Secure Programming Techniques

Margus Freudenthal, Mart Oruaas

2022-02-11
Formalities

- Course: MTAT.07.015 (3 EAP)
- Lecturers: Margus Freudenthal, Mart Oruaas
- Format: online Zoom lectures, maybe occasionally a prerecorded video
- Homework: four homework assignments, worth 10 points each.
  - Mostly practical hacking exercises in a simulated environment
- Seminar: research and presentation about a security issue, worth 10 points
- Exam: written exam, worth 50 points

2022-02-11
Homework

- Four homework assignments, each one worth 10 exam points
  - Aivo Toots and Andres Jõgi will talk about in more detail
- Seminar: research a more complicated security issue/security breach and give 20-minute presentation
  - Worth 10 exam points
  - Please confirm the topic of the presentation with lecturers before committing to it
  - Seminar presentations are in the second half of the semester
CV: Margus Freudenthal, PhD

- 1999-present: Cybernetica AS, development manager
- Architect of various security and mission critical systems:
  - Data exchange layer X-Road and its commercial version UXP
  - Digital signature solution TrueSign
  - Linked timestamping solution (blockchain in 2002 :)
  - Marine radio communication system
  - Customs declaration processing system
- Managing the software development process of the company
- Co-author of the first Estonian digital signature standard
CV: Mart Oruaas

- Various companies over the years, focus on security and performance of mission critical software systems:
  - Took part in introducing PIN-and-chip bank cards to Estonian market, designed a set of necessary key management mechanisms
  - Designed a key management solution for Skype’s P2P network during its early days
  - Team lead for Skype’s network stack developers
  - Principal technical architect of the Smart-ID service during its development until obtaining the QSCD status
  - Currently working on DynaTrace’s next generation analytics engine
What this Course is About?

- Modern programming languages
  - Java
  - Python
  - JavaScript
  - Go/Rust?
- Some notes on C/C++, PHP, Shell (there is quite a bit of bad code written in these languages, so we will give some tips)
What this Course is About?

- Modern application types
  - Web
  - Web services, APIs
  - Mobile applications
  - Applications rather than systems programming (separate, quite specific field)

- We assume high level of abstraction – building a system from existing components as opposed to writing low-level code from scratch
What this Course is About?

- We try to be constructive, not destructive
- We give guidance on how to write secure software, not enumerate badness (all the ways to create security issues)
  - Although, we will use different vulnerabilities as examples
- We emphasize architecture-level issues
- We recommend using existing good protocols and technologies instead of describing how to invent stuff from scratch
Course Structure

- We will use OWASP Top 10 as baseline
- Most of our topics are mapped to OWASP Top 10 issues
- Additionally, we have more general lectures on security models and security architectures as well as more specific topic, such as mobile or cloud security
OWASP Top 10

2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey
First of All, Some Terminology

- Oht (threat)
- Nörkus (vulnerability)
  - Rünne (attack)
  - Murre (breach)
  - Sissetung (penetration)

- Risk (risk)
  - Paljang (exposure)

- Turvarike (compromise)
  - Kahju (loss)
The Kinds of Vulnerabilities

- Vulnerabilities range from simple SQL injections and buffer overruns to very complicated ones, like SPECTRE and PlunderVolt.
- We stick to the deterministic vulnerabilities that assume “normal” behavior of the hardware.
- All kinds of interesting vulnerabilities raising from the complicated nature of modern hardware are important, especially in cloud environments.
- Each of those would require its own course, though.
Why bother?

- Software security has long ceased to be a “computer problem”, it also has a very significant human aspect.
- Thus is it not only a security, but also a safety problem, even outside of the fields that are normally considered safety-critical.
- Broken or badly set up software can kill, in most surprising ways! [Montalbano, 2021]
About 20 years ago, computer security issues were limited to the topmost “horns”. It is everywhere, now. Drawing from [Mäkinen, 2005].
Economically Motivated Attackers

- Only a small minority of incidents is caused by hooliganism or petty revenge.
- Some are caused by state-level attackers: espionage, sanction evasions etc.
- Majority of information security incidents are caused by economically motivated criminals, who have formed a number of well-organized gangs.
- Attacking is pure business.
Car of the Evil Corp’s Leader Maksim Jakubets
An Example

- Vastaamo was a Finnish company running 25 psychotherapy centers
- They had data breaches in 2018 and 2019, did not report them
- Attackers demanded €450k ransom, company refused to pay and attackers started demanding ransom from the patients
- Currently Vastaamo has filed for bankruptcy
An Example

- Colonial Pipeline ransomware attack in May 2021, ransom paid was worth 4.4 million USD at the moment of payout
- Total economic damage is hard to assess, but the whole US east coast had fuel shortage for many days.
- Attack vector: compromised password in combination with a VPN gateway without 2FA
- Attack vector is usually something mundane: e-mail client, VPN gateway, web application with SQL vulnerabilities
Dangers of Digital Transformation

- We have sold IT as the solution to everything – transforming the enterprises to digital
- With digital enterprise come new digital risks
  - Especially in the fields that were not connected before, for example manufacturing
- After being hit with attacks, enterprises are hesitant to continue their digital journey
Economic Issues of Security

- The brokenness of software can be explained by economic motivations of various actors in the system.
- In most cases, buyers of security products have no information if the product is good enough (at all): “market of lemons” – that dynamic removes really good products from the market.
- In most cases, those suffering from the consequences are not the same who need to make the investment.
- Reading material: a Ross Anderson’s seminal article [Anderson, 2001], Bruce Schneier has written a lot about economics of security in his blog.
What IS a Vulnerability

- Bugs
- Idea
- Implementation
- OK
- Vulnerabilities
It all comes down to ways of triggering the unintended functionality.
[Dullien, 2017]
Security vs. Reliability

- Simple example: Unix ls command
- What if it contains buffer overflow in processing input parameters, allowing execution of arbitrary code?
- For ordinary command-line use, this is a reliability issue
  - You cannot get any more privileges than you already have
- It will become a security issue if the program is used on a boundary of two security zones
- Did the original developer of ls anticipate this usage and designed/tested for it?
Avoiding Vulnerabilities

- There is no silver bullet
- But there is a range of techniques that help to avoid both vulnerabilities and bugs
- It is always easier to avoid whole classes of weaknesses – for example, by a choice of programming language in use
- “Programmers should not be glorified accountants” – any measure of avoiding defects that relies solely on the carefulness of any single programmer is doomed to fail
- Everything that could be automated should be automated – obtain and use proper tooling
General Principles

- Security must be designed into the system from the beginning, not patched in later
- Security can be audited only relative to knowledge in specific point of time
- Enumerating badness does not work
- Many simple bugs can appear as vulnerabilities when the data to trigger it comes from another security domain (untrusted source)
- Complexity is your enemy
Questions?
References I


References II


References III

Weird machines, exploitability, and provable unexploitability.

The Unfalsifiability of Security Claims.
References IV


References V
