KU Leuven

Ma Ingenieurswetenschappen: Computerwetenschappen

Development of Secure Software

Course notes

Author:
Helena Brekalo

2015-2016

Contents

Chapter 1

Lesson 1

1.1 Slides: 1. Introduction - Security in Software Development(1)

Oral exam: 3/4 of the points
Project: 1/4 → needs to be defended at teh exam. Can't be redone in august, grade is kept.

Slide 3-6: Why is software security so important?

- Increasing amount of computers: estimated to be at 2 billion 5 years ago. At the moment there are more smartphones than computers and the next wave is that almost all (like IoT) things get internet connectivity these days. This brings its own set of challenges because a PC is different than a car (problems with Jeep!). Much worse things can happen in terms of human safety (pacemakers!). These things pose new threats.
- There's more and more software. The sizes of the code bases are incredible. A good estimate about the number of bugs is that well-maintained programs have about 100 bugs (I think? -¿ or 1 bug for every 100 lines). Bugs can become vulnerabilities as soon as they're discovered. The size of the systems is growing every year, OS's get bigger every year. There will never be bug-free systems.
- The impact of software grows. Everything runs on software. The important thing is that bugs, 20 years ago, could harm the ICT-industry, now they can harm everything. The yield of cyber weapons is increasing enormously over the past few years.
- All the software is more and more connected. We've had software in cars and fridges for years, but it didn't talk to the outside world. Now all these things start talking to each other, getting multiple Internet connections. They become more exposed to possible attacks. It's also seen in the way business is organized: the ordering software of one company may immediately talk to software of another company. It makes things faster and easier, but if there is a bug in there, it may have a bigger impact than when humans are involved.

Slide 7: Definition of computer security: in security we care about the case where we are up against intelligent issues, not issues that may arise randomly. We focus on maintaining some good property in the presence of intelligent adversaries.

Slide 8 a.f.: Examples of things that have gone wrong:

- Internet worms and viruses: it's still important today! A worm doesn't need human interaction (can work autonomously). First it was through the exchange of floppy disks, then by requiring human interaction (let people click on images,...). Worms don't need human interaction, that's even "worse" (example: in 30 minutes, the entire world got more infected), they happen even while you're sleeping (Slammer worm) -¿ completely different concern than human-interactive viruses.
- Website defacements
- Jailbreaking or rooting: any device can get jailbroken (the famous ones). Example: 2011 Sony hack, heartbleed (you could specify the length of the payload, but you could make this bigger and the server would provide more than it was actually supposed to send. You couldn't notice it afterwards, but important data could be stolen).

Slide 27: Assets: things that are a value in the system (services, hardware,...).

Slide 28: Adversary model: in order to reason about security as an engineer, you need to model your adversary. You need to make assumptions about what they can and cannot do. You have to be explicit about what adversaries you worry about.

Slide 29: You can also start from threats instead of the security goals of your system. Sometimes it makes sense to think about security in the two ways.
Threats can be classified (e.v. STRIDE by MSFT):

- Spoofing: pretending you're someone you're not
- Tampering:
- Repudiation: deny doing something that you have done
- Information disclosure: breaking confidentiality
- DOS
- Elevation of privileges: expand the access rights you have (you're a student but work as an admin).

They're too vague about themselves, you have to be more specific when defining the threats.

Slide 30: Security argument: you should be able to say that for the security goals that you had in mind/the threats you wanted to counter and under the given adversary model, you can explain that the goals cannot be broken. This is how you defend a security design. It's tricky to get them right because of the intelligent ways of the adversaries.

Slide 31: Vulnerability: take many forms and enter at many stages (during development, construction or operation).

Slide 32: Countermeasures: reduce the number of vulnerabilities in the system. They can be preventive, detective or reactive.

Slide 34: How well the program is documented (and administrated) impacts the security.

Slide 36 a.f.: Case study: e-mail: divides the internet in a number of domains that have domain names that are familiar. Each of these domains has a number of people that have addresses in these domains and machines that support the system. There are 2 special machines: mail storage server (pop) and mail transfer server (will route the messages $\rightarrow$ $\sim$smtp). Email (simple version used here): users use the client software to put together the mail and then they'll press 'send' and the mail client will contact the mail transfer server to deliver it. Then the server has the logic to find the right storage server to put the mail in (look at the domain of the recipient and then contact that). If it has a local recipient, then it will immediately hand it over to the ail storage server. Otherwise t will contact the other mail transfer server who will handle the delivery at the appropriate storage server. User 2 then receives the email via its mail client.

Potential security goals:

1. Confidentiality: an email message can only be seen/read by sender and recipient(s).
2. Integrity: modifications to an email message after sending by the sender should be detectable by the recipient.
3. Only users authorized by a domain can send messages from that domain.
4. Only a specific user u at a specific domain d can send messages as u@d.
5. Messages delivered to the system should reach specified recipients' inboxes.
6. Who communicates with whom is confidential.
7. It should be impossible to send viruses to spread through email.

Countermeasures:

1. Encryption, but where depends: you can encrypt end-to-end or inbetween.