Computer Security

Software vulnerabilities and defensive programming

Marius Minea

14 October 2015

Simple (classic) buffer overflow

Aleph One, Smashing the stack for fun and profit, Phrack magazine 7(49)

Overflow of *any* buffer placed on stack

unsafe functions: `strcpy`, `strcat`, `scanf` with `%s`
- `gets` deleted from C standard in 2011
- safe alternatives for some

but also "by-hand" overflow of index in (local) array

Reason: low abstraction level of C
- (pointer arithmetic, no objects with size info)

How to protect?

Option: detect change

check if RET address altered *before* function return

Two basic ideas:

How to protect?

Option: detect change
  check if RET address altered *before* function return

Two basic ideas:
Check return address itself ⇒ need copy of correct value
Check bytes next to (before) ret address ⇒ canaries
    terminator canary: 0, CR, LF, EOF
    random canary (don't know ⇒ can't put back)
    random XOR canary (must also know control value)

Who/how/when implements these checks?

How to protect?

Option: hamper execution

Attacker must execute injected code:
Non-executable stack / write XOR execute

Attacker must know *what address* to jump to:
Address Space Layout Randomization

What flexibility does the attacker code have?
Is attack still realistic? For 32-bit vs. 64-bit ?

Advanced attacks: return-into-libc

If you can't execute code on stack, try something else

Typical attack is to call `exec` or some other library function
$\Rightarrow$ instead of *executing code* (`call exec`), put address (and parameters) of libc function on stack, in place of normal ret address

Which protections are effective?

Can chain attacks – put multiple library addresses on stack

Generalize: return-oriented programming

Overwriting a pointer

Function pointers (denote code)
- pointers from `longjmp`
- pointers to user functions
- pointers to library functions (PLT: procedure linkage table)

or usual pointers to data

Attacks might be in two steps:
- a buffer overflow overwrites a pointer (to desired address)
- in later code, this is used to overwrite critical area
  - ret address, PLT, etc.