Buffer Overflows

TDT60
20.09.2005
Motivation – a little bit of history

- 1988 – the Morris worm
- 1995 – Thomas Lopatic, Bugtraq
- 1996 – Elias Levy, ”Smashing the Stack for Fun and Profit”
- 2001 – Code Red
- 2003 – SQLSlammer
- 50-60% of CERT/CC-alerts are related to buffer overflows
What is buffer overflow?

• Buffer – contiguous chunk of the same data type in memory

• Buffer overflow occurs when:
  – Trying to write X bytes of data to an allocated space of Y bytes in memory (where X > Y)
  – Result: the next contiguous Y-X bytes are overwritten
Why are buffer overflows possible?

- Bad language design + poor programming

- Unsafe programming languages
  - No bounds check on arrays and pointer references
  - Developer has to check the bounds
  - Unsafe library operations
Program memory organisation

- **Static memory allocation**
  - BSS (block storage segment), data segment, text segment
    - size fixed before the program ever runs – value of individual variables can be changed.

- **Dynamic memory allocation**
  - Stack – program functions
    - Holds information about the context of the current function call
    - LIFO – push, pop
    - Stack pointer
    - Grows down
  - Heap
    - Used for dynamic allocation of data at run-time
    - New()
    - Grows up
Heap overflows

• Are difficult to exploit because
  – The heap grows upwards in memory address space
  – Knowledge about security critical variables is hard to obtain without access to the source code
  – Has to create a buffer that will overflow the target variable
    • Means the buffer needs to have a lower memory address than the target variable
    • A compiler have no respect to the order of appearance in code – that one variable follows directly after another in the source code does not (necessarily) imply consecutive memory addresses
  – May be forced to overwrite data that are in-between
Stack overflows

• Are easier to exploit because
  – There is always something security critical to overwrite on the stack!
    • The return address of a function
  – Two types of stack-allocated data:
    • Nonstatic local variables
    • Parameters to functions
  – Can only overflow items with a lower memory address than the return address
Possible consequences of buffer overflow

- Consequence depends on where the data flows..
- Can reset access flags
- Stack smashing
  - Place some attack code somewhere in memory and overwrite the stack in such a way that control gets passed to the attack code
- Primary goal is often to get an interactive session on the target machine – with the privileges of the program being attacked (preferably root).
- Suid – privilege escalation
  - Suid root programs are often the target of an attack
Defending against buffer overflows

- Defensive programming – avoid unsafe functions – use safe alternatives. Examples:
  - strcpy() – use strncpy()
  - strcat() – use strncat()
  - ...(extensive list are available)
- Choose a safe programming language (if possible)
- Use automatic code-scanning tools like:
  - Libsafe – replaces vulnerable calls with safe versions
- Tools to prevent (stack-based) buffer overflows:
  - StackGuard
  - Stack-Smashing Protector
  - StackGhost
  - Uses canary-values included in stack frames – detects change
Misc

- Windows buffer overflows
  - Many interesting functions are dynamically loaded
  - Requires knowledge of which DLL’s are loaded when your code executes