More Adventures In Format Strings

deanx

Portcullis Computer Security Limited

outline

1. what?
2. pownage
   - heap smash comparison
   - luxuries of the format string
   - PLT Trampoline
3. optimisation
4. demo

what?

What$_{(not)}$?

Smashing the Heap

- Do some funky stuff (DL Style) and call free

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes
- Where?

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes
- Where?
    - PLT

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes
- Where?
  - PLT
- Why?

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes
- Where?
  - PLT
- Why?
  - It's **rw**
  - It's **static**
  - It **will** get called again

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes
- Where?
    - PLT
- Why?
    - It's **rw**
    - It's **static**
    - It **will** get called again
- What?

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes
- Where?
  - PLT
- Why?
  - It's **rw**
  - It's **static**
  - It **will** get called again
- What?
  - Pointer to trampoline that JMP's *edi
  - Where *edi contains your shellcode

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes
- Where?
  - PLT
- Why?
  - It's **rw**
  - It's **static**
  - It **will** get called again
- What?
  - Pointer to trampoline that JMP's *edi
  - Where *edi contains your shellcode
- Problem: What if you have no appropriate registers?

Smashing the Heap

- Do some funky stuff (DL Style) and call free
- Free overwrites 4 bytes
- Where?
  - PLT
- Why?
  - It's **rw**
  - It's **static**
  - It **will** get called again
- What?
  - Pointer to trampoline that JMP's *edi
  - Where *edi contains your shellcode
- Problem: What if you have no appropriate registers?
- Solution: ?

Luxuries of the Format String

- Arbitrary Memory Overwrite

Luxuries of the Format String

- Arbitrary Memory Overwrite
- Non-contiguous Overwrite

Luxuries of the Format String

- Arbitrary Memory Overwrite
- Non-contiguous Overwrite
- Exploit: Follow Heap Smash Just Path

Luxuries of the Format String

- Arbitrary Memory Overwrite
- Non-contiguous Overwrite
- Exploit: Follow Heap Smash Just Path
- Problem: What if you have no appropriate registers?

Luxuries of the Format String

- Arbitrary Memory Overwrite
- Non-contiguous Overwrite
- Exploit: Follow Heap Smash Just Path
- Problem: What if you have no appropriate registers?
  - Again Jmp *edi will fail

Luxuries of the Format String

- Arbitrary Memory Overwrite
- Non-contiguous Overwrite
- Exploit: Follow Heap Smash Just Path
- Problem: What if you have no appropriate registers?
    - Again Jmp *edi will fail
- Solution: Rewrite a register

How to change a Register

- Use the format string

How to change a Register

- Use the format string
- Write a small shellcode to the PLT

How to change a Register

- Use the format string
- Write a small shellcode to the PLT
  - lea edi,[edi-2150]
  - jmp *edi

How to change a Register

- Use the format string
- Write a small shellcode to the PLT
  - lea edi,[edi-2150]
  - jmp *edi
- Point a PLT Entry to your chain code

How to change a Register

- Use the format string
- Write a small shellcode to the PLT
  - lea edi,[edi-2150]
  - jmp *edi
- Point a PLT Entry to your chain code
- Now when it runs *edi will contain your long, stage 2 shellcode

How to change a Register

- Use the format string
- Write a small shellcode to the PLT
  - lea edi,[edi-2150]
  - jmp *edi
- Point a PLT Entry to your chain code
- Now when it runs *edi will contain your long, stage 2 shellcode
- Job Done

optimisation

- Why Optimise?

optimisation

- Why Optimise?
  - Format Strings Are inefficient
  - ∼10 bytes in 2 bytes out (buffer space)
  - Large Logs

optimisation

- Why Optimise?
    - Format Strings Are inefficient
    - ∼10 bytes in 2 bytes out (buffer space)
    - Large Logs
- What Can We Do?

optimisation

- Why Optimise?
  - Format Strings Are inefficient
  - $\sim$10 bytes in 2 bytes out (buffer space)
  - Large Logs
- What Can We Do?
  - Write in any order
  - Write in the most efficient order
  - Write \x0a before \x1a before \x2a before \x3a

optimisation

- Why Optimise?
  - Format Strings Are inefficient
  - $\sim$10 bytes in 2 bytes out (buffer space)
  - Large Logs
- What Can We Do?
  - Write in any order
  - Write in the most efficient order
  - Write \x0a before \x1a before \x2a before \x3a
    - 0x3a1a2a0a
    - 0x0a1a2a3a (1byte) or 0x2a0a3a1a (2 bytes)

optimisation

- Why Optimise?
  - Format Strings Are inefficient
  - ∼10 bytes in 2 bytes out (buffer space)
  - Large Logs
- What Can We Do?
  - Write in any order
  - Write in the most efficient order
  - Write \x0a before \x1a before \x2a before \x3a
    - 0x3a1a2a0a
    - 0x0a1a2a3a (1byte) or 0x2a0a3a1a (2 bytes)
- How?

optimisation

- Why Optimise?
  - Format Strings Are inefficient
  - ~10 bytes in 2 bytes out (buffer space)
  - Large Logs
- What Can We Do?
  - Write in any order
  - Write in the most efficient order
  - Write \x0a before \x1a before \x2a before \x3a
    - 0x3a1a2a0a
    - 0x0a1a2a3a (1byte) or 0x2a0a3a1a (2 bytes)
- How?
  - Write Write Address in order
  - Use %x$hn to pick the x'th memory location

demo

demo!

thanks

mu-b
nico
bambam
doc