Software Security

This page is for the first (2014) edition of the course. The current 2015 edition has its own page at Aalto University.

This is the course space for the University of Helsinki Dept. of Computer Science and Aalto University Dept. of Computer Science and Engineering course on Software Security. The course code is 582708 for University of Helsinki and T-110.6220 for Aalto University. The course is worth 4 credits. It is categorized as a MSc-level course but designed so that you **can include it in a BSc or a MSc degree**.

Course staff:

- Antti Vähä-Sipilä (lecturer, *avs* on IRC, avs@iki.fi, Twitter)
- Sini Ruohomaa (local organization, *Byakushin* on IRC)
- N. Asokan
- Visiting lecturer (session 3): Henri Lindberg from nSense.

The course has

- a Moodle space (Software Security) for returning homework assignments, and
- an Internet Relay Chat (IRC) channel **#tkt-swsec** @ IRCnetwhich can be used for discussing assignments, questions on lectures etc.
    - You can find instructions on how to get on IRC from DistSys'12 workshop exercise 4.
- UH course feedback form and an Aalto course feedback form

Schedule

The lectures are held in the Exactum building in Kumpula, Gustaf Hällströmin katu 2B (main entrance from Pietari Kalmin katu), first floor, **room D122**.

Lectures on **Wednesdays 16-18**, supporting workshop for homework assignments on **Mondays 16-18** (same room, D122).

There are **no lecture slides** on this course; the detailed lecture outlines in bullet-point lists appear below before the lecture.

Students' lecture notes, in contrast, should describe the lecture contents in educative prose that can be understood stand-alone by other students.

Date	Lecture outline, reading list ("What the lecturer tried to say")	Lecture notes (assigned to) ("What the course participants heard")	Weekly homework project
Wed 12.3. 16-18 D122	Intro & L1: How software breaks (native) L1 additional reading list	Blythe, Nyman, Freitag, Hafeez, Mäkipaja, Seemann dl first version 19.3., final 26.3. (1+1wk)	Project 1, dl Tue 25.3. (13 days*) Software that got fuzzed in this exercise Support for Ex. 1** on Mon 17.3.
Wed 19.3.	L2: How software breaks (web) L2 additional reading list	Ahmad, Lindberg, Koskinen, Hopearuoho, Lindert, Radhakrishnan, Tynkkynen dl first version 26.3., final 2.4.	Project 2, dl. Tue 1.4.
(Tue 25.3.)	(Optional: OWASP Helsinki meeting***) Continuous Security Testing in a DevOps World		
Wed 26.3.	L3: Security in a software project L3 additional reading list	Aulaskari, Kyllönen, Ukkonen, Muona, Niemistö, Jaakkola dl first version 2.4., final 9.4.	Project 3, dl. Tue 8.4. Tools that got reviewed Support for Ex. 3 on Mon 31.3.
Wed 2.4.	L4: Architectural risk analysis 1 L4 additional reading list	Oat, Skudnov, Kinnunen, Faghihi, Lado-Villar dl first version 9.4., final 16.4.	Project 4, dl. Tue 15.4. Support for Ex. 4 on Mon 7.4.
Wed 9.4.	L5: Privacy engineering, architectural risk analysis 2, design patterns L5 additional reading list	Paavolainen, Lehikoinen, Oinonen, Hilden, Liu, Laukkanen dl first version 16.4., final 23.4.	Project 5, dl. Tue 22.4. Support for Ex. 5 on Mon 14.4.
Wed 16.4.	L6: Software security and society L6 additional reading list	Sandvik, Lietzen, Karhunen, Oksanen, Svenn, Sorvisto, Chauhan, Tillmanns dl first version 23.4., final 30.4.	Project 6, dl. Tue 29.4. Support for Ex. 6 on **Wed 23.4. (see below)**
Wed 23.4.	Support for Ex. 6 (note: on teaching break)	-	(No workshop Mon 21.4. or 28.4.)

*) Note that **new projects come out weekly**, so we recommend completing the projects on the first week since their publication.

**) The workshops have minimal fixed program, so you can also come ask questions about the previous week's project or something completely different. We also have a possibility to go into more detail on some topics.

***) OWASP Helsinki organised a session on security in Continuous Integration - things like test automation for security tests. We had Stephen de Vries visiting to tell about his BDD-Security tools. OWASP is a great opportunity to network with security-minded web application developers.

Homework and evaluation

A 4-credit course is equivalent to circa 106 hours of work. Attending lectures (6 à 2h) takes 12 h and supporting workshop sessions (6 à 2h) takes 12 h maximum, general administrative overhead ca. 5 h.

The course contains weekly projects and lecture notes. Estimated time use 9 h per project (54 h total).

- **Weekly projects** consist of e.g. trying out tools (hands-on work), evaluating them (essay), architecture analysis and modification (application of theory), and discussing software security in a larger context (essay).
- Homework is returned in writing, via Moodle, and graded 0 (not done) - 5 (reaches learning objectives).
- A minimum number of weekly projects (out of 6) need to be done in order to pass, irrespective of the grading.
- Each weekly project submission has equal weight.
- We recommend that you complete each project by next Wednesday (a new project is published then)
    - Final deadlines for projects are **13 days after the lecture** they are connected to (Tuesday at 23:55).
    - Submissions later than this **are not graded**.

In addition, students produce **lecture notes** summarizing one lecture (estimated time use: ca. 1 credit, **25 hours**).

- Think of it as one chapter in a book about the course for another student, can use material from course book etc as well (remember proper citations, no cut-paste!).
    - See example of lecture notes from the Mobile Platform Security course.
- Several students assigned to cover each lecture, **individual** returns (= no copying).
- First deadline: **1 week after lecture** (Wednesday at 23:55**),** returns in writing through Moodle.
- First versions are visible to other course students, and can then be discussed and further polished **within 1 week** (same dl); you get a quick grade and feedback for your work. The final version is then returned 2 weeks after the lecture and also graded.
- Collaborative discussion is encouraged during the second week to improve your notes.
- Final lecture notes will be **published** in course pages.

Learning objectives, prerequisites and methods

The course learning objectives matrix describes what we expect you to learn on this course (and demonstrate you have learned it).

- The 'approaches the learning objectives' column is equivalent to sufficient knowledge to pass the course, while 'reaches the learning objectives' is equivalent to a 5/5 grade.
- Please note the **prerequisite knowledge** column in the matrix. For UH students at BSc level, you should have passed the Tietorakenteet (Data structures) course and have knowledge equivalent to Tietoliikenteen perusteet (Introduction to data communications) and Käyttöjärjestelmät/Tietokoneen toiminta (Operating systems, Computer organization).

The course will be graded based on weekly projects (70%) and lecture notes (one lecture, 30%, i.e. 10% first version and 20% second version). **There will be no exam.**

Lecture outline

- Session 1: How software breaks: native code
    - Software security terminology, discovering input processing issues with a fuzzer, and overview of basic old-school exploitation techniques.
    - Weekly exercise: Fuzz testing.
- Session 2: How software breaks: web applications
    - Privilege elevation on the web, output encoding and tainting as defensive principles, web session attacks, web security testing with attack proxies.
    - Weekly exercise: Intercepting proxies and simple web vulnerabilities.
- Session 3: Security in a software project
    - The internals of software projects, software security activities in project work and related standards, visiting lecturer from security industry (Henri Lindberg, nSense), a parade and classification of typical security tools.
    - Weekly exercise: Evaluating the usefulness of a security tool (essay).
- Session 4: Architectural risk analysis (threat modelling)
    - The needs of architectural risk analysis, visualization, security boundaries, taint analysis, security services provided for data flows and stores. STRIDE as an example. Risk-based decision making.
    - Weekly exercise: Threat modelling for a simple system.
- Session 5: Reliability and privacy analysis, and examples of security patterns
    - Extending architectural risk analysis beyond security to reliability and privacy. Applying three security patterns: Kerckhoffs' principle, bootstrapping trust, and adding security boundaries to abstract away problems.
    - Weekly exercise: Modify architectural design through applying a pattern.
- Session 6: Software security and society
    - Software security as a negative externality, disclosure models and vulnerability markets, software security regulation, and the role of software security in intelligence and warfare.
    - Weekly exercise: An essay concerning one of the topics where software security has societal impact.

This information is also available in a 2-pager course syllabus.

Supplementary material

The course has two supplementary books: "**Secure Coding**" is recommended (but not compulsory) as a course book, while "**Software Security: Building Security in**" is recommended for further reading. A very recently published "**Threat Modeling: Designing for Security**" looks very promising too (although the lecturer hasn't yet had time to read it cover to cover).

"Secure Coding" is available for online and PDF reading through the Kumpula university library (via the Dawsonera service), and additional paper copies have been ordered. "Secure Coding" has 4 print copies ordered to the library, and "Software Security: Building security in" 2 print copies. Both the paper and electronic versions can be found through the Helka search (to simplify your first login to Dawsonera, click on the book's name, then the red 'e' link on the right) as well as from searching in the library pages (the second link is the online version). For Dawsonera access, use your University of Helsinki (AD) or Aalto credentials to **log in** via Shibboleth. For the Threat Modeling book, see its website.

The development of this course is supported by a gift from Intel as a part of its Intel Labs Security Curriculum Initiative.

? Unknown Attachment

We would also like to thank nSense and Microsoft.

Navigate space

Recently Updated

Course space for Software Security
2018-01-15 • updated by Confluence Admin

Software Security
2016-09-26 • updated by N Asokan • view change

Session2WeeklyExercise.pdf
2014-05-30 • attached by Anonymous

Software Security
2014-05-30 • updated by Sini Ruohomaa • view change

tillmanns_lecture6.pdf
2014-05-30 • attached by Sini Ruohomaa

chauhan_lecture6.pdf
2014-05-30 • attached by Sini Ruohomaa

sorvisto_lecture6.pdf
2014-05-30 • attached by Sini Ruohomaa

svenn_lecture6.pdf
2014-05-30 • attached by Sini Ruohomaa

oksanen_lecture6.pdf
2014-05-30 • attached by Sini Ruohomaa

karhunen_lecture6.pdf
2014-05-30 • attached by Sini Ruohomaa

lietzen_lecture6.pdf
2014-05-30 • attached by Sini Ruohomaa

sandvik_lecture6.pdf
2014-05-30 • attached by Sini Ruohomaa

laukkanen_lecture5.pdf
2014-05-30 • attached by Sini Ruohomaa

liu_lecture5.pdf
2014-05-30 • attached by Sini Ruohomaa

hilden_lecture5.pdf
2014-05-30 • attached by Sini Ruohomaa