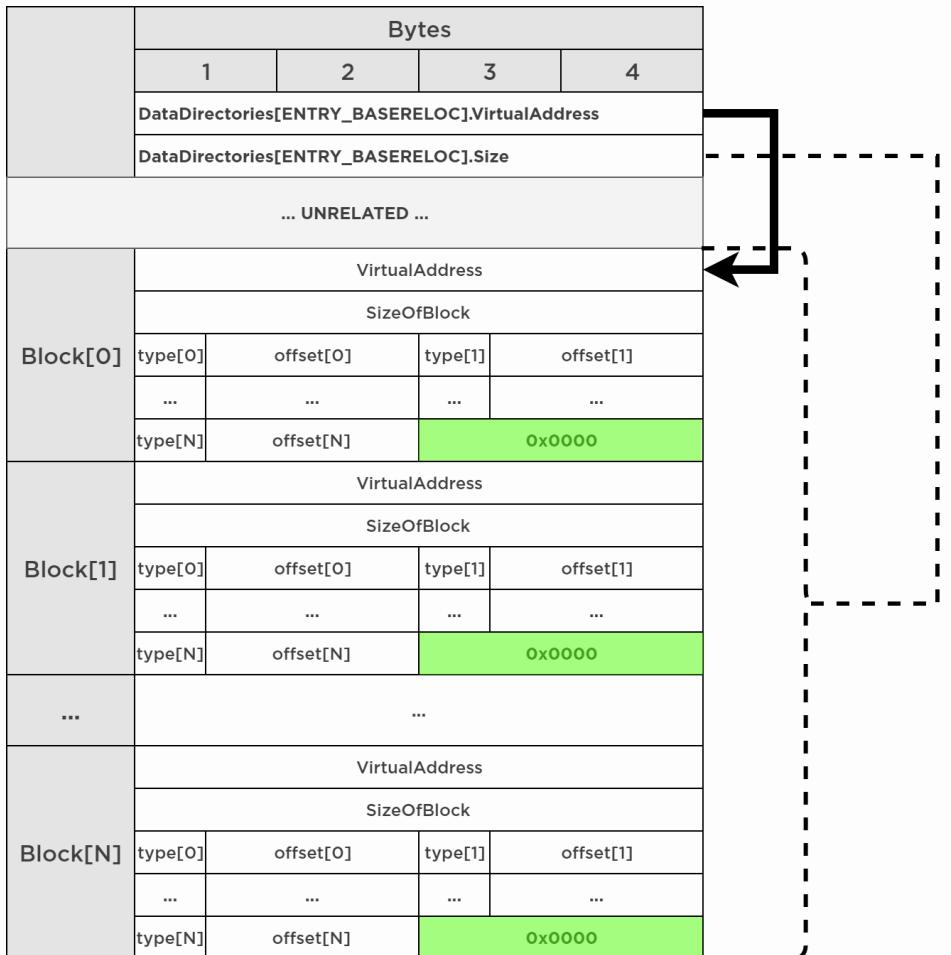
- VirtualAddress points to first reloc block

How Do Relocations Work?



- `VirtualAddress` points to first reloc block
- `Size` is the size, in bytes, of all blocks

How Do Relocations Work?



- `VirtualAddress` points to first reloc block
- `Size` is the size, in bytes, of all blocks
- `(uint16_t)0x0000` marks the end of each block

How Do Relocations Work?

	Bytes			
	1	2	3	4
0x00	VirtualAddress			
0x04	SizeOfBlock			
0x08	type[0]	offset[0]	type[1]	offset[1]
0x0C	type[2]	offset[2]	type[3]	offset[3]

How Do Relocations Work?

	Bytes			
	1	2	3	4
0x00	VirtualAddress			
0x04	SizeOfBlock			
0x08	type[0]	offset[0]	type[1]	offset[1]
0x0C	type[2]	offset[2]	type[3]	offset[3]

Virtual Address for start of the block. Offsets are 12-bit, so a block can cover up to 4096 bytes (or 1024 4-byte addresses)

How Do Relocations Work?

Bytes					
	1	2	3	4	
0x00	VirtualAddress				
0x04	SizeOfBlock				
0x08	type[0]	offset[0]	type[1]	offset[1]	
0x0C	type[2]	offset[2]	type[3]	offset[3]	

The number of bytes which comprise this entire entry, inclusive of address, size, and each entry. In this example, that's $4 + 4 + 2 + 2 + 2 + 2 = 16$.

Virtual Address for start of the block. Offsets are 12-bit, so a block can cover up to 4096 bytes (or 1024 4-byte addresses)

How Do Relocations Work?

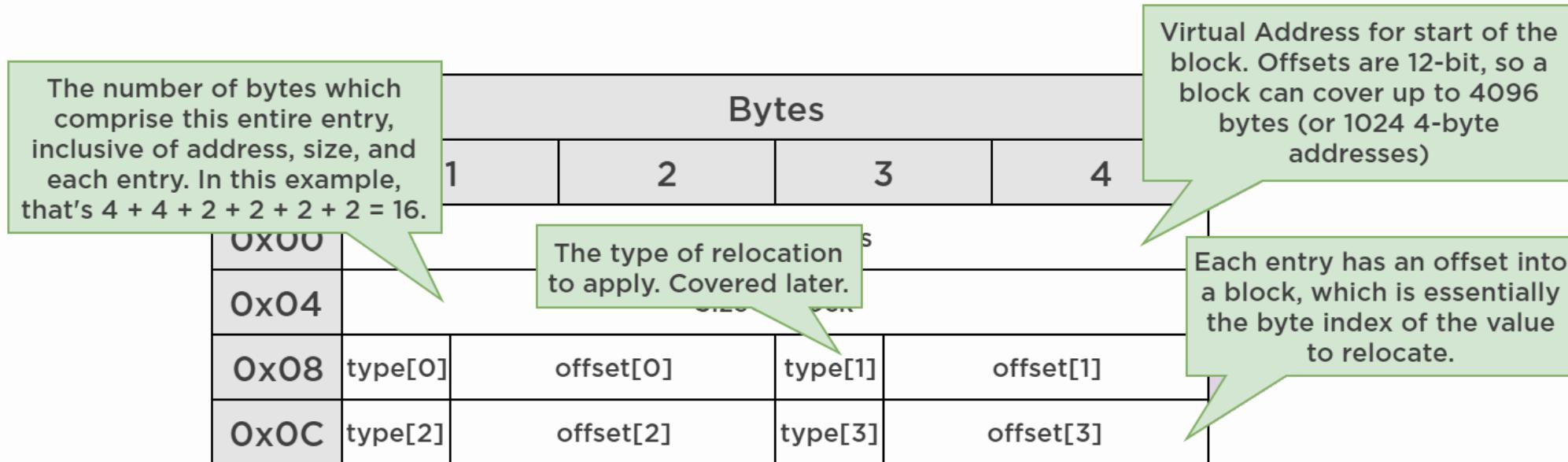
Bytes				
1	2	3	4	
0x00				
0x04				
0x08	type[0]	offset[0]	type[1]	offset[1]
0x0C	type[2]	offset[2]	type[3]	offset[3]

The number of bytes which comprise this entire entry, inclusive of address, size, and each entry. In this example, that's $4 + 4 + 2 + 2 + 2 + 2 = 16$.

Virtual Address for start of the block. Offsets are 12-bit, so a block can cover up to 4096 bytes (or 1024 4-byte addresses)

The type of relocation to apply. Covered later.

How Do Relocations Work?



How Do Relocations Work?

How Do Relocations Work?

Something like this

```
auto delta = base - desiredBase;
for (auto reloc : relocs) {
    auto block = (base + reloc.virtualAddress);
    for (auto entry : reloc.entries) {
        auto adr = block + entry.offset;
        if (entry.type == IMAGE_REL_BASED_HIGHLOW) // <- this
            *((uint32_t*)adr) += delta;
        else if (entry.type == IMAGE_REL_BASED_DIR64)
            *((uint64_t*)adr) += delta;
        else if (entry.type == IMAGE_REL_BASED_HIGH)
            *((uint16_t*)adr) += (uint16_t)((delta >> 16) & 0xFFFF);
        else if (entry.type == IMAGE_REL_BASED_LOW)
            *((uint16_t*)adr) += (uint16_t)delta;
    }
}
```

Controlling Relocations

Controlling Relocations

- Relocations simply use a `+=` operation on data at a specified address

Controlling Relocations

- Relocations simply use a `+=` operation on data at a specified address
- The right-hand side of this operation will be `delta`

Controlling Relocations

- Relocations simply use a `+=` operation on data at a specified address
- The right-hand side of this operation will be `delta`
- `delta` is defined as `base - desiredBase`

Controlling Relocations

- Relocations simply use a `+=` operation on data at a specified address
- The right-hand side of this operation will be `delta`
- `delta` is defined as `base - desiredBase`

Conclusion: to abuse relocations, `base` must be preselected, giving a predictable `delta`. This means ASLR must be tricked.

ASLR Preselection

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

Knowing this, I tried `desiredBase` as:

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

Knowing this, I tried `desiredBase` as:

- `0xFFFFFFFF`

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

Knowing this, I tried `desiredBase` as:

- `0xFFFFFFFF`: PE fails to load; invalid header

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

Knowing this, I tried `desiredBase` as:

- `0xFFFFFFFF`: PE fails to load; invalid header
- `0x00000000`

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

Knowing this, I tried `desiredBase` as:

- `0xFFFFFFFF`: PE fails to load; invalid header
- `0x00000000`: PE fails to load; invalid header

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

Knowing this, I tried `desiredBase` as:

- `0xFFFFFFFF`: PE fails to load; invalid header
- `0x00000000`: PE fails to load; invalid header
- `0xFFFF0000`

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

Knowing this, I tried `desiredBase` as:

- `0xFFFFFFFF`: PE fails to load; invalid header
- `0x00000000`: PE fails to load; invalid header
- `0xFFFF0000`: PE loads at base `0x00010000`

ASLR Preselection

- `desiredBase` is the only means of controlling ASLR
- `delta` is dependant on `desiredBase`

Conclusion: because of how `delta` is derived, `desiredBase` must be made to force ASLR mapping at a predetermined address which isn't `desiredBase` itself.

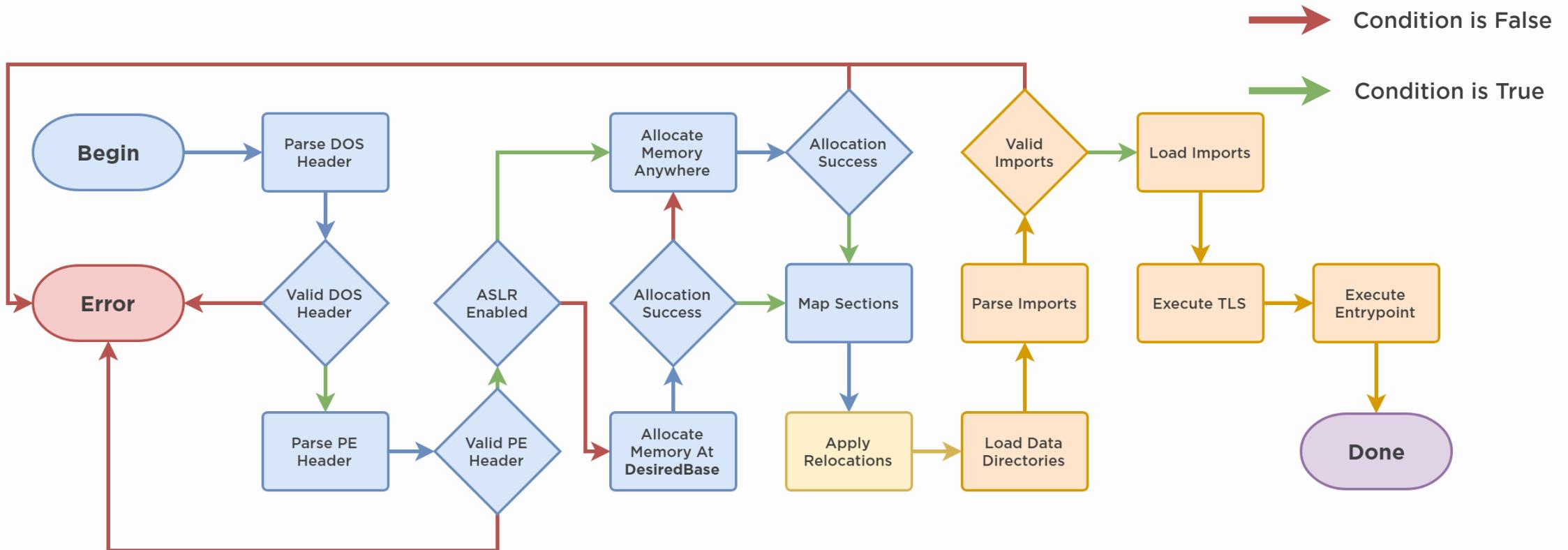
Knowing this, I tried `desiredBase` as:

- `0xFFFFFFFF`: PE fails to load; invalid header
- `0x00000000`: PE fails to load; invalid header
- `0xFFFF0000`: PE loads at base `0x00010000`

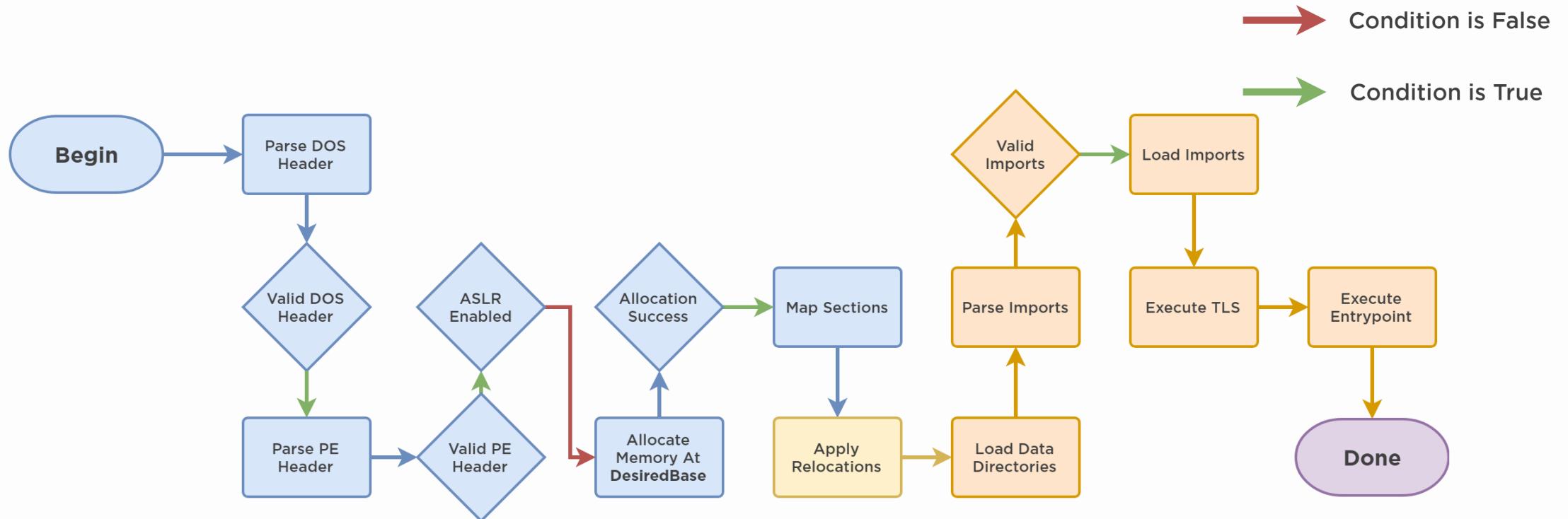
As I later learned, Corkami had already figured all of this out: <https://github.com/corkami/pocs/>

PE Loader Breakdown

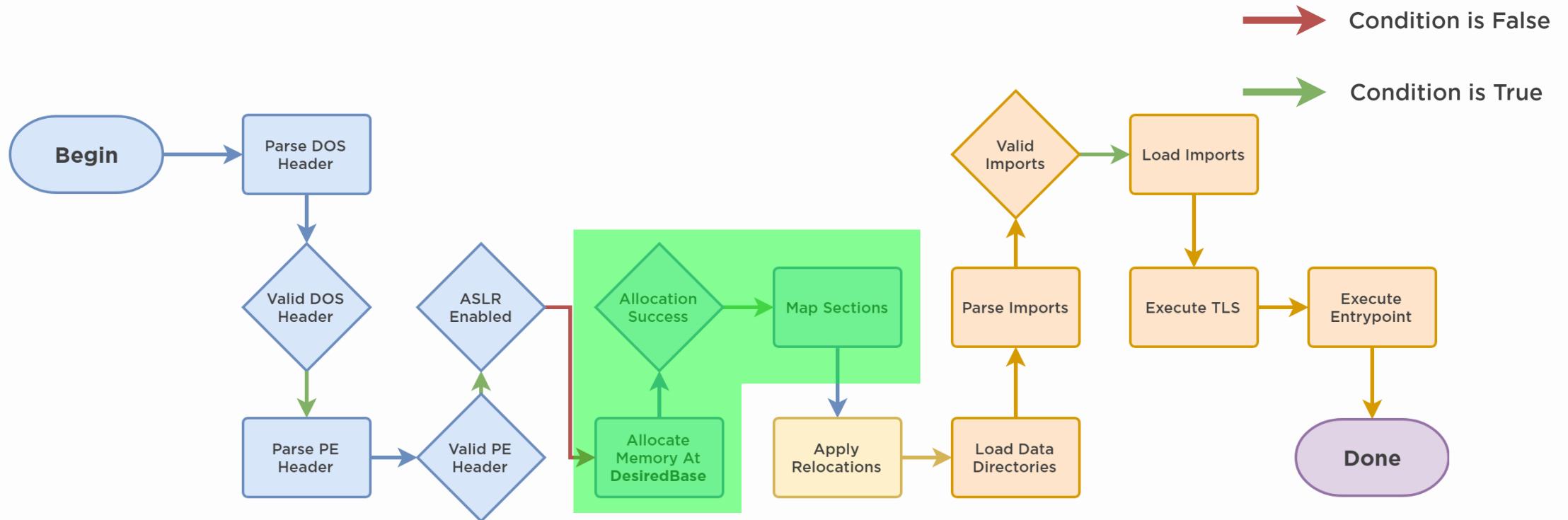
PE Loader Breakdown



PE Loader Breakdown



PE Loader Breakdown



Targets For Relocation Obfuscation

Targets For Relocation Obfuscation

- Import Table is loaded post-reloc

Targets For Relocation Obfuscation

- Import Table is loaded post-reloc
- Many sections are mapped pre-reloc, but not used until execution which is post-reloc

Targets For Relocation Obfuscation

- Import Table is loaded post-reloc
- Many sections are mapped pre-reloc, but not used until execution which is post-reloc
- `entryPoint` isn't used until execution, post-reloc; the memory will be read-only, however, unless DEP is off

Targets For Relocation Obfuscation

- Import Table is loaded post-reloc
- Many sections are mapped pre-reloc, but not used until execution which is post-reloc
- `entryPoint` isn't used until execution, post-reloc; the memory will be read-only, however, unless DEP is off

Conclusion: can mangle imports, code and resource sections, and optionally the `entryPoint` if DEP is off on the target machine.

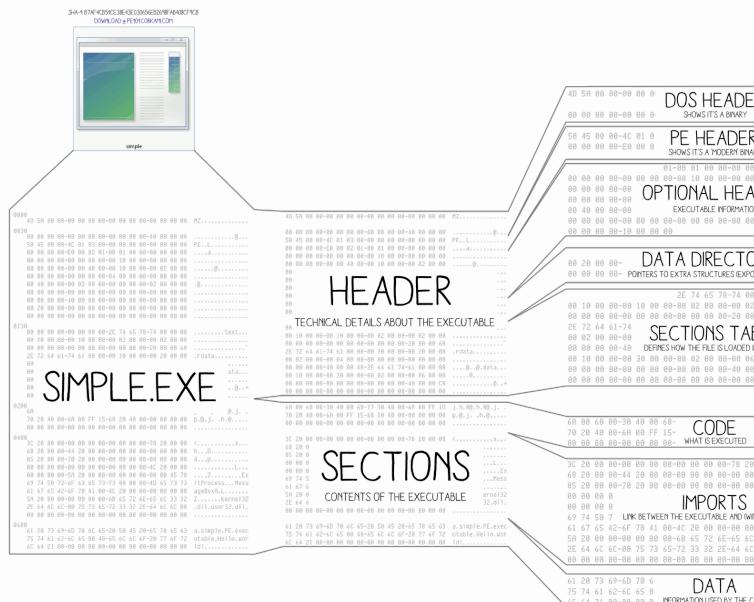
Targets For Relocation Obfuscation

Targets For Relocation Obfuscation

PE¹⁰¹
a windows executable walkthrough

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE



FIELDS	VALUES	EXPLANATION			
e_magic	'MZ'	CONSTANT SIGNATURE			
e_lfanew	0x40	OFFSET OF THE PE HEADER 1			
Signature	'PE', 0, 0	CONSTANT SIGNATURE			
Machine	0x14c [intel 386]	PROCESSOR ARHMIPSINTEL..			
NumberofSections	3	NUMBER OF SECTIONS 2			
SizeofOptionalHeader	0xe0	RELATIVE OFFSET OF THE SECTION TABLE 2			
Characteristics	0x102 [32b EXE]	EXE/DLL..			
Magic	0x10b [32b]	32 BITS/64BITS			
AddressofEntryPoint	0x1000	WHERE EXECUTION STARTS 5			
ImageBase	0x400000	ADDRESS WHERE THE FILE SHOULD BE MAPPED IN MEMORY 3			
SectionAlignment	0x1000	WHERE SECTIONS SHOULD START IN MEMORY 2			
FileAlignment	0x200	WHERE SECTIONS SHOULD START ON FILE 2			
MajorSubsystemVersion	4 [NT 4 or later]	REQUIRED VERSION OF WINDOWS			
SizeofImage	0x4000	TOTAL MEMORY SPACE REQUIRED			
SizeofHeaders	0x200	TOTAL SIZE OF THE HEADERS 3			
Subsystem	2 [GUI]	DRIVER/GRAFIQUE/COMMAND LINE/			
NumberofRvaAndSizes	16	NUMBER OF DATA DIRECTORIES 4			
ImportsVA	0x2000	RVA OF THE IMPORTS 4			
NAME	VIRTUALSIZE	VIRTUALADDRESS	PHYSICALSIZE	PHYSICALADDRESS	CHARACTERISTICS
.text	0x1000	0x1000	0x200	0x200	CODE EXECUTE READ
.rdata	0x1000	0x2000	0x200	0x400	INITIALIZED READ
.data	0x1000	0x600	0x3000	0x200	DATA READ WRITE
FOR EACH SECTION A SIZEOFRWADATA SIZED BLOCK IS READ FROM THE FILE AT POINTERTORAWDATA OFFSET. IT WILL BE LOADED IN MEMORY AT ADDRESS IMAGEBASE + VIRTUALADDRESS IN A VIRTUALSIZE SIZED BLOCK, WITH SPECIFIC CHARACTERISTICS.					
X86 ASSEMBLY	EQUIVALENT C CODE				
push 0	push 0x403000				
push 0x403017					
push 0	push 0	MessageBox(0, "Hello world!", "a simple PE executable", 0);			
call [0x402070]					
push 0		ExitProcess();			
call [0x402068]					
IMPORTS STRUCTURES					
0x203c	0x204c, 0 ^{INT*}				
0x2078	kernel32.dll, 0, ExitProcess				
0x2068	0x204c, 0 ^{INT*}				
0x2044	0x205a, 0 ^{INT*}				
0x2085	user32.dll, 0, MessageBoxA				
0x2050	0x205a, 0 ^{INT*}				
CONSEQUENCES					
AFTER LOADING, 0x2068 WILL POINT TO KERNEL32.DLL'S EXITPROCESS 0x2070 WILL POINT TO USER32.DLL'S MESSAGEBOXA					
STRINGS					
a simple PE executable\0					
Hello world!\0					

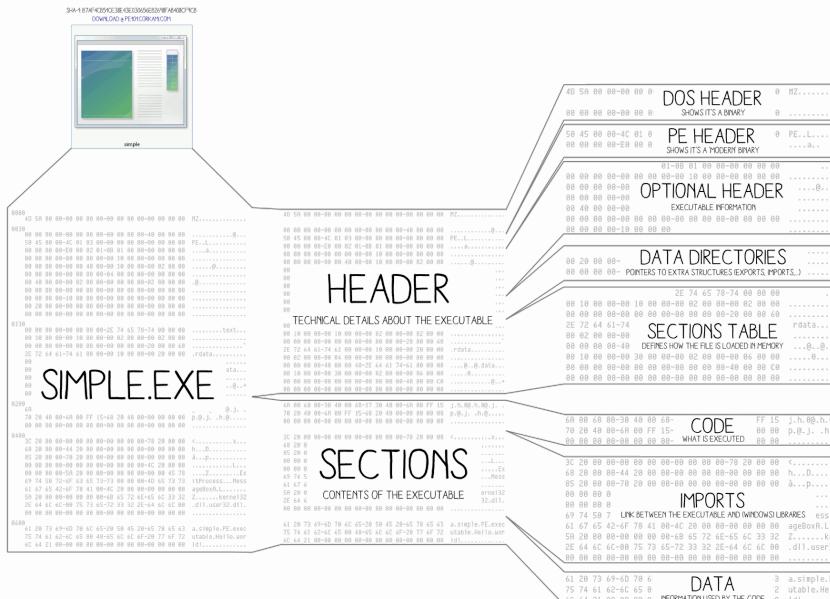
THIS IS THE WHOLE FILE, HOWEVER MOST FILES CONTAIN MORE ELEMENTS.
EXPLANATIONS ARE SIMPLIFIED FOR CONCISENESS.

Targets For Relocation Obfuscation

PE¹⁰¹ a windows executable walkthrough

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE



HEXADECIMAL DUMP

40 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00	MZ.....
00 00 00 00-00 00 00-00 00 00-00 40 00 00 00@...

ASCII DUMP

50 45 00 00-4C 01 03 02 00-00 00 00 00-00 00 00 00	PE.L.....
00 00 00 00-E0 00 02 01...@...

FIELDS

e_magic	'MZ'	CONSTANT SIGNATURE
e_lfanew	0x40	OFFSET OF THE PE HEADER 1

VALUES

Signature	'PE', 0, 0	CONSTANT SIGNATURE
Machine	0x14c [intel 386]	PROCESSOR ARM/MMPS(INTEL)..
NumberofSections	3	NUMBER OF SECTIONS 2
SizeofOptionalHeader	0x0e	RELATIVE OFFSET OF THE SECTION TABLE 2
Characteristics	0x102 [32b EXE]	EXE/DLL..

EXPLANATION

32 BITS/64 BITS
WHERE EXECUTION STARTS 5
ADDRESS WHERE THE FILE SHOULD BE MAPPED IN MEMORY 3
WHERE SECTIONS SHOULD START IN MEMORY 2
WHERE SECTIONS SHOULD START ON FILE 2
REQUIRED VERSION OF WINDOWS
TOTAL MEMORY SPACE REQUIRED
TOTAL SIZE OF THE HEADERS 3
DRIVER/GRAFICAL/COMMAND LINE/
NUMBER OF DATA DIRECTORIES 4

ImportsSVA 0x2000 RVA OF THE IMPORTS 4

SECTIONS TABLE

NAME	VIRTUAL SIZE	VIRTUAL ADDRESS	PHYSICAL SIZE	SIZEOFRAWDATA	POINTERTORAWDATA	CHARACTERISTICS
.text	0x1000	0x1000	0x200	0x200	0x200	CODE EXECUTE READ
.rdata	0x1000	0x2000	0x200	0x400	0x400	INITIALIZED READ
.data	0x1000	0x3000	0x200	0x600	0x600	DATA READ WRITE

FOR EACH SECTION A SIZEOFRAWDATA SIZED BLOCK IS READ FROM THE FILE AT POINTERTORAWDATA OFFSET.
IT WILL BE LOADED IN MEMORY AT ADDRESS IMAGEBASE + VIRTUALADDRESS IN A VIRTUALSIZE SIZED BLOCK, WITH SPECIFIC CHARACTERISTICS.

X86 ASSEMBLY

```
push 0
push 0x403000
push 0x403017
push 0
call [0x402070]
push 0
call [0x402068]
```

EQUIVALENT C CODE

```
MessageBox(0, "Hello World!", a simple PE executable, 0);
ExitProcess();
```

IMPORTS STRUCTURES

DESCRIPTORS	ADDRESS	NAME
0x203C	0x204c, 0 ^{NH*}	kernel32.dll
0x2078	0, ExitProcess	
0x2068	0x204c, 0 ^{NAT*}	
0x2044	0x205a, 0 ^{NH*}	
0x2085	0, MessageBoxA	
0x2070	0x205a, 0 ^{NAT*}	

CONSEQUENCES

AFTER LOADING,
0X 02068 WILL POINT TO KERNEL32.DLL'S EXITPROCESS
0X 02070 WILL POINT TO USER32.DLL'S MESSAGEBOXA

STRINGS

61 20 73 65-60 70 6C 65-20 50 45 20-65 78 65 63	a.simple.PE.executable.Hello.world!()
75 74 61 62-6C 65 00 48-65 6C 6C 6F-2B 77 6F 72	
6C 64 21 00	

THIS IS THE WHOLE FILE, HOWEVER, MOST PE FILES CONTAIN MORE ELEMENTS.
EXPLANATIONS ARE SIMPLIFIED, FOR CONCISENESS.

The Final Attack

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file
- Apply original relocations for base of 0x00010000

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file
- Apply original relocations for base of 0x00010000
- Turn ASLR off by flipping a bit in the PE Header

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file
- Apply original relocations for `base` of `0x00010000`
- Turn ASLR off by flipping a bit in the PE Header
- Set `desiredBase` to `0xFFFF0000`

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file
- Apply original relocations for `base` of `0x00010000`
- Turn ASLR off by flipping a bit in the PE Header
- Set `desiredBase` to `0xFFFF0000`
- Loop over data to obfuscate in `uint32_t`-sized chunks, decrementing each by `0x00010000 - 0xFFFF0000` (expected value of `delta`)

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file
- Apply original relocations for `base` of `0x00010000`
- Turn ASLR off by flipping a bit in the PE Header
- Set `desiredBase` to `0xFFFF0000`
- Loop over data to obfuscate in `uint32_t`-sized chunks, decrementing each by `0x00010000 - 0xFFFF0000` (expected value of `delta`)
- Discard original relocations table

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file
- Apply original relocations for `base` of `0x00010000`
- Turn ASLR off by flipping a bit in the PE Header
- Set `desiredBase` to `0xFFFF0000`
- Loop over data to obfuscate in `uint32_t`-sized chunks, decrementing each by `0x00010000 - 0xFFFF0000` (expected value of `delta`)
- Discard original relocations table
- Generate new relocations table containing the location of each decrement done inside of the loop (using `IMAGE_REL_BASED_HIGHLOW`)

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file
- Apply original relocations for `base` of `0x00010000`
- Turn ASLR off by flipping a bit in the PE Header
- Set `desiredBase` to `0xFFFF0000`
- Loop over data to obfuscate in `uint32_t`-sized chunks, decrementing each by `0x00010000 - 0xFFFF0000` (expected value of `delta`)
- Discard original relocations table
- Generate new relocations table containing the location of each decrement done inside of the loop (using `IMAGE_REL_BASED_HIGHLOW`)
- Save new PE file to disk

The Final Attack

Due to the nature of the attack, it works best as a tool which rebuilds regular PE files.

- Load target PE file
- Apply original relocations for `base` of `0x00010000`
- Turn ASLR off by flipping a bit in the PE Header
- Set `desiredBase` to `0xFFFF0000`
- Loop over data to obfuscate in `uint32_t`-sized chunks, decrementing each by `0x00010000 - 0xFFFF0000` (expected value of `delta`)
- Discard original relocations table
- Generate new relocations table containing the location of each decrement done inside of the loop (using `IMAGE_REL_BASED_HIGHLOW`)
- Save new PE file to disk
- ??? profit

ROUND ONE

FIGHT

Testing The Attack

Testing The Attack

- Windows 7

Testing The Attack

- Windows 7 ... works !

Testing The Attack

- **Windows 7** ... works !
- **Windows 8**

Testing The Attack

- **Windows 7** ... works !
- **Windows 8** ... nobody uses this shit

Testing The Attack

- **Windows 7** ... works !
- **Windows 8** ... nobody uses this shit
- **Windows 10**

Testing The Attack

- Windows 7 ... works !
- Windows 8 ... nobody uses this shit
- Windows 10



Testing The Attack

- Windows 7 . works
- Windows 8 . nobody uses this shit
- Windows 10



THE END

Exploring New Terrain

Exploring New Terrain

- Embed PE copies for all possible base addresses

Exploring New Terrain

- Embed PE copies for all possible base addresses ... way too big

Exploring New Terrain

- Embed PE copies for all possible base addresses ... way too big 
- Tweaking ASLR Configuration

Exploring New Terrain

- Embed PE copies for all possible base addresses ... way too big 
- Tweaking ASLR Configuration ... works!

Exploring New Terrain

- Embed PE copies for all possible base addresses ... way too big 
- Tweaking ASLR Configuration ... works!
 - Set Mandatory ASLR to On

Exploring New Terrain

- Embed PE copies for all possible base addresses ... way too big 🍆
- Tweaking ASLR Configuration ... works!
 - Set Mandatory ASLR to On
 - Set Bottom-Up ASLR to Off

Exploring New Terrain

- Embed PE copies for all possible base addresses ... way too big 🍆
- Tweaking ASLR Configuration ... works!
 - Set Mandatory ASLR to On
 - Set Bottom-Up ASLR to Off

```
[HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\  
Image File Execution Options\NAME_OF_EXE]  
"MitigationAuditOptions"=hex:00,00,00,00,00,00,00,00,\  
          00,00,00,00,00,00,00,00  
"MitigationOptions"=hex:00,01,22,00,00,00,00,00,\  
          00,00,00,00,00,00,00,00
```

Exploring New Terrain

- Embed PE copies for all possible base addresses ... way too big 🍆
- Tweaking ASLR Configuration ... works!
 - Set Mandatory ASLR to On
 - Set Bottom-Up ASLR to Off

```
[HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\  
Image File Execution Options\NAME_OF_EXE]  
"MitigationAuditOptions"=hex:00,00,00,00,00,00,00,00,\  
          00,00,00,00,00,00,00,00  
"MitigationOptions"=hex:00,01,22,00,00,00,00,00,\  
          00,00,00,00,00,00,00,00
```

Conclusion: it can work on Windows 10

Exploring New Terrain

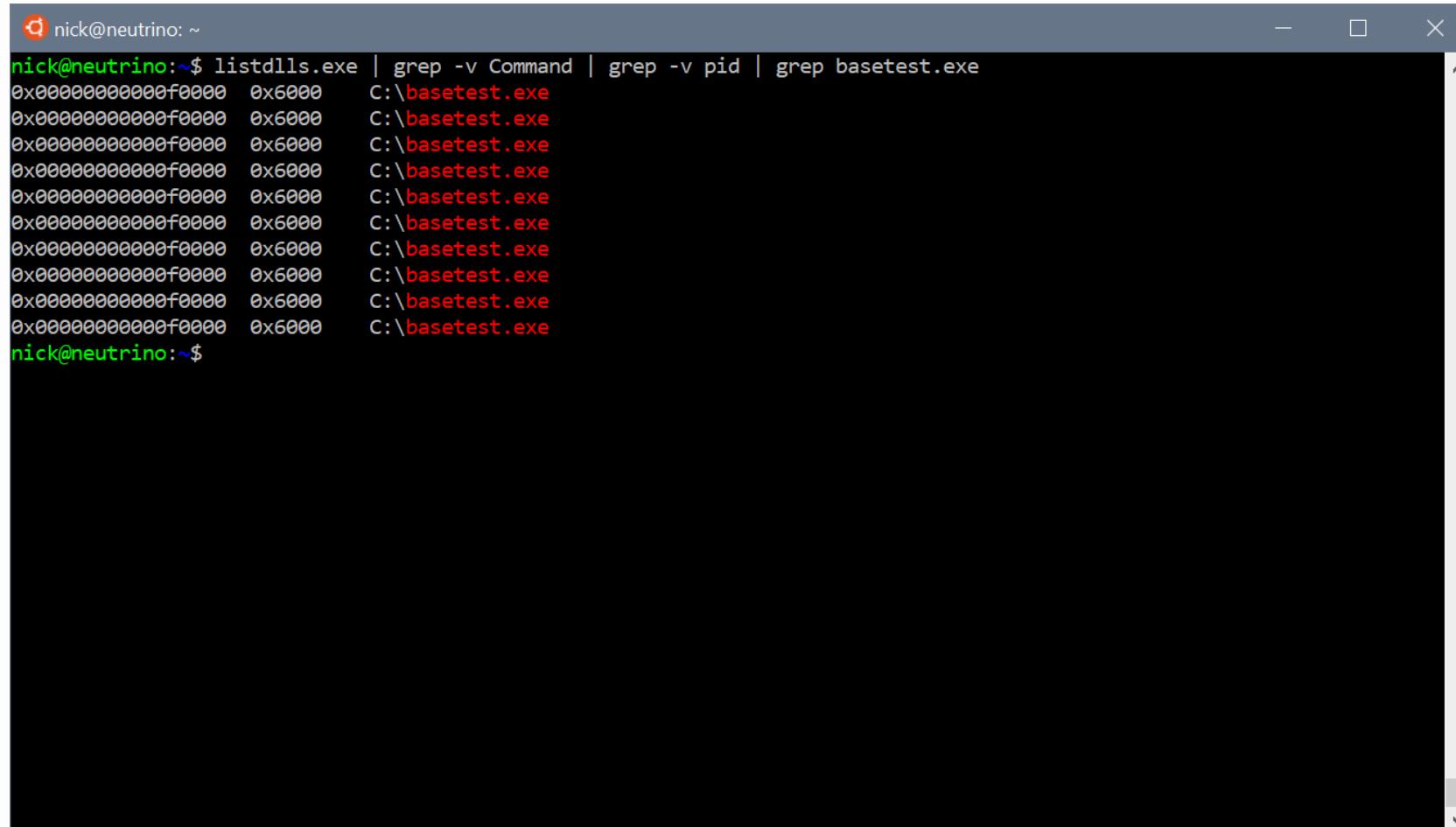
- Embed PE copies for all possible base addresses ... way too big 🍆
- Tweaking ASLR Configuration ... works!
 - Set Mandatory ASLR to On
 - Set Bottom-Up ASLR to Off

```
[HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\  
Image File Execution Options\NAME_OF_EXE]  
"MitigationAuditOptions"=hex:00,00,00,00,00,00,00,00,\  
          00,00,00,00,00,00,00,00  
"MitigationOptions"=hex:00,01,22,00,00,00,00,00,\  
          00,00,00,00,00,00,00,00
```

Conclusion: it can work on Windows 10, but I don't like it

A New Hope

A New Hope



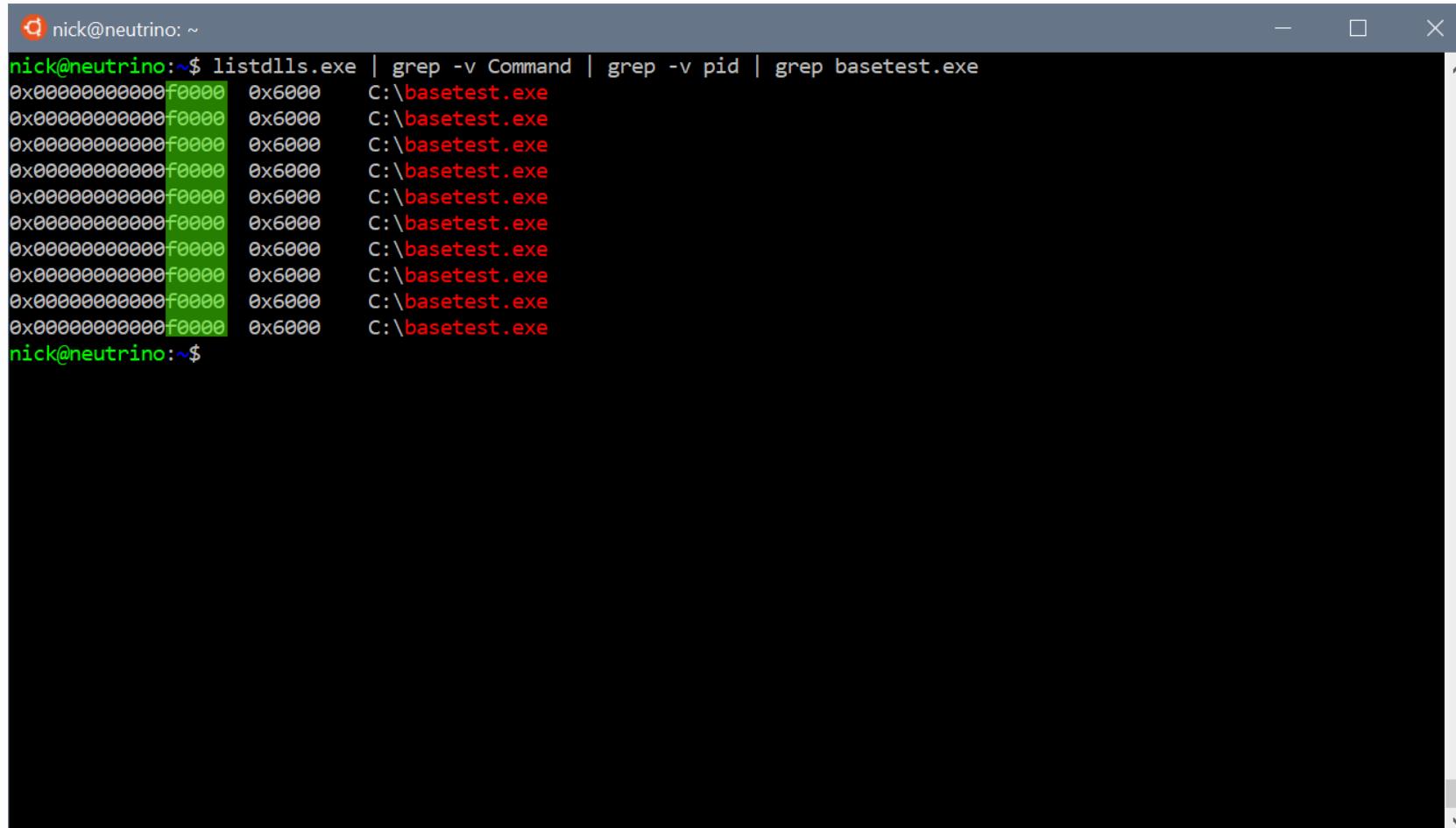
The screenshot shows a terminal window with a dark background and light-colored text. The window title bar indicates the session is running on a machine named 'neutrino' with a user named 'nick'. The command entered by the user is:

```
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
```

The output of the command shows multiple entries for the file 'basetest.exe' located at 'C:\basetest.exe', with a memory address of '0x6000' and a base offset of '0x00000000f0000'. There are ten such entries displayed.

```
0x00000000f0000 0x6000 C:\basetest.exe
```

A New Hope



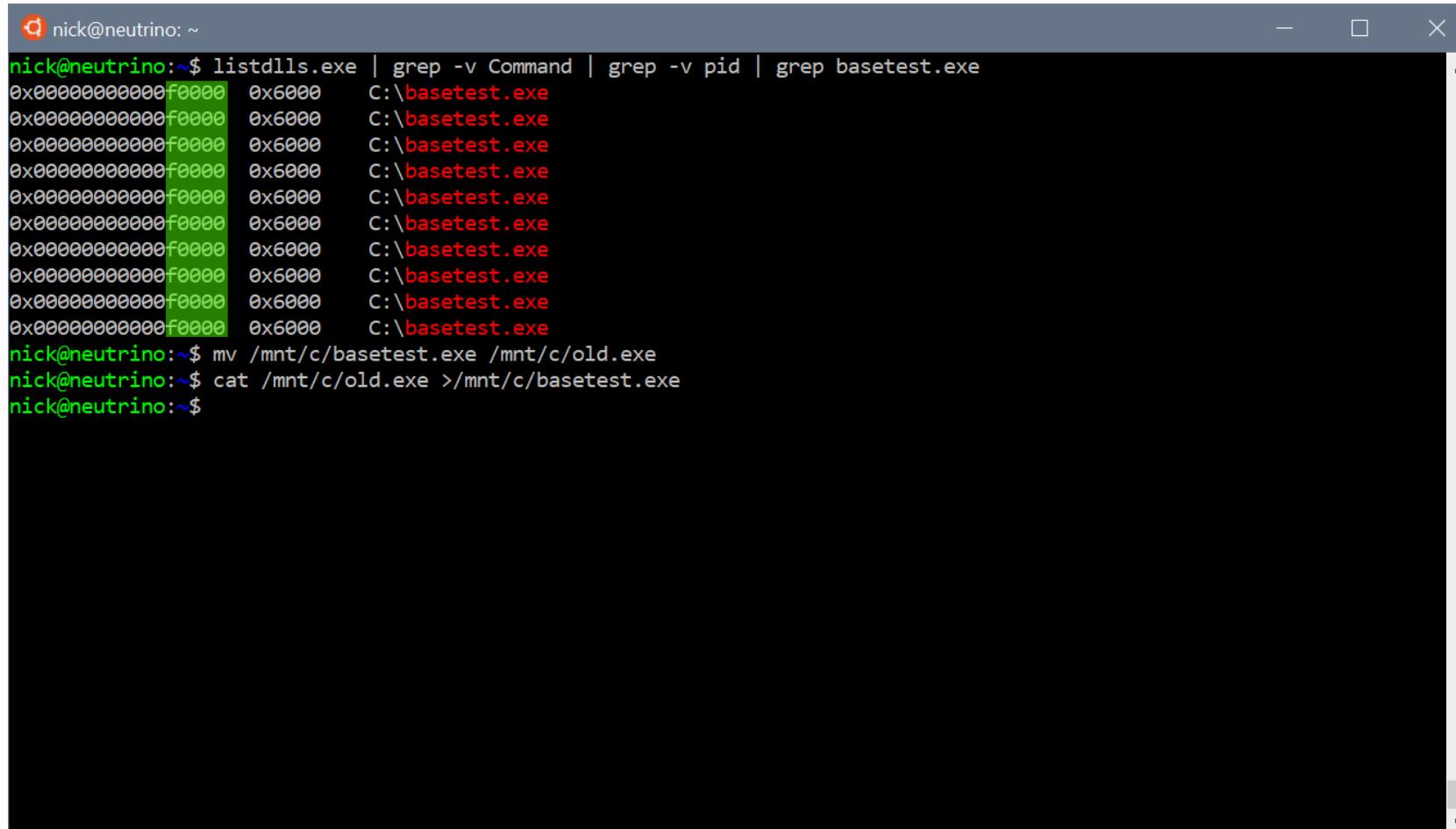
The screenshot shows a terminal window with a dark background and light-colored text. The window title bar indicates the session is running on a machine named 'neutrino'. The command entered by the user is:

```
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
```

The output of the command shows multiple entries for the file 'basetest.exe' located at 'C:\basetest.exe', each with a memory address starting with '0x0000000000F0000' and a process ID of '0x6000'. The text is color-coded, with the file name 'basetest.exe' appearing in red.

Memory Address	Process ID	File Path
0x0000000000F0000	0x6000	C:\basetest.exe

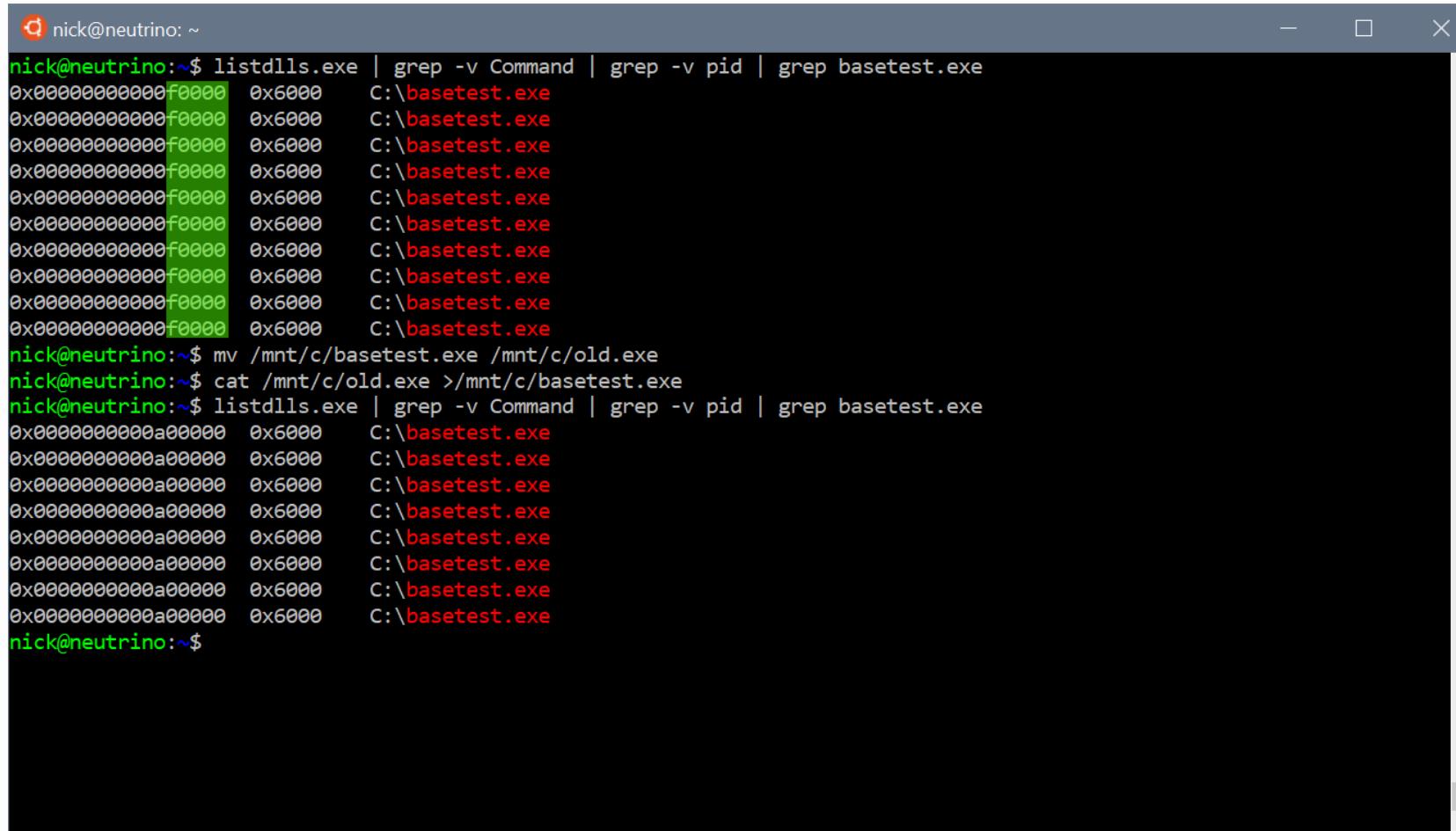
A New Hope



The screenshot shows a terminal window with a dark background and light-colored text. The window title is "nick@neutrino: ~". The terminal output is as follows:

```
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x000000000000F0000 0x6000 C:\basetest.exe
nick@neutrino:~$ mv /mnt/c/basetest.exe /mnt/c/old.exe
nick@neutrino:~$ cat /mnt/c/old.exe >/mnt/c/basetest.exe
nick@neutrino:~$
```

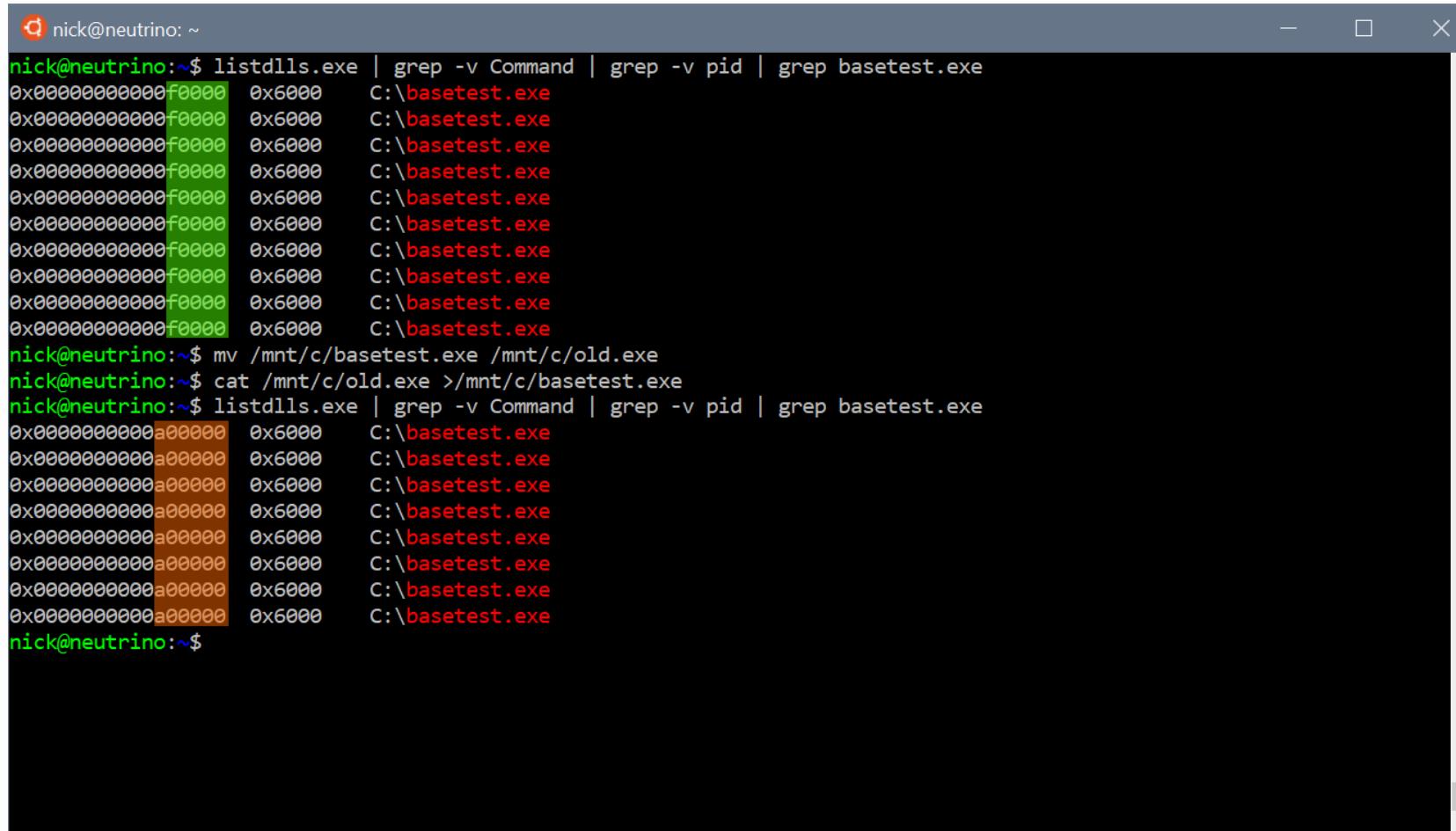
A New Hope



The screenshot shows a terminal window with a dark background and light-colored text. It displays a sequence of commands and their outputs related to file manipulation and memory dump analysis.

```
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x0000000000F0000 0x6000 C:\basetest.exe
nick@neutrino:~$ mv /mnt/c/basetest.exe /mnt/c/old.exe
nick@neutrino:~$ cat /mnt/c/old.exe >/mnt/c/basetest.exe
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x0000000000a0000 0x6000 C:\basetest.exe
nick@neutrino:~$
```

A New Hope

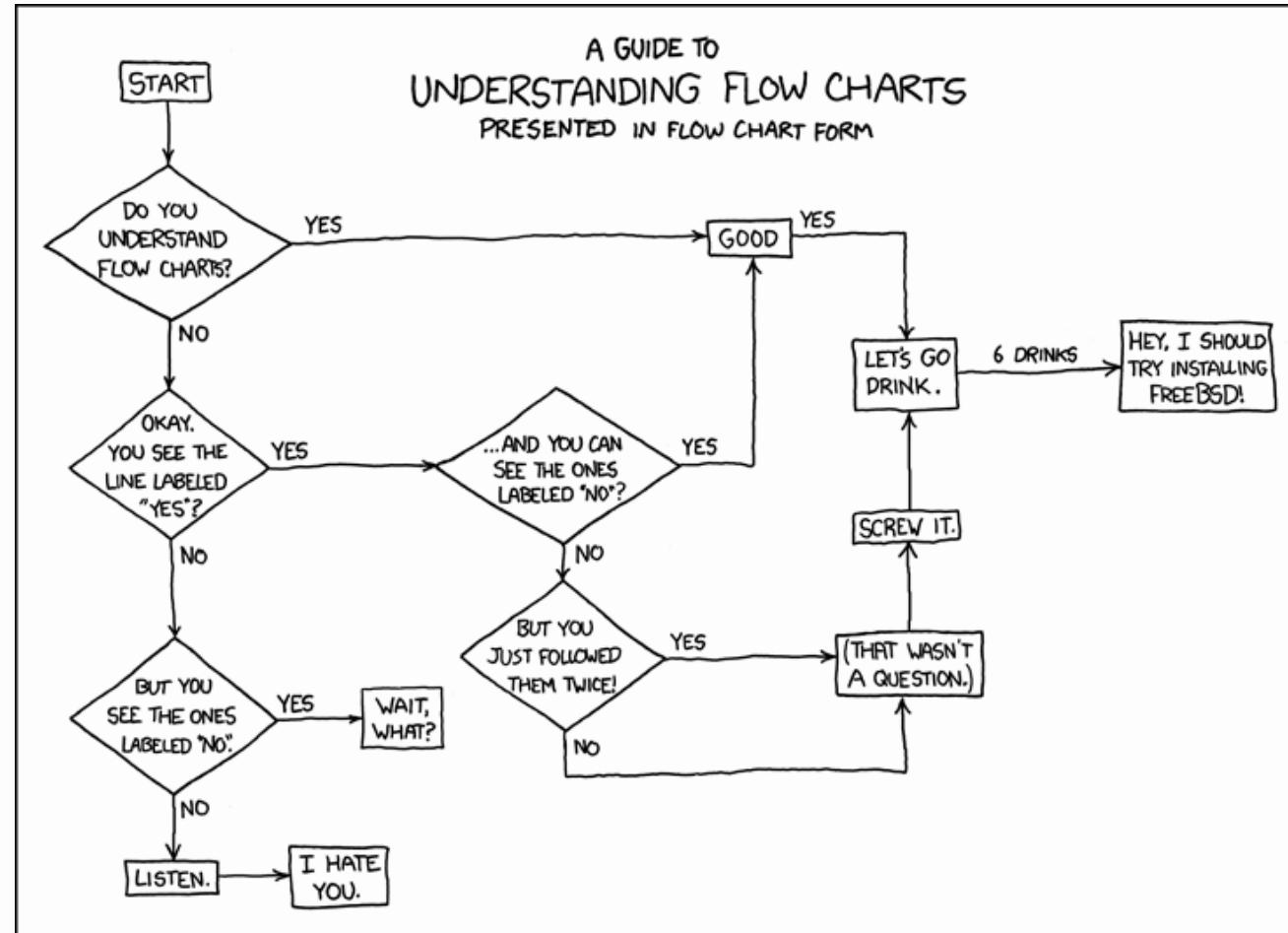


The screenshot shows a terminal window titled "nick@neutrino: ~". The terminal displays the following command-line session:

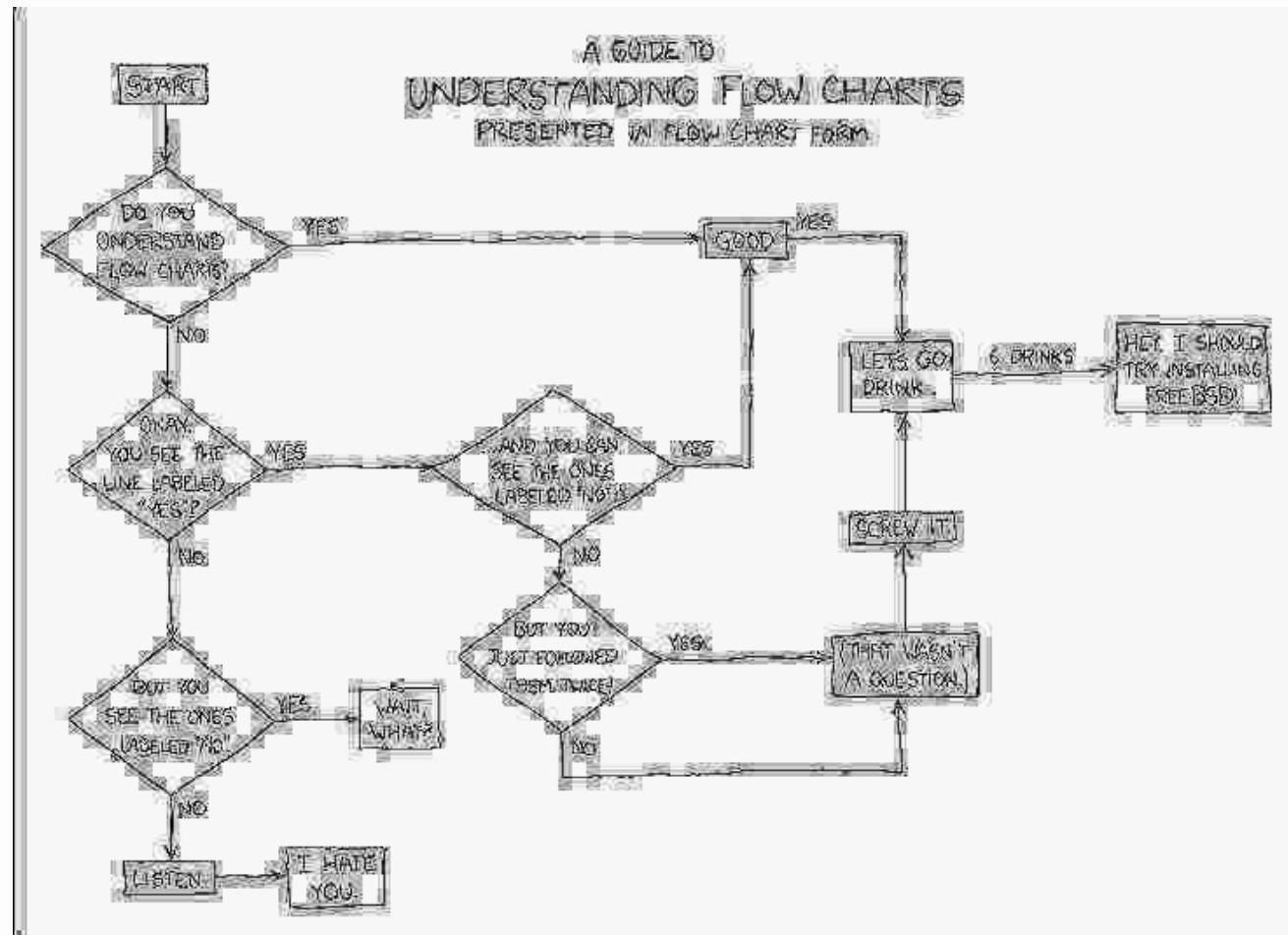
```
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x0000000000F0000 0x6000 C:\basetest.exe
nick@neutrino:~$ mv /mnt/c/basetest.exe /mnt/c/old.exe
nick@neutrino:~$ cat /mnt/c/old.exe >/mnt/c/basetest.exe
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x0000000000a0000 0x6000 C:\basetest.exe
nick@neutrino:~$
```

Preselection via File Mapping Invalidation

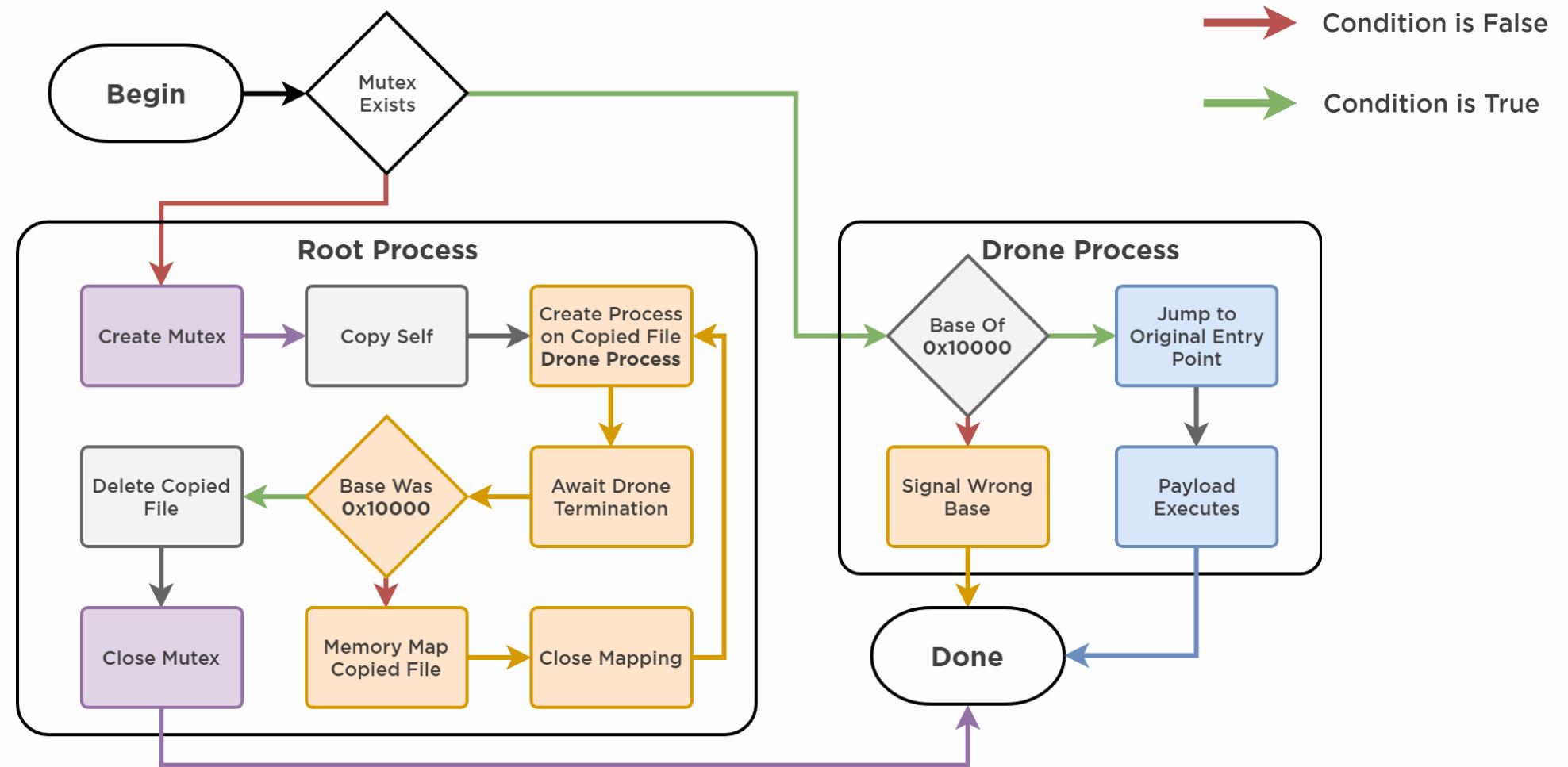
Preselection via File Mapping Invalidation



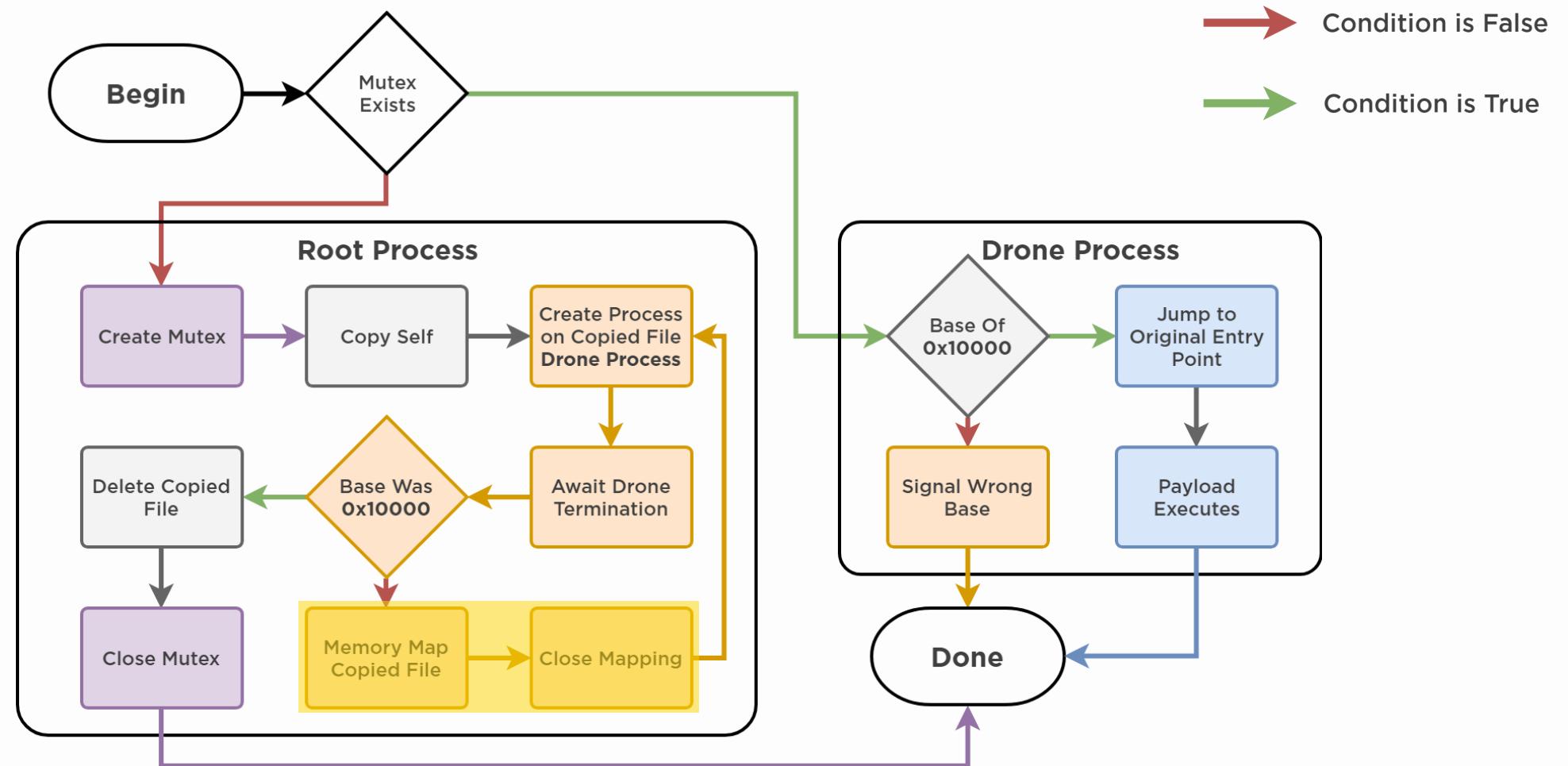
Preselection via File Mapping Invalidation



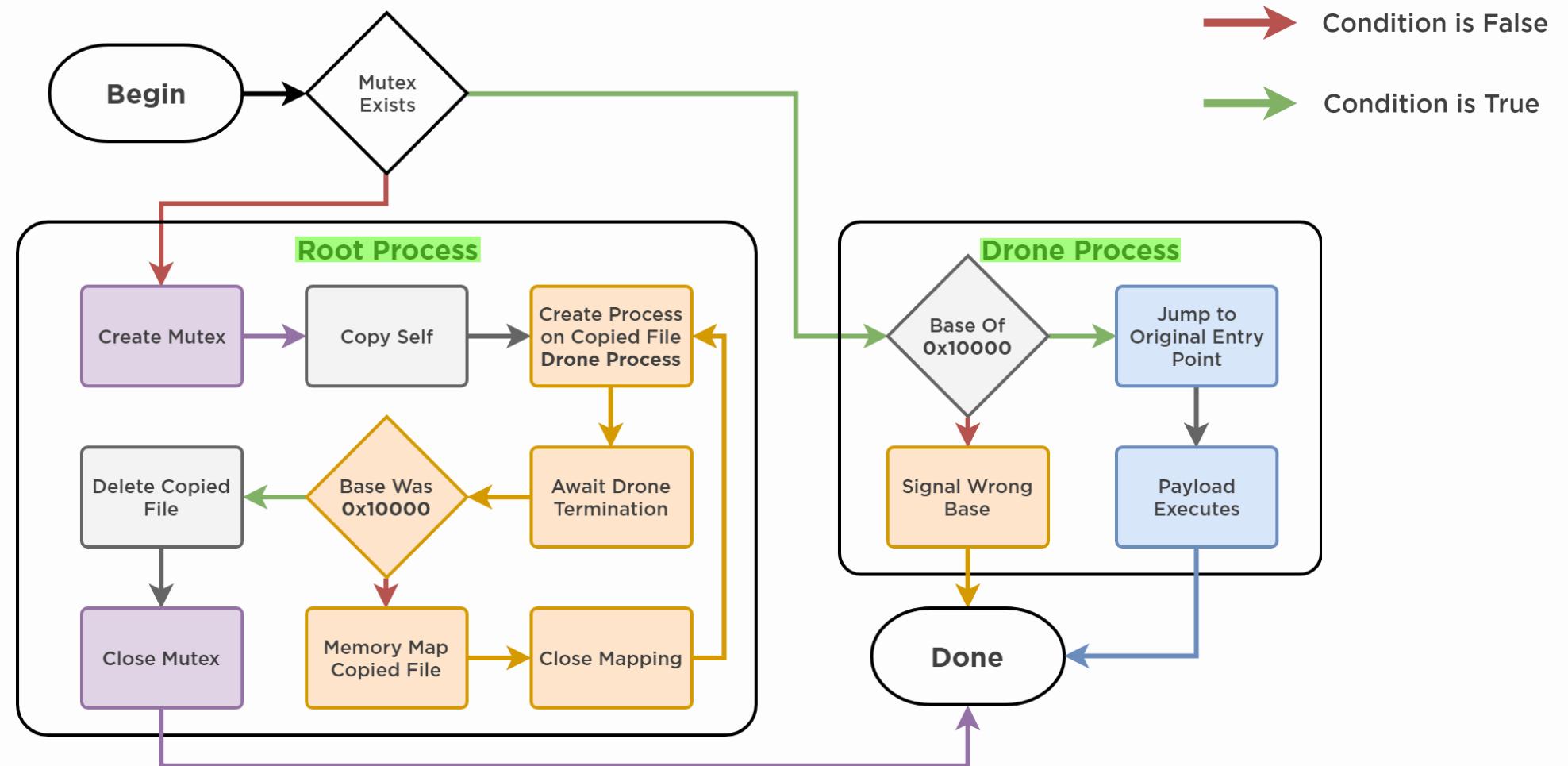
Preselection via File Mapping Invalidation



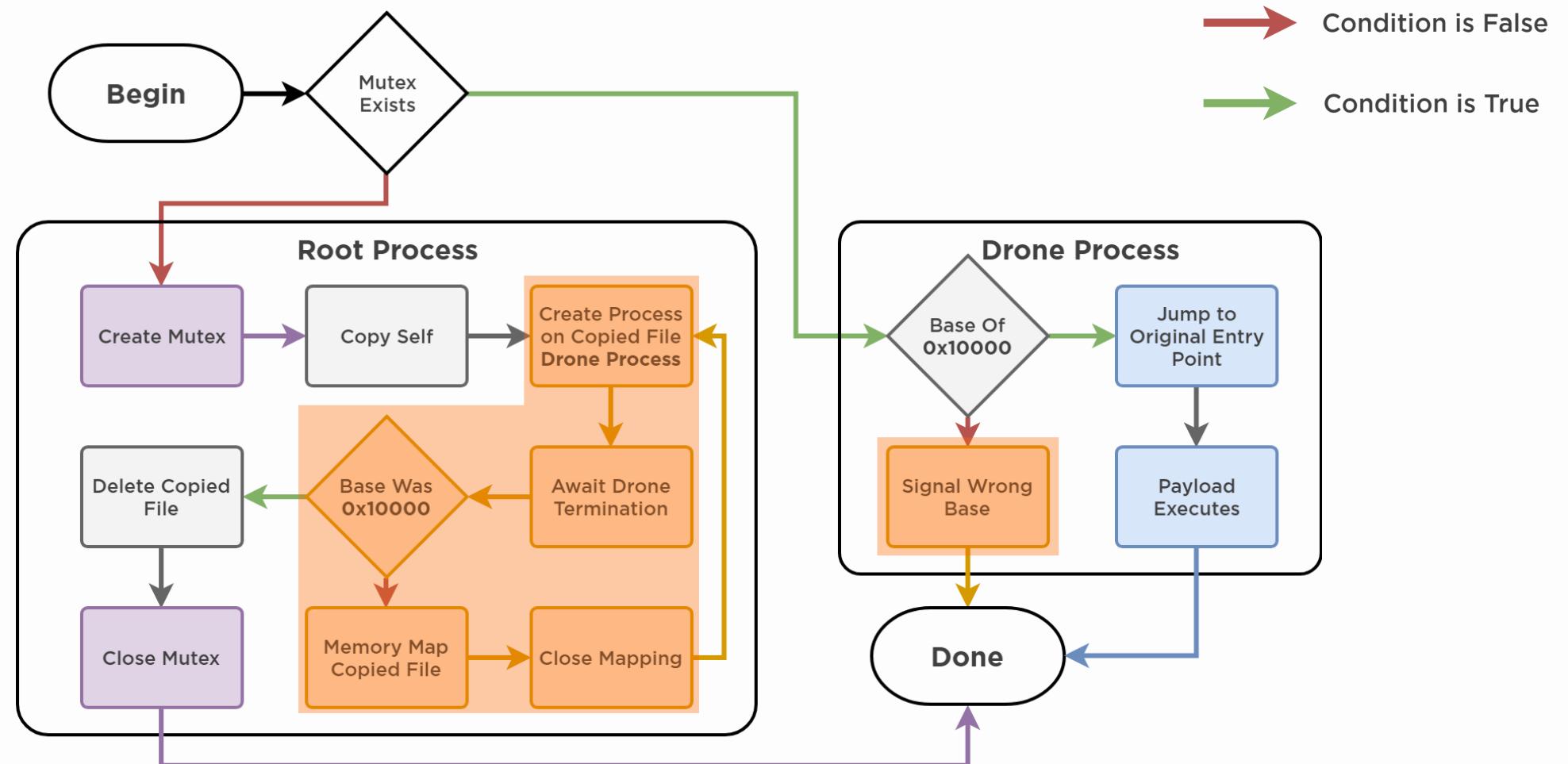
Preselection via File Mapping Invalidation



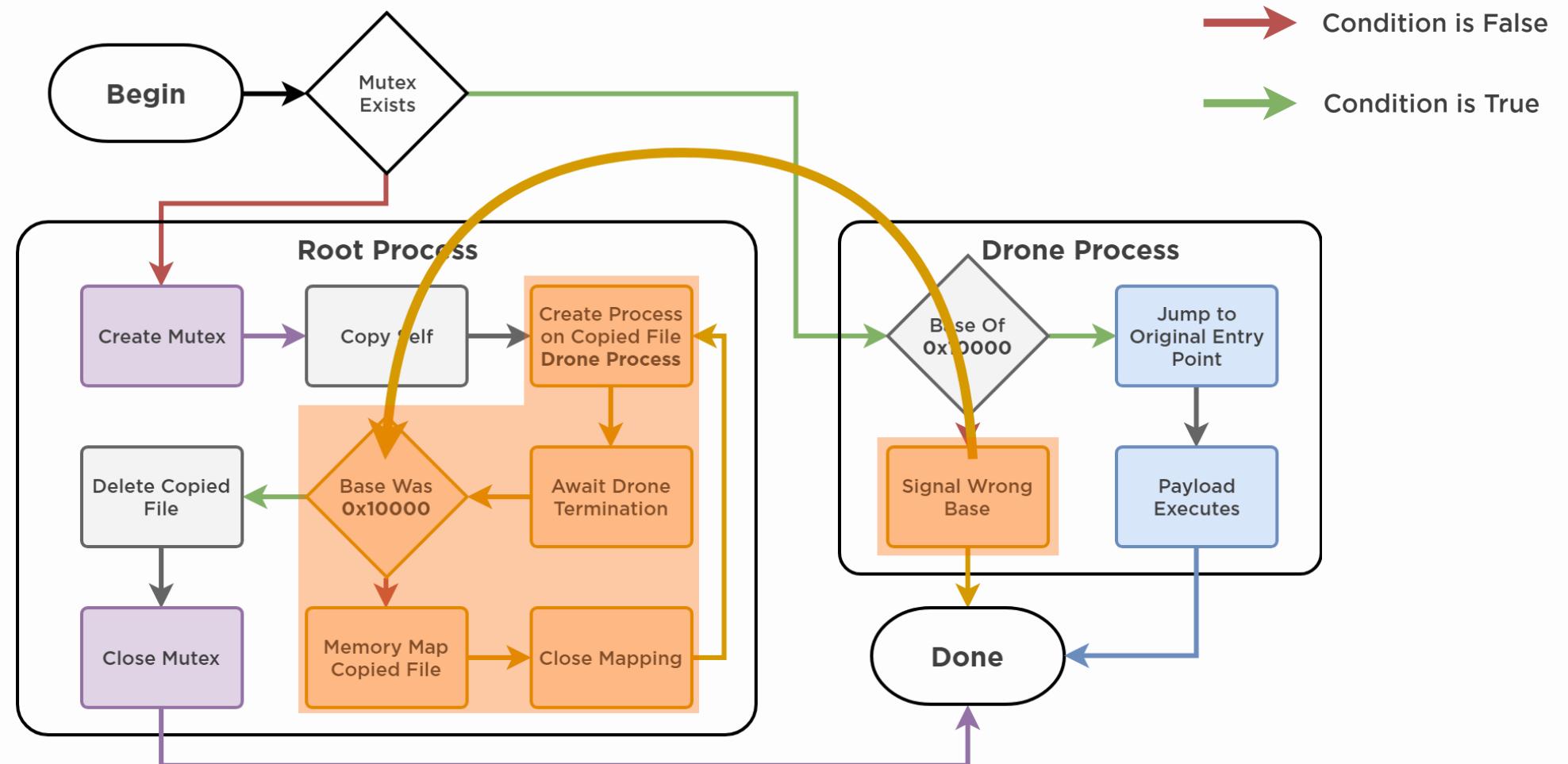
Preselection via File Mapping Invalidation



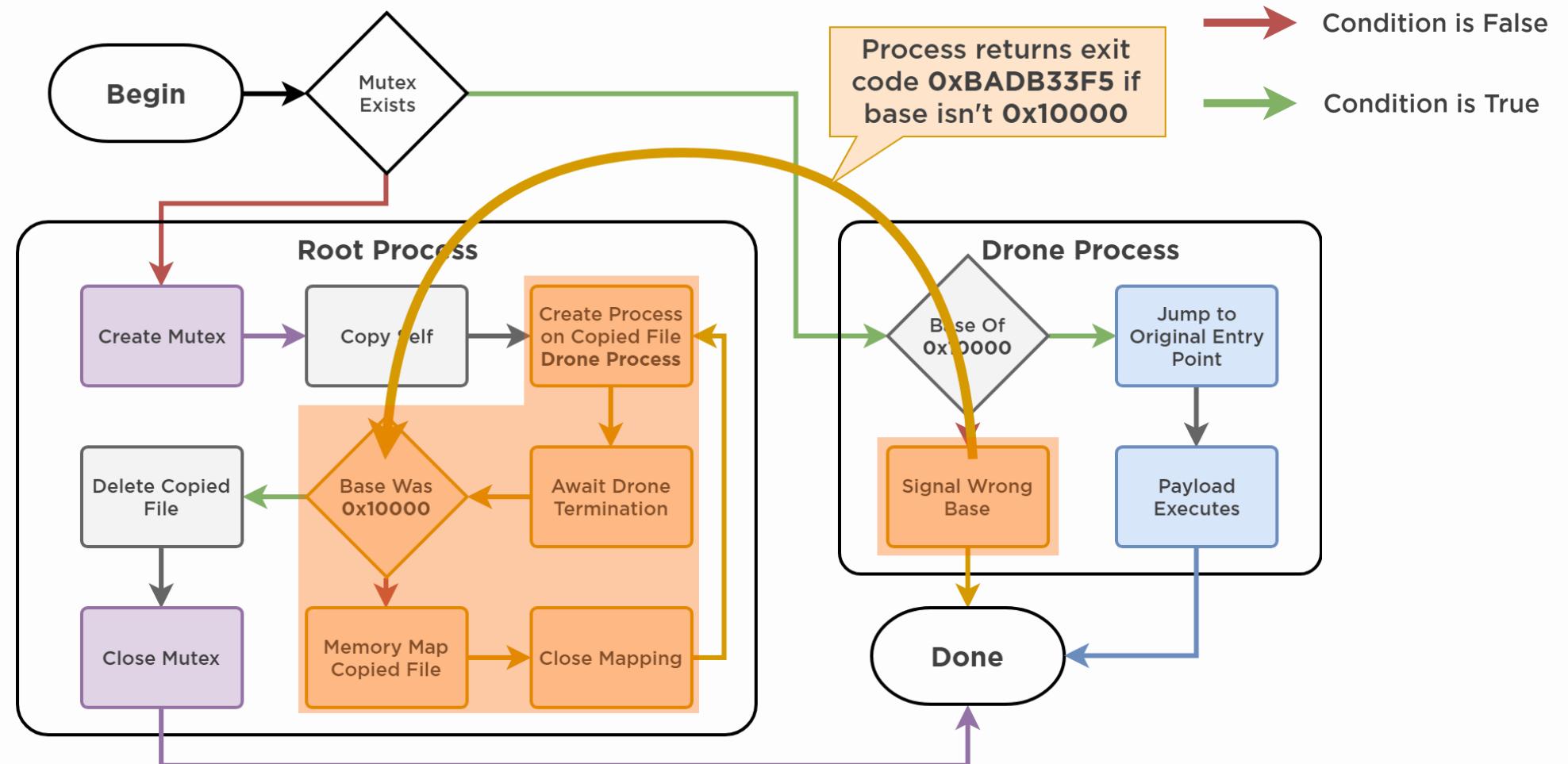
Preselection via File Mapping Invalidation



Preselection via File Mapping Invalidation



Preselection via File Mapping Invalidation



Weaponization

Weaponization

The tool must:

Weaponization

The tool must:

- Create a new section with enough room for the code

Weaponization

The tool must:

- Create a new section with enough room for the code
- Embed the code inside of this new section

Weaponization

The tool must:

- Create a new section with enough room for the code
- Embed the code inside of this new section
- Inform the embedded code of the true `EntryPoint`

Weaponization

The tool must:

- Create a new section with enough room for the code
- Embed the code inside of this new section
- Inform the embedded code of the true `EntryPoint`
- Overwrite `EntryPoint` to point to the embedded code

Weaponization

The tool must:

- Create a new section with enough room for the code
- Embed the code inside of this new section
- Inform the embedded code of the true `EntryPoint`
- Overwrite `EntryPoint` to point to the embedded code



Weaponization

The tool must:

- Create a new section with enough room for the code
- Embed the code inside of this new section
- Inform the embedded code of the true `EntryPoint`
- Overwrite `EntryPoint` to point to the embedded code



For this to work, the ASLR preselection code must be:

Weaponization

The tool must:

- Create a new section with enough room for the code
- Embed the code inside of this new section
- Inform the embedded code of the true `EntryPoint`
- Overwrite `EntryPoint` to point to the embedded code



For this to work, the ASLR preselection code must be:

- Position-agnostic

Weaponization

The tool must:

- Create a new section with enough room for the code
- Embed the code inside of this new section
- Inform the embedded code of the true `EntryPoint`
- Overwrite `EntryPoint` to point to the embedded code



For this to work, the ASLR preselection code must be:

- Position-agnostic
- Generically embeddable in any PE

Weaponization

The tool must:

- Create a new section with enough room for the code
- Embed the code inside of this new section
- Inform the embedded code of the true `EntryPoint`
- Overwrite `EntryPoint` to point to the embedded code



For this to work, the ASLR preselection code must be:

- Position-agnostic
- Generically embeddable in any PE


```

LDR_GET_PROC(f_GetModuleHandle,
LDR_GET_PROC(f_CreateMutexA,
LDR_GET_PROC(f_GetLastError,
LDR_GET_PROC(f_CloseHandle,
LDR_GET_PROC(f_GetCurrentProcessId,
LDR_GET_PROC(f_GetModuleFileNameW,
LDR_GET_PROC(f_CreateProcessW,
LDR_GET_PROC(f_WaitForSingleObject,
LDR_GET_PROC(f_GetExitCodeProcess,
LDR_GET_PROC(f_CreateFileW,
LDR_GET_PROC(f_CreateFileMappingW,
LDR_GET_PROC(f_MapViewOfFileEx,
LDR_GET_PROC(f_UnmapViewOfFile,
LDR_GET_PROC(f_CopyFileW,
LDR_GET_PROC(f_DeleteFileW,
LDR_GET_PROC(f_lstrcatW,
LDR_GET_PROC(f_ExitProcess,

kernel32Handle, HMODULE(__stdcall *)(PUCHAR), LDR_STR_SIG('G', 'e', 't', 'M', 'o', 'd', 'u', 'l', 'e', 'H', 'a', 'n', 'd', 'l', 'e', 'A'));
kernel32Handle, HANDLE(__stdcall *)(PVOID, BOOL, LPCSTR), LDR_STR_SIG('C', 'r', 'e', 'a', 't', 'e', 'M', 'u', 't', 'e', 'x', 'A'));
kernel32Handle, DWORD(__stdcall *)(), LDR_STR_SIG('G', 'e', 't', 'L', 'a', 's', 't', 'E', 'r', 'o', 'r');
kernel32Handle, BOOL(__stdcall *)(HANDLE), LDR_STR_SIG('C', 'l', 'o', 's', 'e', 'H', 'a', 'n', 'd', 'l', 'e');
kernel32Handle, DWORD(__stdcall *)(), LDR_STR_SIG('G', 'e', 't', 'C', 'u', 'r', 'r', 'e', 'n', 't', 'P', 'r', 'o', 'c', 's', 'I', 'd');
kernel32Handle, DWORD(__stdcall *)(HMODULE, LPWSTR, DWORD), LDR_STR_SIG('G', 'e', 't', 'M', 'o', 'd', 'u', 'l', 'e', 'F', 'i', 'l', 'e', 'N', 'a');
kernel32Handle, BOOL(__stdcall *)(LPCWSTR, LPWSTR, PVOID, PVOID, DWORD, LPVOID, LPSTARTUPINFO, LPPROCESS_INFORMATION), LDR_STR_SI;
kernel32Handle, DWORD(__stdcall *)(HANDLE, DWORD), LDR_STR_SIG('W', 'a', 'i', 't', 'F', 'o', 'r', 'S', 'i', 'n', 'g', 'l', 'e', 'o', 'b', 'i', 'e';
kernel32Handle, BOOL(__stdcall *)(HANDLE, LPDWORD), LDR_STR_SIG('G', 'e', 't', 'E', 'x', 'i', 't', 'C', 'o', 'd', 'e', 'P', 'r', 'o', 'c', 'e';
kernel32Handle, HANDLE(__stdcall *)(LPCWSTR, DWORD, DWORD, LPSECURITY_ATTRIBUTES, DWORD, DWORD, HANDLE), LDR_STR_SIG('c', 'r', 'e', 'a', 't', 'e';
kernel32Handle, HANDLE(__stdcall *)(HANDLE, LPSECURITY_ATTRIBUTES, DWORD, DWORD, DWORD, LPCWSTR), LDR_STR_SIG('C', 'r', 'e', 'a', 't', 'e', 'F', 'i', 'l', 'e', 'N', 'a');
kernel32Handle, LPVOID(__stdcall *)(HANDLE, DWORD, DWORD, DWORD, SIZE_T, LPVOID), LDR_STR_SIG('M', 'a', 'p', 'v', 'i', 'w', 'o', 'f', 'F', 'i', 'l', 'e');
kernel32Handle, BOOL(__stdcall *)(LPCVOID), LDR_STR_SIG('U', 'n', 'm', 'a', 'p', 'v', 'i', 'w', 'o', 'f', 'F', 'i', 'l', 'e');
kernel32Handle, BOOL(__stdcall *)(LPCWSTR, LPCWSTR, BOOL), LDR_STR_SIG('C', 'o', 'p', 'y', 'F', 'i', 'l', 'e', 'W');
kernel32Handle, BOOL(__stdcall *)(LPCWSTR), LDR_STR_SIG('D', 'e', 'l', 'e', 't', 'e', 'F', 'i', 'l', 'e', 'W');
kernel32Handle, LPWSTR(__stdcall *)(LPWSTR, LPCWSTR), LDR_STR_SIG('l', 's', 't', 'r', 'c', 'a', 't', 'e', 'W');
kernel32Handle, VOID(__stdcall *)(DWORD), LDR_STR_SIG('E', 'x', 'i', 't', 'P', 'r', 'o', 'c', 'e', 's', 't');

// define strings
DEFINE_STR(const char*, mutexName, 'r', 'e', 'l', 'o', 'c', ' ', 'p', 'a', 'c', 'k', ' ', 'm', 'u', 't');
DEFINE_STR(const wchar_t*, copySuffix, '2', 0x00, '.', 0x00, 'e', 0x00, 'x', 0x00, 'e', 0x00);

```

Weaponization

Weaponization

It works!

Weaponization

It works!

Caveats

Weaponization

It works!

Caveats

- It can be slow, averaging of 200 iterations to land

Weaponization

It works!

Caveats

- It can be slow, averaging of 200 iterations to land
- Imports can't be obfuscated

Weaponization

It works!

Caveats

- It can be slow, averaging of 200 iterations to land
- Imports can't be obfuscated

Advantages

Weaponization

It works!

Caveats

- It can be slow, averaging of 200 iterations to land
- Imports can't be obfuscated

Advantages

- Base can be anything, not just 0x00010000!

Weaponization

It works!

Caveats

- It can be slow, averaging of 200 iterations to land
- Imports can't be obfuscated

Advantages

- Base can be anything, not just 0x00010000!
- As a side effect, some form of symbolic execution is needed to discover the intended base in order to fix up the file for analysis.

ROUND TWO

FIGHT

Use Cases

Use Cases

- Annoying analysts

Use Cases

- Annoying analysts
- Breaking automated static analysis systems

Use Cases

- Annoying analysts
- Breaking automated static analysis systems
- Breaking tools

Use Cases

- Annoying analysts
- Breaking automated static analysis systems
- Breaking tools
- Breaking AV Parsers

Potential Improvements

Potential Improvements

- More obfuscations

Potential Improvements

- More obfuscations
 - New targets

Potential Improvements

- More obfuscations
 - New targets
 - Multiple passes

Potential Improvements

- More obfuscations
 - New targets
 - Multiple passes
- Header Scrambling

Potential Improvements

- More obfuscations
 - New targets
 - Multiple passes
- Header Scrambling
- Combining with runtime packers

Potential Improvements

- More obfuscations
 - New targets
 - Multiple passes
- Header Scrambling
- Combining with runtime packers
- Support for 64bit binaries

Potential Improvements

- More obfuscations
 - New targets
 - Multiple passes
- Header Scrambling
- Combining with runtime packers
- Support for 64bit binaries
- Support for DLLs

Potential Improvements

- More obfuscations
 - New targets
 - Multiple passes
- Header Scrambling
- Combining with runtime packers
- Support for 64bit binaries
- Support for DLLs
- Selective obfuscations

THE END

Find Me

<https://nickcano.com>

<https://github.com/nickcano>

<https://twitter.com/nickcano93>

<https://nostarch.com/gamehacking>

<https://pluralsight.com/authors/nick-cano>

Resources

<https://msdn.microsoft.com/en-us/library/ms809762.aspx>

<https://github.com/corkami/pocs/tree/master/PE>

Source Code

<https://github.com/nickcano/RelocBonus>

<https://github.com/nickcano/RelocBonusSlides>

