Security in Software Applications

Warm-up

SAPIENZA
UNIVERSITÀ DI ROMA

Daniele Friolo

friolo@di.uniroma1.it

https://danielefriolo.github.io

Prerequisites

- Introductory security course ?
- Elementary OS and DB
- Basic programming skills, in particular
  - C(++)
    - eg. `malloc()`, `free()`, `*(p++)`,
      - `strings` in C using `char*`
  - Java
    - eg. **public, final, private, protected**
  - Bits of PHP and javascript

Sample C Code

```
char* copying_a_string(char* a) {
   char* b = malloc(strlen(a));
   strcpy(b,a);
   return(b);
}
int using_pointer_arithmetic(int pin[]) {
   int sum = 0;
   int *pointer = pin;
   for (int i=0; i<4; i++ ){
    sum = sum + *pointer; pointer++;
   }
   return sum;
}
```

Sample Java Code

```
public int summingAnArray(int[] pin)
            throws NullPointerException,
                   ArrayIndexOutOfBoundsException{

   int sum = 0;
    for (int i=0; i<4; i++ ){
            sum = sum + a[i];
   }
   return sum;
}
```

Sample Java OO Code

```
final class A {
    public final static SOME_CONSTANT 2;
    private B b1, b2;

    protected A ShallowClone(Object o)
                throws ClassCastException {
            x = new(A);
            x.b1 =((A)o).b1;
            x.b2 =((A)o).b2;
            return x;
    }
}
```

Trusting TRUST

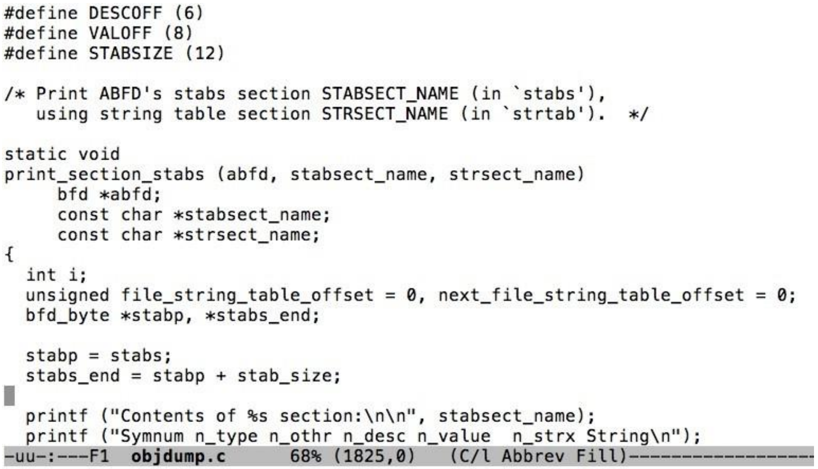
SAPIENZA
UNIVERSITÀ DI ROMA

Trusting Trust

Ken Thompson: Co-Creator of *UNIX* and *C*

Turing Award: 1983

Trusting Trust



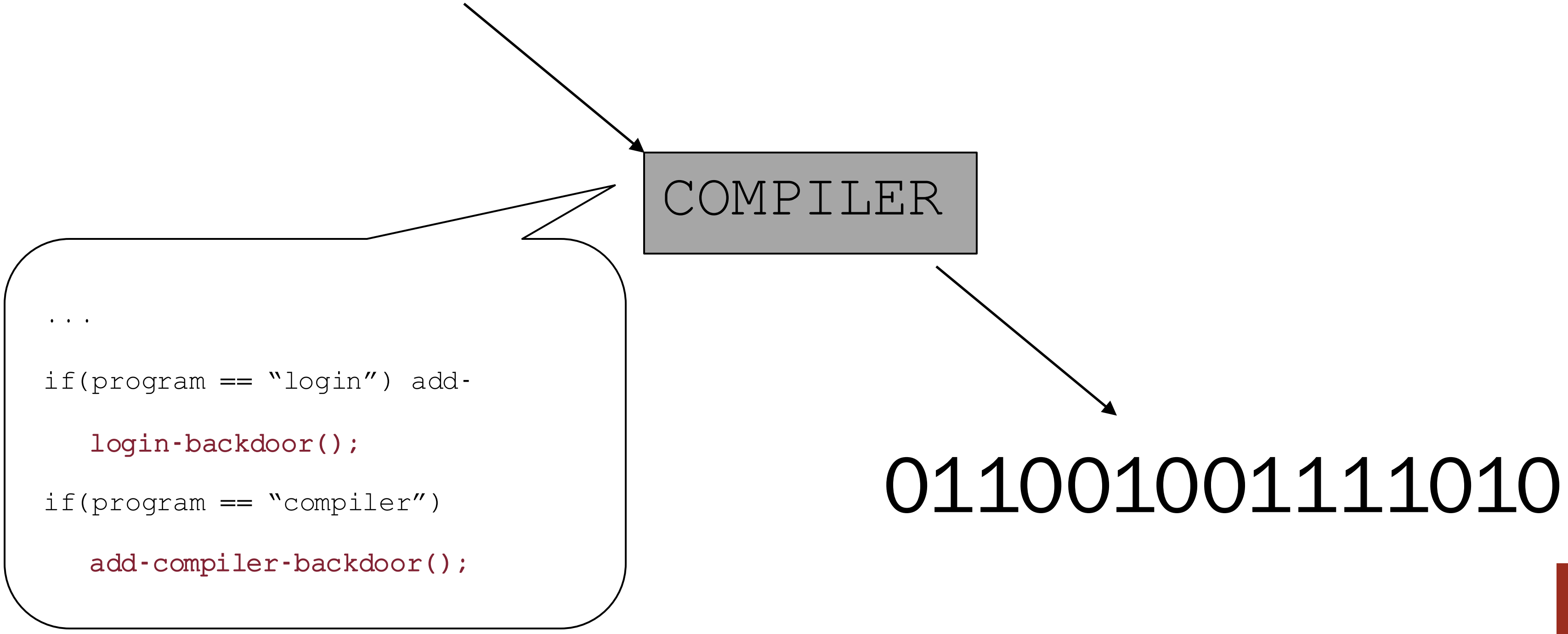
```
#define DESCOFF (6)
#define VALOFF (8)
#define STABSIZE (12)

/* Print ABFD's stabs section STABSECT_NAME (in `stabs'),
   using string table section STRSECT_NAME (in `strtab').  */

static void
print_section_stabs (abfd, stabsect_name, strsect_name)
     bfd *abfd;
     const char *stabsect_name;
     const char *strsect_name;
{
  int i;
  unsigned file_string_table_offset = 0, next_file_string_table_offset = 0;
  bfd_byte *stabp, *stabs_end;

  stabp = stabs;
  stabs_end = stabp + stab_size;

  printf ("Contents of %s section:\n\n", stabsect_name);
  printf ("Symnum n_type n_othr n_desc n_value  n_strx String\n");
-uu-:---F1  objdump.c      68% (1825,0)    (C/l Abbrev Fill)-----------------
```



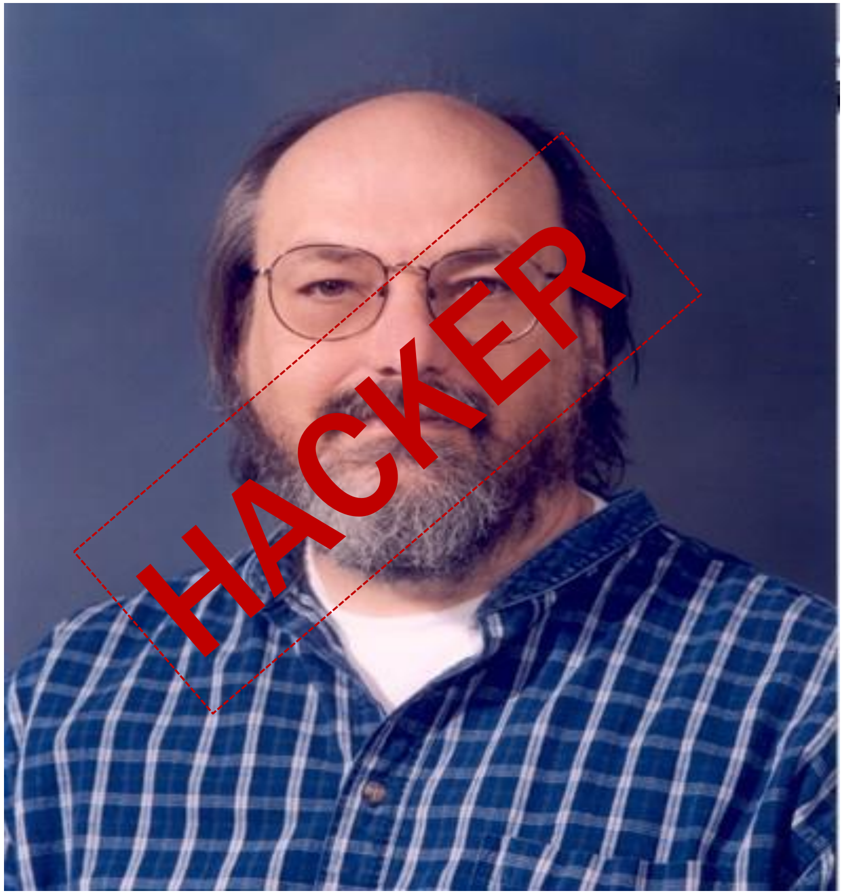
COMPILER

```
...

if(program == "login") add-
   login-backdoor();
if(program == "compiler")
   add-compiler-backdoor();
```

0110010011111010

Trusting Trust



Ken Thompson: Co-Creator of *UNIX* and *C*

Turing Award: 1983

Context

What is a computer system?

- *(Classical) computer*: mainframe, server, desktop
- *Mobile device*: phone, tablets, audio/video player, etc.. . . Up to iot, smart cards, . . .
- *Embedded (networked) systems*: inside a car, a plane, a washing machine, etc.
- *Clouds*
- But also *industrial networks* (ICS, scada), . . . etc.
- And certainly many more !

Two main interesting characteristics:

1. Includes hardware + **software**
2. Open/connected to the **outside world** . . .

Context

What does security mean?

- A set of "high-level" security goals:

CIA = *confidentiality, integrity, availability (+ non repudiation + . . .)*

- Is it specific to the computer system we consider?
- How to deal with "unsecure executions"?

- Something beyond *safety* and *fault-tolerance*:
  - Notion of **intruder**, with specific capabilities
  - Notion of **threats**, with a "threat model"

  There is an "external actor" with an attack objective in mind, and able to elaborate a dedicated strategy to achieve it (not a hazard)

- A definition "by default":
  - *Functional properties*: what the system **should** do
  - *Security properties*: what the system should **not** do, how it should **not** behave

Software Secuirty: an example

Consider 2 programs:

- `Compress`, to compress a file `f`
- `Uncompress`, to uncompress a (compressed) file `C`

Expected behavior (the one we try to validate)

```
Uncompress(Compress(f)) = f
```

(1)

What about uncompressing an arbitrary (i.e., maliciously crafted) file ? (e.g., CVE-2010-0001 for gzip)

```
If C is not Compress(f) for any f then
    Uncompress(C) = "Error_Msg"
```

(2)

Actually (2) is much more difficult to validate than (1)

Some Definitions

Bug: an error (or defect/flaw/failure) introduced in a SW

- At the *specification/design/algorithmic level*
- At the *programming/coding level*
- Or even by the compiler (or other program transformation tools).

Vulnerability: a bug that opens a security breach

- *Non-exploitable vulnerability*: there is no (known!) way for an attacker to use this bug to corrupt the system

- *Exploitable* vulnerability: this bug can be used to elaborate an Attack (i.e., Write an exploit)

Exploit: a concrete program input to take advantage of a vulnerability (from an attacker point of view)

Malware: a piece of code “injected” inside a computer to corrupt it (usually exploiting existing vulnerabilities)

Countermeasures

Several existing mechanisms to **enforce** SW security

- At the programming level:
  - Disclosed vulnerabilities -> language weaknesses databases
    -> Secure coding patterns and libraries
  - Aggressive compiler options + code instrumentation
    -> early detection of unsecure code
- At the OS level:
  - Sandboxing
  - Address space randomization
  - Non executable memory zones
  - Etc.
- At the hardware level:
  - Trusted platform modules (TPM)
  - Secure crypto-processor (intel SGX, …)
  - CPU tracking

What is Secure?
(in the sense of Engineering Secure Software)

SAPIENZA
UNIVERSITÀ DI ROMA

Software Security

What is your favorite software development technology? (Language, tool, library, etc.)

Have you ever written software where security mattered?

- Did you do anything about it then?

How do you know that you have delivered secure software?

- Try to think of examples
- what are your indicators?
- How will you convince others that your software is secure?

Takeaways

- Security is not black-and-white
- Security is "until proven insecure"
- Security "theater"
  - Feeling safer vs. Being safer
  - People act on their perception of reality, not necessarily on reality
- Protection can be costly
  - E.G. Personal liberty and privacy
- Eliminating a threat vs. Protection
- Vulnerability vs. Exploit vs. Threat

An Engineer Concern

How do you know that you have built a system that cannot be broken into?

- What evidence do you look for?
- How do you know you are done?
- How do you prioritize security against everything else drawing upon your time?

SE is a zero-sum game

“If I need to focus more energy on security, what should we take away?”

Vulnerability

Informally, a *bug* with security consequences

A design flaw or poor coding that may allow an attacker to exploit software for a malicious purpose

- Non-software equivalent to “lack of shoe-examining at the airport”
- E.G. Allowing easily-guessed passwords (poor coding)
- E.G. Complete lack of passwords when needed (design flaw)
- Mcgraw: 50% are coding mistakes, 50% are design flaws

Alternative definition: “an instance of a fault that violates an [implicit or explicit] security policy”

Exploit and Threat

Exploit: a piece of software, a chunk of data, or a sequence of commands that takes advantage of a vulnerability in an effort to cause unintended or unanticipated behavior

I.E. Maliciously using a vulnerability

- Can manual or automated
- Viruses are merely automated exploits
- Many different ways to exploit just one vulnerability

Threat – two usages of the word

(A) an actor or agent that is a source of danger, capable of violating confidentiality, availability, or integrity of information assets and security policy

E.G. Black-hat hackers

(B) A class of exploits

E.G. Spoofing

Exploit/Threat/Vulnerability Protection

Protect against exploits?

Anti-virus, intrusion detection, firewalls, etc.

Protect against threats?

Use forensics *to find and eliminate policy, incentives, deterrents, etc.*

Protect against vulnerabilities?

Engineer secure software!

Software Security is...

NOT a myth but a reality

Insecure software causes *immeasurable* harm

Sony, NSA, android, browsers... just read the news

Software Security is...

NOT an arcane black art

Much of it seems arcane

- Finding a severe vulnerability w/o source code
- Crafting the exploit
- Endless clever ways to break software

But, you have much more knowledge than the attackers do

Do not just leave it to the others, take responsibility for knowing security

Software Security is...

NOT a set of features

Secure software > security software

Although **tools** and **experts** are helpful

- You cannot just deploy a magical tool and expect all vulnerabilities to disappear
- You cannot outsource all of your security knowledge

Even if you are using a security library, know *how* to use it properly

Software Security is…

NOT a problem for just mathematicians

Cryptography

- Is important and needed
- Cannot solve all of your security problems
- Pick-proof lock vs. Open window

Proofs, access control rules, and verification are helpful, but inherently incomplete

Software Security is…

NOT a problem for just networking and operating systems

Software had security problems long before we had the internet

If you left a window open in your house, would you try to fix the roads?

Software Security is...

A reality that everyone must face

- Not just developers, all stakeholders

A learnable mindset for software engineers

The ability to prevent *unintended functionality*

- At *all* layers of the stack in *all* parts of your system