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?

```
00219E85  E8 8692F2FF      CALL Tibia.00143110
00219E8A  68 482A6900      PUSH Tibia.00692A48
00219E8F  8D85 40FFFFFF    LEA EAX,DWORD PTR SS:[EBP-C0]
00219E95  50               PUSH EAX
00219E96  E8 3A2D2B00      CALL Tibia.004CCBD5
00219E9B  CC               INT3
00219E9C  CC               INT3
00219E9D  CC               INT3
00219E9E  CC               INT3
00219E9F  CC               INT3
00219EA0  55               PUSH EBP
00219EA1  8BEC             MOV EBP,ESP
00219EA3  6A FF            PUSH -1
00219EA5  68 FA355400      PUSH Tibia.005435FA
00219EAA  64:A1 00000000   MOV EAX,DWORD PTR FS:[0]
00219EB0  50               PUSH EAX
00219EB1  81EC 00010000    SUB ESP,100
00219EB7  A1 80836900      MOV EAX,DWORD PTR DS:[698380]
00219EBC  33C5             XOR EAX,EBP
00219EBE  8945 EC          MOV DWORD PTR SS:[EBP-14],EAX
00219EC1  53               PUSH EBX
00219EC2  56               PUSH ESI
00219EC3  57               PUSH EDI
00219EC4  50               PUSH EAX
00219EC5  8D45 F4          LEA EAX,DWORD PTR SS:[EBP-C]
00219EC8  64:A3 00000000   MOV DWORD PTR FS:[0],EAX
00219ECE  8965 F0          MOV DWORD PTR SS:[EBP-10],ESP
00219ED1  8BF9             MOV EDI,ECX
00219ED3  C745 FC 0000000( MOV DWORD PTR SS:[EBP-4],0
00219EDA  BE 29808500      MOV ESI,Tibia.00858029
00219EDF  8985 08FFFFFF    MOV DWORD PTR SS:[EBP-F8],ESI
00219EE5  83FF 03          CMP EDI,3
00219EE8  0F87 C6000000    JA Tibia.00219FB4
00219EEE  FF24BD F0A12100  JMP DWORD PTR DS:[EDI*4+21A1F0]
00219EF5  BE A0249700      MOV ESI,Tibia.008724A0          ASCII "You are using an outdated client which is no longer supp
00219EFA  8985 08FFFFFF    MOV DWORD PTR SS:[EBP-F8],ESI
00219F00  B8 88258700      MOV EAX,Tibia.00872588          ASCII "Error"
00219F05  EB 10            JMP SHORT Tibia.00219F17
00219F07  BE 80238700      MOV ESI,Tibia.00872380          ASCII "You are using an outdated client which is no longer supp
00219F0C  8985 08FFFFFF    MOV DWORD PTR SS:[EBP-F8],ESI
00219F12  B8 94248700      MOV EAX,Tibia.00872494          ASCII "Information"
00219F17  50               PUSH EAX
00219F18  8D4D BC          LEA ECX,DWORD PTR SS:[EBP-44]
00219F1B  E8 4006F2FF      CALL Tibia.0013AA60
00219F20  C645 FC 01       MOV BYTE PTR SS:[EBP-4],1
00219F24  83FF 03          CMP EDI,3
00219F27  74 0A            JE SHORT Tibia.00219F33
00219F29  83FF 01          CMP EDI,1
00219F2C  B9 5C7F7000      MOV ECX,Tibia.00707F5C
00219F31  75 05            JNZ SHORT Tibia.00219F38
00219F33  B9 847F7000      MOV ECX,Tibia.00707F84
00219F38  8B01             MOV EAX,DWORD PTR DS:[ECX]
00219F3A  8945 D8          MOV DWORD PTR SS:[EBP-28],EAX
00219F3D  8B41 04          MOV EAX,DWORD PTR DS:[ECX+4]
00219F40  8945 DC          MOV DWORD PTR SS:[EBP-24],EAX
00219F43  8B41 08          MOV EAX,DWORD PTR DS:[ECX+8]
00219F46  8945 E0          MOV DWORD PTR SS:[EBP-20],EAX
00219F49  8B41 0C          MOV EAX,DWORD PTR DS:[ECX+C]
00219F4C  8945 E4          MOV DWORD PTR SS:[EBP-1C],EAX
00219F4F  8B41 10          MOV EAX,DWORD PTR DS:[ECX+10]
```

Why Attack Relocations?

```
00219E85  E8 8692F2FF     CALL Tibia.00143110
00219E8A  68 482A6900     PUSH Tibia.00692A48
00219E8F  8D85 40FFFFFF   LEA EAX,DWORD PTR SS:[EBP-C0]
00219E95  50              PUSH EAX
00219E96  E8 3A2D2B00     CALL Tibia.004CCBD5
00219E9B  CC              INT3
00219E9C  CC              INT3
00219E9D  CC              INT3
00219E9E  CC              INT3
00219E9F  CC              INT3
00219EA0  55              PUSH EBP
00219EA1  8BEC            MOV EBP,ESP
00219EA3  6A FF           PUSH -1
00219EA5  68 FA355400     PUSH Tibia.005435FA
00219EAA  64:A1 00000000  MOV EAX,DWORD PTR FS:[0]
00219EB0  50              PUSH EAX
00219EB1  81EC 00010000   SUB ESP,100
00219EB7  A1 80836900     MOV EAX,DWORD PTR DS:[698380]
00219EBC  33C5            XOR EAX,EBP
00219EBE  8945 EC         MOV DWORD PTR SS:[EBP-14],EAX
00219EC1  53              PUSH EBX
00219EC2  56              PUSH ESI
00219EC3  57              PUSH EDI
00219EC4  50              PUSH EAX
00219EC5  8D45 F4         LEA EAX,DWORD PTR SS:[EBP-C]
00219EC8  64:A3 00000000  MOV DWORD PTR FS:[0],EAX
00219ECE  8965 F0         MOV DWORD PTR SS:[EBP-10],ESP
00219ED1  8BF9            MOV EDI,ECX
00219ED3  C745 FC 0000000 MOV DWORD PTR SS:[EBP-4],0
00219EDA  BE 29008500     MOV ESI,Tibia.00850029
00219EDF  89B5 08FFFFFF   MOV DWORD PTR SS:[EBP-F8],ESI
00219EE5  83FF 03         CMP EDI,3
00219EE8  0F87 C6000000   JA Tibia.00219FB4
00219EEE  FF24BD F0A12100 JMP DWORD PTR DS:[EDI*4+21A1F0]
00219EF5  BE A0248700     MOV ESI,Tibia.008724A0
00219EFA  89B5 08FFFFFF   MOV DWORD PTR SS:[EBP-F8],ESI
00219F00  B8 88258700     MOV EAX,Tibia.00872588
00219F05  EB 10           JMP SHORT Tibia.00219F17
00219F07  BE 80238700     MOV ESI,Tibia.00872380
00219F0C  89B5 08FFFFFF   MOV DWORD PTR SS:[EBP-F8],ESI
00219F12  B8 94248700     MOV EAX,Tibia.00872494
00219F17  50              PUSH EAX
00219F18  8D4D BC         LEA ECX,DWORD PTR SS:[EBP-44]
00219F1B  E8 400BF2FF     CALL Tibia.0013AA60
00219F20  C645 FC 01      MOV BYTE PTR SS:[EBP-4],1
00219F24  83FF 03         CMP EDI,3
00219F27  74 0A           JE SHORT Tibia.00219F33
00219F29  83FF 01         CMP EDI,1
00219F2C  B9 5C7F7000     MOV ECX,Tibia.00707F5C
00219F31  75 05           JNZ SHORT Tibia.00219F38
00219F33  B9 847F7000     MOV ECX,Tibia.00707F84
00219F38  8B01            MOV EAX,DWORD PTR DS:[ECX]
00219F3A  8945 D8         MOV DWORD PTR SS:[EBP-28],EAX
00219F3D  8B41 04         MOV EAX,DWORD PTR DS:[ECX+4]
00219F40  8945 DC         MOV DWORD PTR SS:[EBP-24],EAX
00219F43  8B41 08         MOV EAX,DWORD PTR DS:[ECX+8]
00219F46  8945 E0         MOV DWORD PTR SS:[EBP-20],EAX
00219F49  8B41 0C         MOV EAX,DWORD PTR DS:[ECX+C]
00219F4C  8945 E4         MOV DWORD PTR SS:[EBP-1C],EAX
00219F4F  8B41 10         MOV EAX,DWORD PTR DS:[ECX+10]
```

```
00219E85  E8 8692F2FF     CALL Tibia.00143110
00219E8A  68 482A6900     PUSH Tibia.00692A48
00219E8F  8D85 40FFFFFF   LEA EAX,DWORD PTR SS:[EBP-C0]
00219E95  50              PUSH EAX
00219E96  E8 3A2D2B00     CALL Tibia.004CCBD5
00219E9B  CC              INT3
00219E9C  CC              INT3
00219E9D  CC              INT3
00219E9E  CC              INT3
00219E9F  CC              INT3
00219EA0  55              PUSH EBP
00219EA1  8BEC            MOV EBP,ESP
00219EA3  6A FF           PUSH -1
00219EA5  68 FA355400     PUSH Tibia.005435FA
00219EAA  64:A1 00000000  MOV EAX,DWORD PTR FS:[0]
00219EB0  50              PUSH EAX
00219EB1  81EC 00010000   SUB ESP,100
00219EB7  A1 80836900     MOV EAX,DWORD PTR DS:[698380]
00219EBC  33C5            XOR EAX,EBP
00219EBE  8945 EC         MOV DWORD PTR SS:[EBP-14],EAX
00219EC1  53              PUSH EBX
00219EC2  56              PUSH ESI
00219EC3  57              PUSH EDI
00219EC4  50              PUSH EAX
00219EC5  8D45 F4         LEA EAX,DWORD PTR SS:[EBP-C]
00219EC8  64:A3 00000000  MOV DWORD PTR FS:[0],EAX
00219ECE  8965 F0         MOV DWORD PTR SS:[EBP-10],ESP
00219ED1  8BF9            MOV EDI,ECX
00219ED3  C745 FC 0000000 MOV DWORD PTR SS:[EBP-4],0
00219EDA  BE 29008500     MOV ESI,Tibia.00850029
00219EDF  89B5 08FFFFFF   MOV DWORD PTR SS:[EBP-F8],ESI
00219EE5  83FF 03         CMP EDI,3
00219EE8  0F87 C6000000   JA Tibia.00219FB4
00219EEB  90              NOP
00219EEC  90              NOP
00219EF0  90              NOP
00219EF1  90              NOP
00219EF2  90              NOP
00219EF3  90              NOP
00219EF4  90              NOP
00219EF5  BE A0248700     MOV ESI,Tibia.008724A0          ASCII "You are using an outdated client which is no longer supp
00219EFA  89B5 08FFFFFF   MOV DWORD PTR SS:[EBP-F8],ESI
00219F00  B8 88258700     MOV EAX,Tibia.00872588          ASCII "Error"
00219F05  EB 10           JMP SHORT Tibia.00219F17
00219F07  BE 80238700     MOV ESI,Tibia.00872380          ASCII "You are using an outdated client which is no longer supp
00219F0C  89B5 08FFFFFF   MOV DWORD PTR SS:[EBP-F8],ESI
00219F12  B8 94248700     MOV EAX,Tibia.00872494          ASCII "Information"
00219F17  50              PUSH EAX
00219F18  8D4D BC         LEA ECX,DWORD PTR SS:[EBP-44]
00219F1B  E8 400BF2FF     CALL Tibia.0013AA60
00219F20  C645 FC 01      MOV BYTE PTR SS:[EBP-4],1
00219F24  83FF 03         CMP EDI,3
00219F27  74 0A           JE SHORT Tibia.00219F33
00219F29  83FF 01         CMP EDI,1
00219F2C  B9 5C7F7000     MOV ECX,Tibia.00707F5C
00219F31  75 05           JNZ SHORT Tibia.00219F38
00219F33  B9 847F7000     MOV ECX,Tibia.00707F84
00219F38  8B01            MOV EAX,DWORD PTR DS:[ECX]
00219F3A  8945 D8         MOV DWORD PTR SS:[EBP-28],EAX
00219F3D  8B41 04         MOV EAX,DWORD PTR DS:[ECX+4]
```

Why Attack Relocations?

Why Attack Relocations?

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  .  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
```

PE Header Sidebar

PE Header Sidebar

PE Header Sidebar

PE Header Sidebar

PE Header Sidebar

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?

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)

	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?

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.

	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]

How Do Relocations Work?

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.

Each entry has an offset into a block, which is essentially the byte index of the value to relocate.

	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]

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

Targets For Relocation Obfuscation

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 containg 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 containg 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 containg 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

```
nick@neutrino: ~
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
nick@neutrino:~$
```

A New Hope

```
nick@neutrino: ~
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
nick@neutrino:~$
```

A New Hope

```
nick@neutrino: ~
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\basetest.exe
0x0000000000f0000  0x6000    C:\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:~$
```

A New Hope

```
nick@neutrino: ~
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  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
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
nick@neutrino:~$
```

A New Hope

```
nick@neutrino: ~
nick@neutrino:~$ listdlls.exe | grep -v Command | grep -v pid | grep basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  0x6000    C:\basetest.exe
0x00000000000f0000  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
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  0x6000    C:\basetest.exe
0x0000000000a00000  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

```
// HELPERS FOR VARIADIC MACROS
#define EXPAND(x) x
#define __ARG_N(\
    _1, _2, _3, _4, _5, _6, _7, _8, _9, _10, \
    _11, _12, _13, _14, _15, _16, _17, _18, _19, _20, \
    _21, _22, _23, _24, _25, _26, _27, _28, _29, _30, \
    _31, _32, _33, _34, _35, _36, _37, _38, _39, _40, \
    _41, _42, _43, _44, _45, _46, _47, _48, _49, _50, \
    _51, _52, _53, _54, _55, _56, _57, _58, _59, _60, \
    _61, _62, _63,N,...) N
#define __ARGS_R() \
    ,9,8,7,6,5,4,3,2,1,0, \
    ,9,8,7,6,5,4,3,2,1,0, \
    ,9,8,7,6,5,4,3,2,1,0, \
    ,9,8,7,6,5,4,3,2,1,0, \
    ,9,8,7,6,5,4,3,2,1,0, \
    ,9,8,7,6,5,4,3,2,1,0, \
    ,9,8,7,6,5,4,3,2,1,0,
#define __ARG_1_(...) EXPAND(__ARG_N(__VA_ARGS__))
#define __ARG__(...) __ARG_1_(__VA_ARGS__,__ARGS_R())
#define __ARG_PLUS_ONE_(...) __ARG_(0,0,0,0,0,0,0,0,0,0,__VA_ARGS__)

#define __VFUNC_(name, n) name##n
#define _VFUNC(name, n) __VFUNC_(name, n)
#define VFUNC(func, ...) EXPAND(_VFUNC(func, __ARG_(__VA_ARGS__)) (__VA_ARGS__))

// DATAS/STRINGS IN A NAME
#define LOWER(c) (((((char)(c) <= 0x5A && ((char)(c) >= 0x41) ? ((char)(c) + 0x20 : ((char)(c))
#define LDR_GET_MODULE(varName, len, firstChar, lastCharBeforeExt) \
    PVOID varName = NULL; \
    auto module1*varName = (PLDR_MODULE)LoaderData->InLoadOrderModuleList.Flink; \
    do { \
        auto checkLen = module1*varName->FullDllName.Length / 2; \
        /*printf("%s %d %c %c %c
```

```
LDR_GET_PROC(f_GetModuleHandle,      kernel32Handle, HMODULE(__stdcall *)(PUCHAR), LDR_STR_SIG('G', 'e', 't', 'M', 'o', 'd', 'u', 'l', 'e', 'H', 'a', 'n', 'd', 'l', 'e', 'A'));
LDR_GET_PROC(f_CreateMutexA,         kernel32Handle, HANDLE(__stdcall *)(PVOID, BOOL, LPCSTR), LDR_STR_SIG('C', 'r', 'e', 'a', 't', 'e', 'M', 'u', 't', 'e', 'x', 'A'));
LDR_GET_PROC(f_GetLastError,         kernel32Handle, DWORD(__stdcall *)(), LDR_STR_SIG('G', 'e', 't', 'L', 'a', 's', 't', 'E', 'r', 'r', 'o', 'r'));
LDR_GET_PROC(f_CloseHandle,          kernel32Handle, BOOL(__stdcall *)(HANDLE), LDR_STR_SIG('C', 'l', 'o', 's', 'e', 'H', 'a', 'n', 'd', 'l', 'e'));
LDR_GET_PROC(f_GetCurrentProcessId,  kernel32Handle, DWORD(__stdcall *)(), LDR_STR_SIG('G', 'e', 't', 'C', 'u', 'r', 'r', 'e', 'n', 't', 'P', 'r', 'o', 'c', 'e', 's', 's', 'I', 'd'))
LDR_GET_PROC(f_GetModuleFileNameW,   kernel32Handle, DWORD(__stdcall *)(HMODULE, LPWSTR, DWORD), LDR_STR_SIG('G', 'e', 't', 'M', 'o', 'd', 'u', 'l', 'e', 'F', 'i', 'l', 'e', 'N', 'a
LDR_GET_PROC(f_CreateProcessW,       kernel32Handle, BOOL(__stdcall *)(LPCWSTR, LPWSTR, PVOID, PVOID, BOOL, DWORD, LPVOID, LPCWSTR, LPSTARTUPINFOW, LPPROCESS_INFORMATION), LDR_STR_SI
LDR_GET_PROC(f_WaitForSingleObject,  kernel32Handle, DWORD(__stdcall *)(HANDLE, DWORD), LDR_STR_SIG('W', 'a', 'i', 't', 'F', 'o', 'r', 'S', 'i', 'n', 'g', 'l', 'e', 'O', 'b', 'j', '
LDR_GET_PROC(f_GetExitCodeProcess,   kernel32Handle, BOOL(__stdcall *)(HANDLE, LPDWORD), LDR_STR_SIG('G', 'e', 't', 'E', 'x', 'i', 't', 'C', 'o', 'd', 'e', 'P', 'r', 'o', 'c', 'e',
LDR_GET_PROC(f_CreateFileW,          kernel32Handle, HANDLE(__stdcall *)(LPCWSTR, DWORD, DWORD, LPSECURITY_ATTRIBUTES, DWORD, DWORD, HANDLE), LDR_STR_SIG('C', 'r', 'e', 'a', 't', 'e
LDR_GET_PROC(f_CreateFileMappingW,   kernel32Handle, HANDLE(__stdcall *)(HANDLE, LPSECURITY_ATTRIBUTES, DWORD, DWORD, DWORD, LPCWSTR), LDR_STR_SIG('C', 'r', 'e', 'a', 't', 'e', 'F',
LDR_GET_PROC(f_MapViewOfFileEx,      kernel32Handle, LPVOID(__stdcall *)(HANDLE, DWORD, DWORD, DWORD, SIZE_T, LPVOID), LDR_STR_SIG('M', 'a', 'p', 'V', 'i', 'e', 'w', 'O', 'f', 'F',
LDR_GET_PROC(f_UnmapViewOfFile,      kernel32Handle, BOOL(__stdcall *)(LPCVOID), LDR_STR_SIG('U', 'n', 'm', 'a', 'p', 'V', 'i', 'e', 'w', 'O', 'f', 'F', 'i', 'l', 'e'));
LDR_GET_PROC(f_CopyFileW,            kernel32Handle, BOOL(__stdcall *)(LPCWSTR, LPCWSTR, BOOL), LDR_STR_SIG('C', 'o', 'p', 'y', 'F', 'i', 'l', 'e', 'W'));
LDR_GET_PROC(f_DeleteFileW,          kernel32Handle, BOOL(__stdcall *)(LPCWSTR), LDR_STR_SIG('D', 'e', 'l', 'e', 't', 'e', 'F', 'i', 'l', 'e', 'W'));
LDR_GET_PROC(f_lstrcatW,             kernel32Handle, LPWSTR(__stdcall *)(LPWSTR, LPCWSTR), LDR_STR_SIG('l', 's', 't', 'r', 'c', 'a', 't', 'W'));
LDR_GET_PROC(f_ExitProcess,          kernel32Handle, VOID(__stdcall *)(DWORD), LDR_STR_SIG('E', 'x', 'i', 't', 'P', 'r', 'o', 'c', 'e', 's', 's'));

// 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

Source Code

https://github.com/nickcano/RelocBonus

https://github.com/nickcano/RelocBonusSlides

Resources

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

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