Application Security (apsi)

Lecture at FHNW

Lecture 1, 2021

Arno Wagner, Michael Schläpfer, Rolf Wagner

<arno@wagner.name>, <{michael.schlaepfer,rolf.wagner}@fort-it.ch>

Agenda

- Who are your lecturers?
- Administrative issues and lecture overview
- Why is "Application Security" important?
- Introductory examples

Administrative Issues

- Both APSI courses are combined online. For on-site, lectures AWagner / Schläpfer+RWagner are synchronized and cover the same material.
- "Drehbuch": Expected this week on FHNW server
- Materials: On Github: https://github.com/Fort-IT/FHNW-apsi
- Exam during the semester:
  - 90 Minute (Date: 22.11.2021)

    Exam results and discussion: At start of the following lecture (6.12.2021)
  - Final Exam (MSP), covers whole lecture, including Lectures 14 and 15.
- Mandatory attendance: Only during exams
- Lecture:
  - Monday. 12:15-15:00, Refer to Drehbuch
  - 3 x 45min lecture + 2 x 15min break

Who is Your Lecturer (Arno Wagner, Group 1)

Dr. Arno Wagner

- Dr. sc. techn. in the area of Information Security from ETH Zurich
- Dipl. Inform. University of Karlsruhe
- Senior Security Consultant and CTO (more of a "chief engineer")
  Consecom AG, Zurich, until 2020
- Since 2020: IT and IT Security Auditor with
  Acons Governance & Audit, Zurich

- Interested in Security for along time now
- More of a Linux/UNIX person than a Windows person
- Diverse teaching experience including ETHZ, HSR, ZHAW

whoami (Michael Schläpfer, Group 2)

Dr. Michael Schläpfer

- Dr. sc. ETH Zurich in the area of Information Security
- MSc ETH Computer Science
- Dipl.-Ing. FH Informatik @ZHAW
- Executive MBA HSG
- Founder and Senior Security Consultant FortIT GmbH, Zurich

- Strong interest in various topics related to Information Security and ML
- Currently involved in projects around IAM, Fraud Management, and API Security

whoami (Rolf Wagner)

Rolf Wagner

- MSc ETH Computer Science in the area of Information Security
- Dipl.-Ing. FH Informatik @HSLU
- Co-Founder FortIT AG & Bug Bounty Hub AG, Zurich

- Strong interest in various topics related to Information and Cyber Security: *Application Security, Risk Management Frameworks, Zero Trust Security Model, Cloud Security*
- Currently involved in projects around *Security Awareness, Information Security Management System and Managed Security Services*

Why is Application Security Important?

- Almost all software is network-reachable in some form today
- A lot of software is exposed to the Internet
  Much of it is not standard-software, but custom implementations
  → This will not change anytime soon
- „Secure“ programming languages do not exist (and may never exist)
- There is always something to steal/use/sabotage on computers

Will it stay important?

- Yes, very much so
  The problem seems not to be accessible to technological solutions
- There are far too few experts and that is unlikely to change

What is expected of you?

- "*Unselfconscious Design*" (simplified): Use the approaches and techniques everybody else uses (including the "hype du jour"), whithout in-dept reflection. Works well for standardized, well established tasks
- "*Selfcounsious Design*" (simplified): Understand the problem to be solved, then select the best suited approaches and techniques from the known ones and, in some cases, design new ones to fit a specific requirement. This approach is required for good results to new or not well understood problems. such as writing secure software.

Unselfonscious Design (knowing the established and customary approaches and techniques and being able to apply them) is fine for technicians.

You are going to be engineers. In addition to being competent technicians, engineers need to be able to apply selfconsious design competently and recognize cases where it is needed.

Lecture Overview and Organization

=> Drehbuch

=> Excel with lecture plan

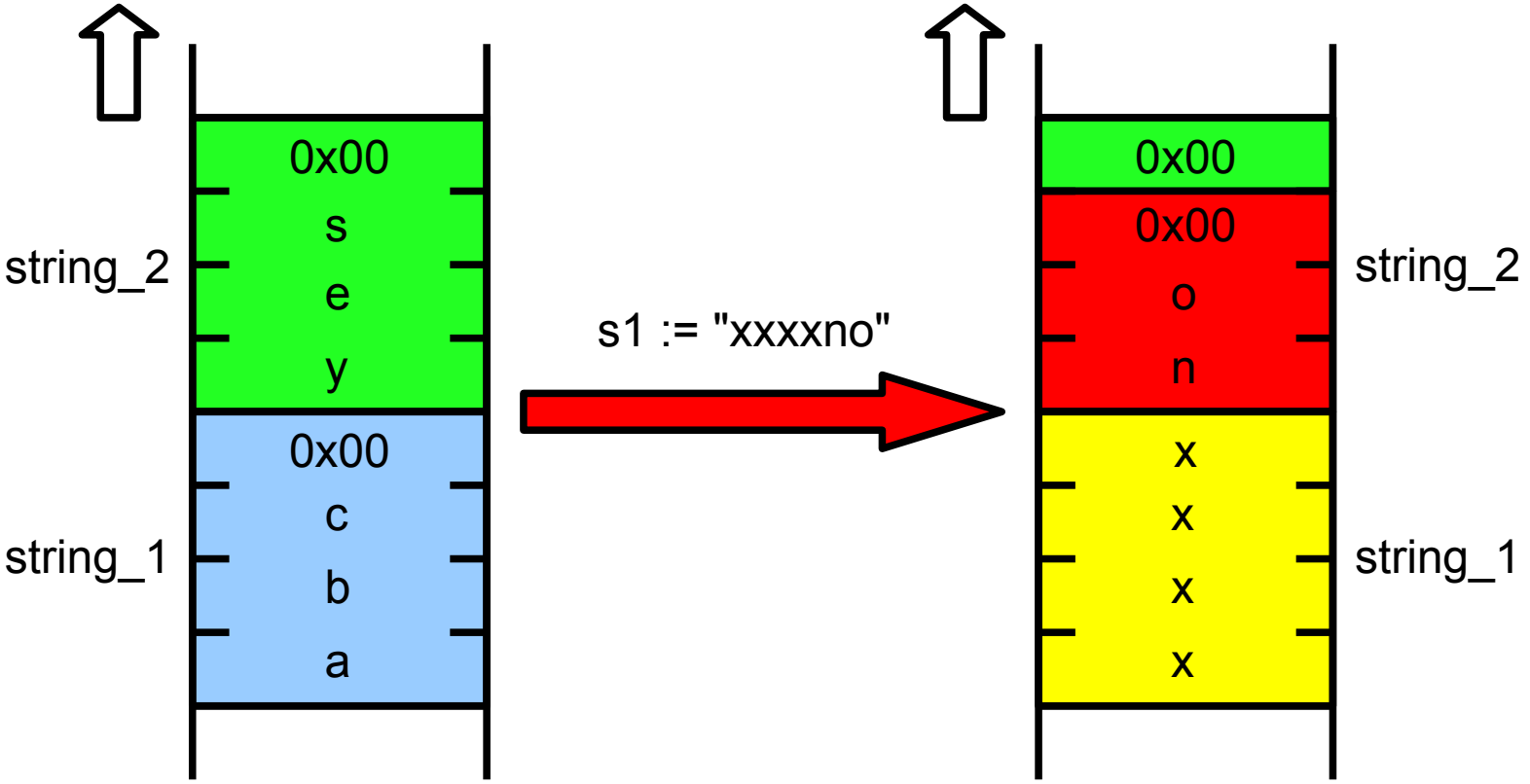
The most infamous Implemenation error: Buffer Overflow

- A too large object overwrites memory next to it
  - This is an "emergent property" of linear adressible RAM
    This means it is *surprising, unexpected funtionality*
- Old problem: First described 1972 (US military), First attack: Morris worm (1988)
- Easy to find for attackers and often easy to attack
- Defense is expensive (coding time and/or run-time)

Hence:

- All attempts to really fix this have failed. It will not go away anytime soon.
  → Even Web-Assembler (WASM) can be attacked by some variants
- Understanding its basics is important, even if it very low-level.
- It can be encountered in any language via libraries, run-time systems, system calls and even hardware faults.
- You may have to fix it yourself, temporarily or permanently

How does a Buffer Overflow Work?



Buffer Overflow Demo

```
#include <string.h>
#include <stdio.h>

int main (int argc, char *argv[]) {
  char s2[4] = "yes";    // set s2 to "yes"
  char s1[4] = "abc";    // set s1 to "abc"
  strcpy(s1, argv[1]);   // copy argv[1] into s1
  puts(s2);              // print s2
}
```

Note: May need longer input due to alignment

Note: For the demo to work, stack protection may have to be turned off:

```
gcc -fno-stack-protector ….
```

(Not Really) a Fix: Stack Protection

Idea: Put a random value on the stack and verify it

Usually only done to protect the return address of a function:

Stack: <local vars><random value><return address>

Problems:

- Were to place? Last example not fixed by this!
- When to check?
- Does cause additional effort.
- Does not fix buffer-overflow on the heap!

More on this: Next lecture

Really a fix for Buffer Overflow: Checked Buffers

Idea: Do size checks on every buffer insertion

How ensure these are done?

- Have a code style guide that requires them
- Do code inspection (code review by a different person)
- Use buffer libraries that do these checks
- Use a language with safe buffers
  Note: May cause other problems, for example memory exhaustion!

Example: Data Leakage by Behavior (1)

Assume: A system that lets an user log in over the web with a user-name and an account number.

- If the user is correct, but the account wrong, it prints "invalid account"
- If user and account are wrong, it prints "invalid input"
- If user is wrong, but account is valid, it prints "cannot access account"

What is the problem here?

Example: Data Leakage by Behavior (2)

Weaker forms:

- Different timing (example: there is one more DB query for the account when user is unknown and one less if the user is unknown)
- Different background color on different errors, because they are not from same file and one was changed
- Different next page (URL) for different errors
- Etc.

How to fix this?

- Go into exactly the same error flow for all errors
- Perform exactly the same queries in all situations
  Problem: Query-time may depend on validity of argument!
- Add artificial, fixed delays (difficult to get right)

Example: Data Leakage by Error Output

- Traces of uncaught exceptions (this happens time and again...)
- Program and library versions
- Code-internal information
- OS version, patch-level, etc.

Problems:

- Identification of insecure libraries
- Identification of insecure OS versions, missing patches, etc.
- Aids in assessment of how "worthwhile" a target is

The Economics of Attacks on Software

This is not an attempt to cover this fully!
Also refer to the daily news for new cases…

- Historically, J. Random Hacker was living in his parent's basement
  → Attacking systems was difficult and did not pay well
  → The Internet still delievered a lot of easy targets
- New development: Ransomware

  What is ransomware?
  - Attackers break in and encrypt all online-reachable data stores
    Systems, data-partitions, file-servers, online "backup", etc.
  - Attackers then demand money to supply a decryption key
  - Attackers may also steal data and threaten to leak secrets
  - Attackers employ pressure techniques like increasing fees for later responses

Ransomware has Gotten More Professional

Classical ransomware:

- Unreliable, buggy
- Often just random targetting, also against private individuals
- Often decryption not even possible

Modern ransomware:

- Works reliably
- Often used in a targetted fashion (with phising or manual intervention after initial compromise is reached)
- High-value targets preferred, because of high potential pay-off
- Competent use of encryption

What Does This mean?

Ransomware-based enterprises now have a steady source of income

This means they can start to do strategic business development!

- They can hire people and offer careers
- They can compete for the best talent
- They can offer "reputation" to their "customers"!
  If "crapbot" attacked you how are you going to know decryption works?
  But if our "bestbot" did it, clean decryption is assured! Or your money back!
  → They can increase payment probability by offering a better "product"!
- They can use good software engineering practices, with extensive tests, development and production versions, product updates and security fixes, etc.

And all that means they will stay around, probably for a long, long time.
Because now, this thing is profitable !

Defense Against Ransomware

- Do not have/use insecure software (APSI tries to cover some basics…)
- Be prepared!
  - Have offline or carefully secured backups (WORM: Write Once, Read Many)
  - Be prepared to restore most or all of your infrastructure from backups and clean images. This critically involves realistic DR (Disaster Recovery) tests!
  - Have BCM (Business Continuity Management) measures in place that give you the time to restore things.
  - For large enterprises: Segment the infrastructure, so not everything is affected.
    In particular: No shared AD or IAM for production and "office"!

  If done right: You may recover a lot faster than "paying" would have given you.
- Do not feed the trolls! Not being prepared is bad for everybody, not just you!

Why is Ransomware Even Possible?

- IT and Software security is still seen as a cost, not a benefit.
  - How many mandatory lectures on IT and Software security do you have in your studies? Right ...
- It is a "Black Swan" (low probability event) → This is currently changing
- These pesky security people get in the way all the time and demand things!

  Hence: The CISO of Equifax was a music major (seriously!).
- IT is seen as "too expensive" anyways, completely ignoring that most businesses are dead if their IT stops working for too long.
- IT is a new thing in human history. Its risks and benefits are not well understood.
  Historical perspective: Technologies need around 50-150 years from lab-demo to reliable everyday technology. That will still take some time.

Also see next slide for some pretty critical other factors...

Human Factors

Core issue: Incompetent people often think they are great at things

- "Unskilled and Unaware of It": Justin Kruger and David Dunning, Journal of Personality and Social Psychology, 1999, Vol. 77, No. 6.] 121-1134
  Ref.: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.2655&rep=rep1&type=pdf

Organizations typically promote people until they are incompetent

- Reference: Dr. Laurence J. Peter, Raymond Hull (1970). The Peter Principle. Pan Books. ISBN 0-330-02519-8. Reference: http://en.wikipedia.org/wiki/Peter_principle

The technology skill-level of people can be extremely low

- "There is considerable overlap between the intelligence of the smartest bears and the dumbest tourists." -- Yosemite park ranger on bear-proof trashcans
  Reference: http://www.schneier.com/blog/archives/2006/08/security_is_a_t.html

Organizations often discourage criticism of those in power

- "Shoot the messenger": Very basic beginner's mistake in management, very common.
  Reference: https://en.wikipedia.org/wiki/Shooting_the_messenger

Exercises

- Get a Linux + gcc + gdb environment to work with
  → Linux VM image offered with this lecure recommended
- If no Linux/Unix experience: Work through a small Linux tutorial
  - Using the shell: man, cd, ls, cp, mv, rm, mkdir, rmdir
  - Installing packages (Debian: apt-get, dselect, aptitude)
  - Editing and storing texts (editor: your choice, e.g. vim, emacs, joe, nano)
  - Calling gcc, gdb, (DDD for a graphical front-end for gdb, if desired)
- If no C-experience: Work though a simple C tutorial, including pointers
  - Experiment until you understand pointers, stack layout, local variables
- Re-create and analyze the buffer-overflow example using gdb

Note: In principle, you can do this on another Unix-like platform or even on Windows (via Cygwin). But some things may not work or be different.