1 Summary

How do you do software. There are number of steps in the software development process that can go wrong from security’s point of view. To add security as a part of the software development model there has been various variations of altering the model. Also use of external security specialists is quite usual.

2 Content of Lecture

The lecture had a guest lecturer, Henri Lindberg from nSense, who lectured about the reality of the situation in the security business. That is from the perspective of a security consultant. The theme was academic practices vs. reality. Antti Vähä-Sipilä finished the lecture with the topic being "How to do secure software". Note that secure software is different from security software. By secure software we mean a software, that is mostly not vulnerable. Saying that software is not vulnerable is kind of an impossibility, new vulnerabilities are found all the time, so we’ll be content with saying it’s mostly not vulnerable. By security software is meant a software that somehow directly tries to enhance information security.

3 Security in software process — current situation

The requirement for every software is that it should be secure, we should not leak our customers data, it’s also about being humane. If the user trust the software as much as to disclose sensitive data to software, then the software should be expected to try its best to protect it. Good example are search engines, how we blindly give information to corporations behind them. Software should be secure.

However, the current situation does not comply with this. Doing your software secure does not bring you instant wins. Its prevention work (in most cases), i.e. it only affects the profits when the leak happens. However, doing your software actually secure does add to software budget. So in essence, making secure software doesn’t bring instant wins but rather strains the budget. Typically, this means that features are first, security second. It doesn’t help at all that most software projects are almost always late and over budget.

How software is then made secure? One way is to have vulnerability assessment done on your software. Vulnerability assessment is the process of identifying, quantifying and ranking
the vulnerabilities in the system. It is quite usual to have an external security specialist do the vulnerability assessment of the system. Having an external (as in the person who did not code the system) do an analysis of the security of the software is hard in many ways. Firstly, it might not cause any action at all even if developers are made aware of the existing problems. In good cases vulnerabilities are fixed 0.5 to 3 years after deployment. Also vulnerabilities might not be taken seriously without being demonstrated first. This is quite sad since writing the exploit is the hardest part, and no one gains anything from the exploit, except a nice whoa effect.

Big problem for the security of the software is outsourcing developer work. This causes huge risks with trust boundaries. Outsourced code should be reviewed extremely well, which takes time, and that costs, which in turn kind of negates the cause of outsourcing. Outsourcing is lowering quality and costs for short-term profits. Usually it’s the product owner or management, not developers, behind the act of outsourcing.

Certification of secure software and developers is a nice concept, but unfortunately it doesn’t work all that well. Lecturer gave a nameless example of a certified software, which broke down with your everyday hacking methods. There were also mentions of certification being able to be bought. The Certification method of most common and well known developer certificates is a multiple-choice test. Apparently one can find the results for multiple choices with a bit of Googling.

One of the late movements in the software world is being agile. Agile is just about the opposite of waterfall model. Instead of having a predefined route (requirements, design, implementation, verification, maintenance) for the software, these all are done at the same time. Of course being agile doesn’t fit all projects, and there are many variations of agile movements. However, making the software agile might be very bad for the security of the software. In the worst case it means that new features are done quickly while no one is testing or doing the software secure.

To have the software done securely is not about having vulnerability assessments done on the software, while they are of course good when done right. The security should be part of the software development process. It’s about caring as well. The next section is about adding the security as a part of the software development process.

4 How to do secure software

Developing software isn’t simple, developing secure software is much harder. If we take a look at the process of doing software:

- Finding requirements ~ product management
- Design
- Coding
- QA ~ testing
- Project management
- Documentation
- Shipping ~ deployment
- User training
• Operations
• Maintenance
• Customer management
• Decommission ~ end of life
• Sales & marketing
• Risk management

Most of these 'sections' have some gotchas where software can fail being secure. With design comes architecture problems. Coding part has its own set of failures, e.g. buffer overflows or using unsafe functions, or coder not knowing what (s)he’s doing. Testing can fail to test various corner cases.

Typically requirements of the software don’t include security. Or if they do it might be something generic like: ”Apply to secure coding practices”. This is quite hard to ‘require’ from a software as it can never be finished. Usually requirement has a definite begin and ending. To add security as a part of the software requirements we could

• add abuse cases, or attacker stories, which could be tested then.
• require testing and crypto features for sensitive data. e.g. User passwords must not be saved anywhere, instead use a heavy weight key derivation functions, such as PBKDF2, due to bitcoin mining making hash functions too fast. [Hou]
• require use of a security specialist, or that each team has one.
• require to comply to a given guideline, which in turn might be converted to a checklist or automated test. e.g. Do not use function that is marked as unsafe, greppable.

These are somewhat of a unicorn in requirements. It’s what we should be aiming for but they’re almost never really seen in real world.

Design is the part of the software project that decides on the high-level view of the software, such as use of design patterns. With design comes various risks that should be noted from security point of view. However, latter lectures will cover this topic. I’ll just settle for noting that threat modeling, or architectural security risk analysis belongs here.

The architecture of the software should be well planned, as it is really hard or even impossible to fix it afterwards. For existing projects, that have failed with design, exists a few ways to fix it. In essence it is putting the software behind a kind of a fence, which filters the inputs it gets. The fence can be e.g. a firewall, that filters out invalid packets.

Coding is the part that produces the program. It is perhaps the single most failures having step. Doing this incorrectly means that a single developer could cause a huge leak on the system. Note that a leak should never be a fault of a single developer, especially so if we’re talking about a trainee. Code should almost always be reviewed, even more so if it handles sensitive data.

The concept of static analysis belongs to coding part. *Static program analysis* looks at the source code and reports what it has found as being questionable. There are various static analysis tools that are usually language indepent. The problem with most static analysis tools are that there are many false positives. This in effect makes the developers question the result of analysis, lowering or even negating the value of static analysis.
One family of static analysis tools are LINT tools. LINT tools are used while coding, and it notifies of the developer when certain part of the code doesn’t match to certain guidelines. These tools are especially useful with languages that are purely interpreted. Some say they increase productivity. Others do not like these. It’s hard to say who is right. “a fool with a tool is still a fool”, said some student from the lectures.

Quality assurance or testing could include use of a fuzz testers where applicable. These were covered in first lecture. Another event that takes place here is the user and automated testing. Testing should include the cases for testing whether the security functionality of the software works as expected. Below are types of testing that could be included.

- Unit testing — both positive and negative tests. Does single features (e.g. hashing) work as expected.
- System testing — Does the system as a whole work right. Does invalid/malformed data break the system (fuzzing).
- Acceptance testing — ”Attacker stories”. Does the system work actually as it was required, i.e. are user passwords properly derived, does the system really require the use of certificates.
- Penetration/intrusion testing — e.g. Hire an external security expert to try to find weaknesses in the system, or also in-house if proper expertise is available.

5 Secure coding guidelines or standards

There exists a number of guidelines and standards about how the software development process could/should be made secure. We’ll go over a few here.

BSIMM (The Building Security In Maturity Model) is a project by consultancy company Cigital. It has annual updates and is based on real world data by interviewing companies. Summary data is released under permissive license. Similar to BSIMM is OpenSAMM (Software Assurance Maturity Model). It has distant roots with BSIMM, but it is not based on actual research.. OpenSAMM was released in 2009 and has been on hiatus since. However, it is being reinvigorated.

The actual ISO standard ISO-27034 #1 is the newest member of ISO 27000 series. Getting the standard means having to spend a few hundred euros. The standard specifies the model used for secure software development, vocabulary and concepts for an organization. Uses Microsoft SDL as an example. Mircosoft SDL (secure development lifecycle) model is the model that microsoft uses. Apparently its been extremely succesful in many ways.

VAHTI is a guideline for secure software development for finnish public sector companies. Currently it is only a guideline, but might raise its status to a standard/requirement (for finnish public sector) later.

References