Welcome to

Introduction to Software Security

Introduction to IT-Security; Dat Introduction to IT Security 2024 Week 7

Henrik Kramselund he/him han/ham xhek@kea.dk @kramse

Slides are available as PDF, kramse@Codeberg https://codeberg.org/kramse/intro-to-it-security-week-7.tex in the repo security-courses

Contact information

- Henrik Kramselund, he/him internet samurai mostly networks and infosec
- Network and security consultant Zencurity, teach at KEA and activist
- Master from the Computer Science Department at the University of Copenhagen (DIKU)
- Email: xhek@kea.dk Mobile: +45 2026 6000

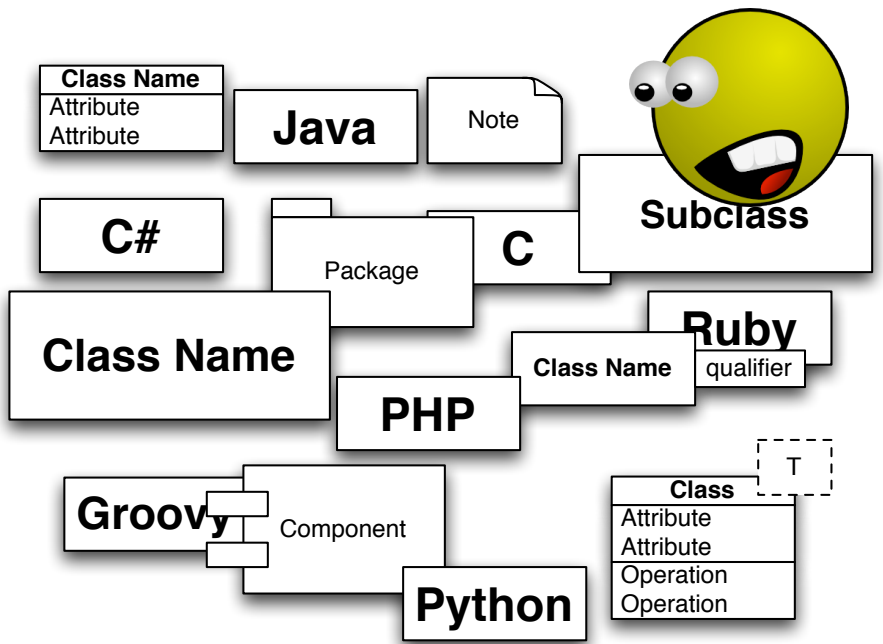
You are welcome to drop me an email

Goals for today

- Introduction to Software Security
- Create a good starting point for learning about *softsec*
- Learn to find resources, files and programs/libraries
- Prepare tools for the exercises

Photo by NESA by Makers on Unsplash

Goals: Software Security



Software security - because software is everywhere

Plan for software security in this course

Picture from the OpenBSD project, software blobs

- Get started looking at this big subject
- 3 days with mix of teaching and doing exercises
- Last day dedicated to catching up and playing with OWASP JuiceShop

Plan for today

- Software Security intro – why do we have problems
- Software is complex
- All software have security vulnerabilities

Exercises

- Git getting started
- UFW firewall – you cannot hack what you cannot talk to!
- OWASP JuiceShop – tool installation, docker and an insecure software application
- Django tutorial – not doing it, but reading

Exercises

Since we are going to be doing exercises, each team will need a laptop

The following are two recommended systems:

- One able to run Docker software servers and web applications

Docker is a toolbox we will use and participants will use a project called OWASP JuiceShop

Course Materials

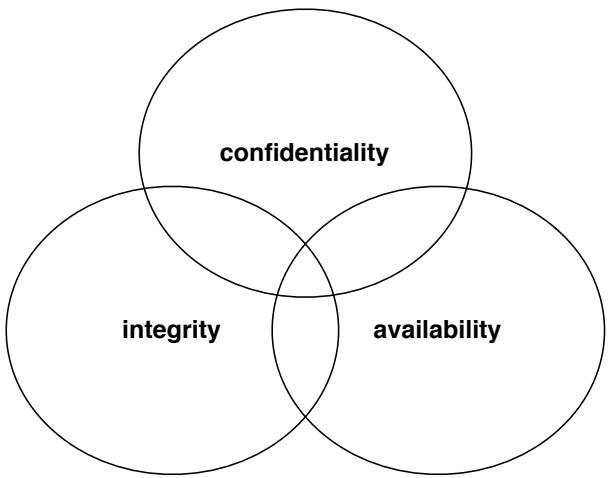
This material is in multiple parts:

- Slide shows - presentation - this file
- Exercises booklet
- Pwning OWASP Juice Shop Official companion guide to the OWASP Juice Shop Can be found online for free, but recommend buying the PDF from https://leanpub.com/juice-shop - suggested price USD 5.99

Additional resources from the internet

Note: the presentation slides are not a substitute for reading and doing exercises, many details are not shown

Confidentiality, Integrity and Availability



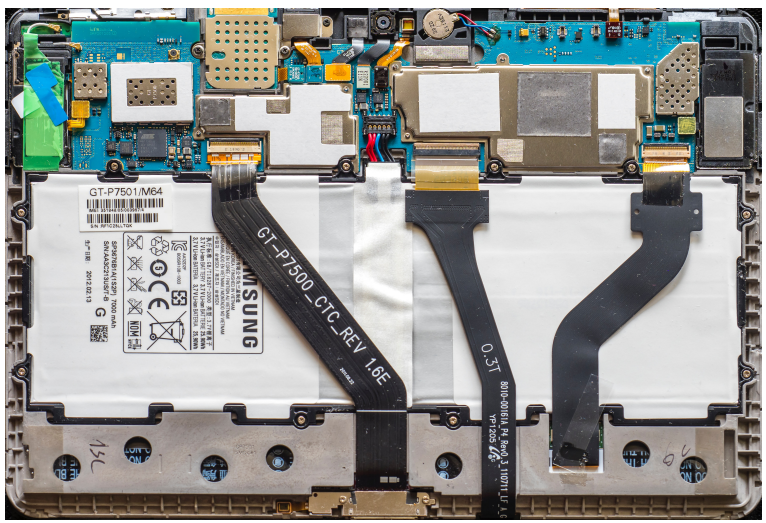
We want to protect something

Confidentiality - data kept a secret

Integrity - data is not subjected to unauthorized changes

Availability - data and systems are available when needed

What is Infrastructure – Software



- Enterprises today have a lot of computing systems supporting the business needs
- These are very diverse and often discrete systems

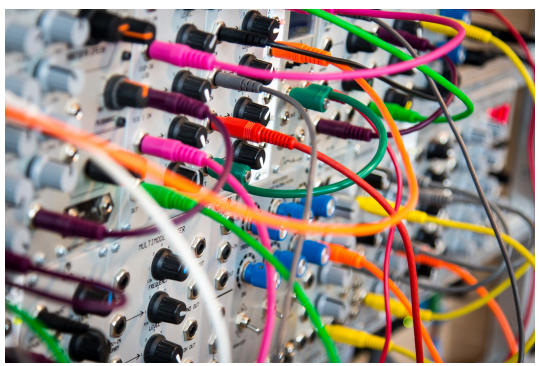
Photo by Alexander Schimmeck on Unsplash

Business Challenges

- Accumulation of software
- Legacy systems
- Partners
- Various types of data
- Employee churn, replacement

Photo by Adam Bignell on Unsplash

Software Challenges



- Complexity
- Various languages
- Various programming paradigms, client server, monolith, Model View Controller
- Conflicting data types and available structures
- Steam train vs electric train

Photo by John Barkiple on Unsplash

Developers Challenges

- Work in teams across organisation - and partners, vendors, sub-contractors
- Work with legacy systems, old technology
- Learn new Technologies

Photo by Kelly Sikkema on Unsplash

Integration Challenges

- Enable communication between components
- Need mediator, interpreter, translator
- Recognize standard patterns

Photo by Thomas Drouault on Unsplash

Leap Years

Who can remember the rules for leap years?

> The leap year problem (also known as the leap year bug or the leap day bug) is a problem for both digital (computer-related) and non-digital documentation and data storage situations which results from errors in the calculation of which years are leap years, or from manipulating dates without regard to the difference between leap years and common years.

Source: https://en.wikipedia.org/wiki/Leap_year and https://en.wikipedia.org/wiki/Leap_year_problem

- Something with February
- Something with every fourth year

Leap Years, the rules

> For example, in the Gregorian calendar, each leap year has 366 days instead of 365, by extending February to 29 days rather than the common 28. These extra days occur in **each year that is an integer multiple of 4 (except for years evenly divisible by 100, but not by 400).** The leap year of 366 days has 52 weeks and two days, hence the year following a leap year will start later by two days of the week.

Source: https://en.wikipedia.org/wiki/Leap_year

- So 1900 was not a leap year
- 2000 was a leap year!

Falsehoods programmers believe about time

kea

> I have repeatedly been confounded to discover just how many mistakes in both test and application code stem from misunderstandings or misconceptions about time. By this I mean both the interesting way in which computers handle time, and the fundamental gotchas inherent in how we humans have constructed our calendar – daylight savings being just the tip of the iceberg.

Source: https://infiniteundo.com/post/25326999628/falsehoods-programmers-believe-about-time

All of these assumptions are wrong

- 1 There are always 24 hours in a day.
- 2. Months have either 30 or 31 days.
- 3. Years have 365 days.
- ...

Lesson, calculations with time are complex, don't implement time software – use libraries!

Assumptions

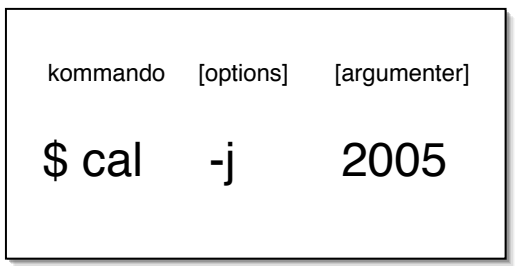
> Any security policy, mechanism, or procedure is based on assumptions that, if incorrect, destroy the superstructure on which it is built.
> Matt Bishop, Computer Security 2019

Example, vendor patches

Important points:

- Is patch correct? Example Spectre and heartbleed
- Vendor test environments equal to intended environments
- Installed correctly - including operator skills

Unix Manual system



It is a book about a Spanish guy called Manual. You should read it. – Dilbert

Manual system in Unix is always there!

Key word search `man -k` see also `apropos`

Different sections, can be chosen

See `man crontab` the command vs the file format in section 5 `man 5 crontab`

A manual page

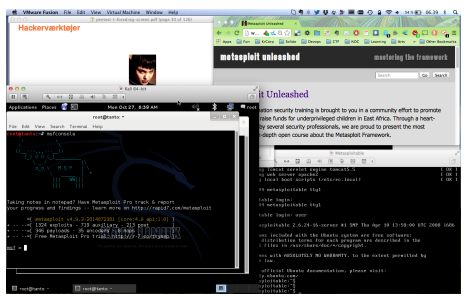
```
NAME
     cal - displays a calendar
SYNOPSIS
     cal [-jy] [[month]  year]
DESCRIPTION
   cal displays a simple calendar.  If arguments are not specified, the cur-
   rent month is displayed.  The options are as follows:
   -j      Display julian dates (days one-based, numbered from January 1).
   -y      Display a calendar for the current year.

The Gregorian Reformation is assumed to have occurred in 1752 on the 3rd
of September.  By this time, most countries had recognized the reforma-
tion (although a few did not recognize it until the early 1900's.)  Ten
days following that date were eliminated by the reformation, so the cal-
endar for that month is a bit unusual.
```

The year 1752

```
  user@Projects:$ cal 1752
...
         April                    May                      June
  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
              1  2  3  4                  1  2      1  2  3  4  5  6
   5  6  7  8  9 10 11   3  4  5  6  7  8  9   7  8  9 10 11 12 13
  12 13 14 15 16 17 18  10 11 12 13 14 15 16  14 15 16 17 18 19 20
  19 20 21 22 23 24 25  17 18 19 20 21 22 23  21 22 23 24 25 26 27
  26 27 28 29 30        24 25 26 27 28 29 30  28 29 30
                        31
          July                   August                 September
  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
              1  2  3  4                     1        1  2 14 15 16
   5  6  7  8  9 10 11   2  3  4  5  6  7  8  17 18 19 20 21 22 23
  12 13 14 15 16 17 18   9 10 11 12 13 14 15  24 25 26 27 28 29 30
  19 20 21 22 23 24 25  16 17 18 19 20 21 22
  26 27 28 29 30 31     23 24 25 26 27 28 29
                        30 31
...
```

Hackerlab Setup



- Hardware: modern laptop CPU with virtualisation
  Dont forget to enable hardware virtualisation in the BIOS
- Virtualisation software: Docker
- Setup instructions can be found in the exercise booklet

It is enough if teams each have one laptop with Docker

About the exercises

You will need in your group – 2-3 persons is recommended: Docker or similar container technology,Browser – Firefox comes to mind and Burp Suite – Java program

I will use a virtual machine with Debian 12 Bookworm for this! Most exercises can be executed from this VM. You may be able to run exercises from your normal operating system, but it may take longer than just adding a Debian VM and doing it.

Technologies used

The following tools and environments are examples that may be introduced in this course:

- Programming languages and frameworks Java, Python, regular expressions
- Development environments – choose your own IDE / Editor – I use Atom
- Networking and network protocols: TCP/IP, HTTP, DNS
- Formats XML, JSON, CSV
- Web technologies and services: REST, API, HTML5, CSS, JavaScript
- Tools like cURL, Git and Github
- Cloud and virtualisation **Docker**
  Kubernetes, Azure, AWS, microservices – cloud is a big part of security today

This list is not complete or a promise

OWASP Juice Shop Project

We will also use the OWASP Juice Shop Tool Project as a running example. This is an application which is modern AND designed to have security flaws.

Read more about this project at: https://www.owasp.org/index.php/OWASP_Juice_Shop_Project
https://github.com/bkimminich/juice-shop

It is recommended to buy the Pwning OWASP Juice Shop Official companion guide to the OWASP Juice Shop from https://leanpub.com/juice-shop - suggested price USD 5.99

Aftale om test af netværk

Straffelovens paragraf 263 Stk. 2. Med bøde eller fængsel indtil 6 måneder straffes den, som uberettiget skaffer sig adgang til en andens oplysninger eller programmer, der er bestemt til at bruges i et anlæg til elektronisk databehandling.

Hacking kan betyde:

- At man skal betale erstatning til personer eller virksomheder
- At man får konfiskeret sit udstyr af politiet
- At man, hvis man er over 15 år og bliver dømt for hacking, kan få en bøde - eller fængselsstraf i alvorlige tilfælde
- At man, hvis man er over 15 år og bliver dømt for hacking, får en plettet straffeattest. Det kan give problemer, hvis man skal finde et job eller hvis man skal rejse til visse lande, fx USA og Australien
- Frit efter: http://www.stophacking.dk lavet af Det Kriminalpræventive Råd
- Frygten for terror har forstærket ovenstående - så lad være!

Exercise CHAOS: Don't Panic – have fun learning

> "It is said that despite its many glaring (and occasionally fatal) inaccuracies, the Hitchhiker's Guide to the Galaxy itself has outsold the Encyclopedia Galactica because it is slightly cheaper, and because it has the words 'DON'T PANIC' in large, friendly letters on the cover."

Hitchhiker's Guide to the Galaxy, Douglas Adams

Your lab setup

- Go to GitHub, Find user Kramse, click through security-courses, find the software-security folder https://github.com/kramse/security-courses/tree/master/courses/system-and-software/software-security
- Look into the files named: `intro-to-it-security-week-7.pdf`, `intro-to-it-security-week-8.pdf` and `intro-to-it-security-exercises.pdf`
- Install Docker on a laptop in your team – you might already have it

Hint: If you have a Debian virtual machine you can install the docker software as a package using the Ansible tool, by checking out a repo `kramse-labs` and running Ansible:

```
sudo apt install ansible git
git clone https://github.com/kramse/kramse-labs.git
cd docker-install
ansible-playbook -v 1-dependencies
```

Why introduce Git and Github?

We introduce Git here also because Github, one of the most popular places to store Git repositories has added security tools which you can use.

An example of a security feature at Github is the use of dependencies, when a project is stored on Github they can scan for outdated dependencies which have security issues.

You can read more about the features available, and some common problems in software in their article: *How GitHub secures open source software* Feb 23, 2021 // 10 min read
https://resources.github.com/security/open-source/how-github-secures-open-source-software/

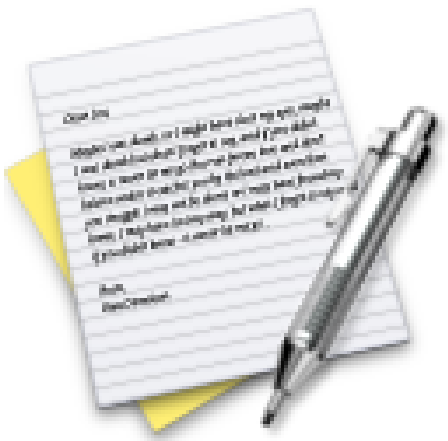
Exercise

Now lets do the exercise

⚠ Git tutorials - 15min

which is number 1 in the exercise PDF.

Exercise



Now lets do the exercise

Enable UFW firewall - 10 min

which is number 2 in the exercise PDF.

Design vs Implementation

Software vulnerabilities can be divided into two major categories:

- Design vulnerabilities
- Implementation vulnerabilities

Even with a well-thought-out security design a program can contain implementation flaws.

Common Secure Design Issues

- Design must specify the security model's structure
  Not having this written down is a common problem
- Common problem AAA Authentication, Authorization, Accounting (book uses audited)
- Weak or Missing Session Management
- Weak or Missing Authentication
- Weak or Missing Authorization

Input Validation

Missing or flawed input validation is the number one cause of many of the most severe vulnerabilities:

- Buffer overflows - writing into control structures of programs, taking over instructions and program flow
- SQL injection - executing commands and queries in database systems
- Cross-site scripting - web based attack type
- Recommend centralizing validation routines
- Perform validation in secure context, controller on server
- Secure component boundaries

Weak Structural Security

Our book describes more design flaws:

- Large Attack surface
- Running a Process at Too High a Privilege Level, dont run everything as root or administrator
- No Defense in Depth, use more controls, make a strong chain
- Not Failing Securely
- Mixing Code and Data
- Misplaced trust in External Systems
- Insecure Defaults
- Missing Audit Logs

Secure Programming for Linux and Unix Howto

More information about systems design and implementation can be found in the free resource:

Secure Programming for Linux and Unix HOWTO, David Wheeler

https://dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf

Chapter 5. Validate All Input details input validation in the context of Unix programs

Chapter 6. Restrict Operations to Buffer Bounds (Avoid Buffer Overflow)

Chapter 7. Design Your Program for Security

This is included to show that software security has been around for decades. Linux is also a very common platform today, being part of both Android, Cloud and Internet of Things (IoT)

Principle of Least Privilege

Definition 14-1 The *principle of least privilege* states that a subject should be given only those privileges that it needs in order to complete the task.

Also drop privileges when not needed anymore, relinquish rights immediately

Example, need to read a document - but not write.

Database systems can often provide very fine grained access to data

Principle of Fail-Safe defaults

Definition 14-3 The *principle of fail-safe defaults* states that, unless a subject is given explicit access to an object, it should be denied access to that object.

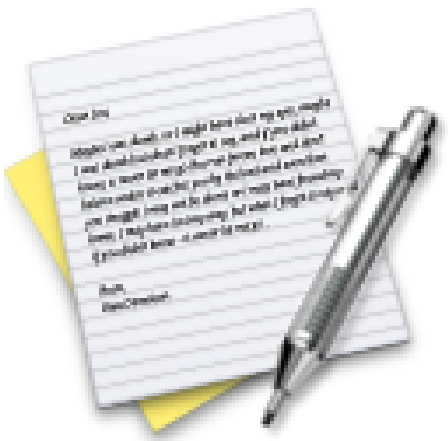
Default access *none*

In firewalls default-deny - that which is not allowed is prohibited

Newer devices today can come with no administrative users, while older devices often came with default admin/admin users

Real world example, OpenSSH config files that come with `PermitRootLogin no`

Exercise

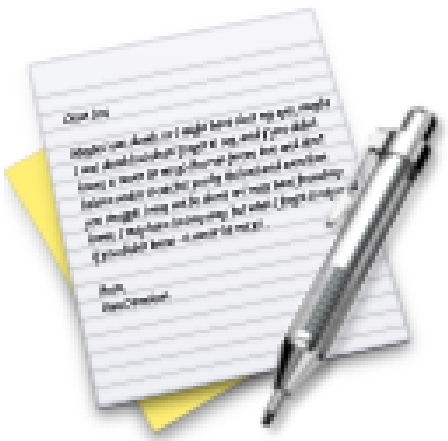


Now lets do the exercise

⚠ Run OWASP Juice Shop 45 min

which is number **3** in the exercise PDF.

Exercise

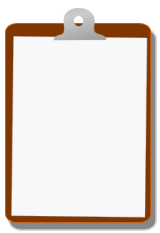


Now lets do the exercise

ℹ Setup JuiceShop environment, app and proxy - up to 60min

which is number **4** in the exercise PDF.

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools