SJSU SAN JOSÉ STATE UNIVERSITY

Lesson 12 – Other Attacks on Software

Yan Chen

CS166 Fall 2024

- Software Reverse Engineering (SRE)
  - Try to understand a software or modify (“patch”) it
  - Attacker only has exe (no source code, no bytecode)
- Tools needed
  - Disassembler: converts exe to assembly (as best it can)
  - Debugger: dynamically check assembly code (often bundled with a disassembler)
  - Hex editor: to view/modify bits of exe (“patch”)
  - (Optional) Process monitor: check file system activities
  - For course usages, online tools are enough

- Can exploit buffer overflow to get serial number...
  - Disassemble .exe first to find return address of correct result
  - Need some trials & errors to find the length of input that can override return address
- Or disassemble .exe, check if can find the serial number
  - May be near “enter serial number” or “serial number is correct”
- Or patch .exe so it will accept all numbers...
  - Find the address of “serial number is correct”
  - Exam the assembly code, find a way to jump to this address for all inputs (e.g., use a hex editor to change “test” to “or”)

- Impossible to prevent SRE on open system
  - Can only make such attacks more difficult...
- Anti-disassembly: to confuse static view of code
  - Encrypted object code, or false disassembly, etc.
- Anti-debugging: to confuse dynamic view of code
  - Check IsDebuggerPresent(), or use multi-threading, etc.
- Tamper-resistance: check if the code is being changed
  - Make patching more difficult
- Code obfuscation: make code hard to understand
  - Waste attackers' time on analyzing dead code, etc

- Salami attack: “slices off” small amounts of money
  - Each slices is too small for for victim to detect...
  - But will eventually get a lot of money if got many slices
  - Possible for “insiders”
- Example: salami attack by the programmer of a bank
  - Programmer “slices off” any fraction of a cent from an account and puts it in his own account
  - No customer notices missing partial cent
  - Bank may not notice any problem
  - Over time, programmer makes lots of money!

- Salami attacks actually occur in real-world....
  - Programmer added a few cents to every employee payroll tax withholding, but credited the extra money to his own tax
  - Rent-a-car franchise in Florida inflated gas tank capacity to overcharge customers
  - Employee reprogrammed Taco Bell cash register: $2.99 item registered as $0.01 (large “slice” of salami – $2.98!)
  - In LA, four men installed computer chip that overstated amount of gas pumped (except for 5 or 10 gallons – since inspector usually asked for 5 or 10 gallons)

- Linearization attack: make the problem "linear"
  - Not only on software – can apply to many other things
- Example: get serial number of a paid program

```
#include <stdio.h>

int main (int argc, const char *argv[]) {
    int i;
    char serial[9] = "S123N456\n";

    for (i = 0; i < 8; i++) {
        if (argv[1][i] != serial[i]) break;
    }

    if (i == 8) {
        printf("\nSerial number is correct\n\n");
    }
}
```

  - Can recover one character at a time based on "timing"

- Example (continued)
  - Since it breaks once found the first wrong character...
  - Correct number takes longer than incorrect
  - Trudy tries all 1st characters
  - The correct 1st character is the one takes longest
  - Then she keeps the correct 1st and guesses all 2nd characters
  - And so on…
  - Same principle as used in lock picking

Work Factor

- Let's analyze the work factor...
- Suppose serial number is 8 characters and each has 128 possible values
  - Then $128^8 = 2^{56}$ possible serial numbers

    Regular exhaustive key search requires 2^{56} tries in worst case

    And need $2^{56} / 2 = 2^{55}$ tries on average
  - Using the linearization attack, 128 tries for each character...

    So, need $8 * 128 = 2^{10}$ tries in total (worst case)

    Which means, need $2^{10} / 2 = 2^9$ tries on average
  - That's way fewer than a regular exhaustive key search!

Example

- Time bomb: "exploded" after certain time
- Let's see a real-life example...
  - In 1986, Donald Gene Burleson told employer to stop withholding taxes from his paycheck
  - His company refused, so he planned to sue his company
  - He used company time to prepare legal docs
  - Company found out and fired him
  - As a revenge, Burleson had been working on malware…
  - After being fired, his software "time bomb" deleted important company data

Result of the Example

- Company was reluctant to pursue the case
- So Burleson sued company for back pay!
  - Then company finally sued Burleson
- Resulted in a slap on the wrist for attacker
  - In 1988 Burleson fined $11,800
  - Case took years to prosecute…
  - Cost company thousands of dollars…
- One of the first computer crime cases
- Many cases since follow a similar pattern
  - Companies reluctant to prosecute

- Midterm 1 Review
  - Crypto basics
  - Symmetric key crypto
  - Public key crypto
  - Hash function
  - Software insecurity
  - Midterm guide – time, format, etc.

- Salami attack
- Linearization attack
- Time bomb

- Consider the following code.

```
int main (int argc, const char *argv[])
{
   char serial[9] = "S123N456";
   if (strcmp(argv[1], serial) == 0)
   {
      printf("\nSerial number is correct!\n\n");
   }
}
```

  - Will a linearization attack succeed?
  - Why or why not? Hint: how does strcmp compare 2 strings?

- Implement a way to check the serial number that “immunes” to a linearization attack

References

- Stamp, Mark, “Information Security, Principles and Practice, 2nd ed.,” Wiley, New Jersey, USA, 2011