Software Security: Security in Software Projects

Summary

These lecture notes summaries the key topics discussed on Software Security course in University of Helsinki Wednesday 26.03.2014 help by Antti Vähä-Sipilä, including some remarks from Henri Lindberg's presentation.

Building secure software is mostly a result of well working project team and the knowledge and security awareness of the individuals inside the team. Security frameworks help development teams to adopt security activities that performed during every development phase enables to achieve greater security and cost benefits.

Foundation for security is the knowledge in the house and a process that considers security as one aspect of software development

Software Security Frameworks

Security Frameworks help organizations to adopt and implement a strategy for software security which ensures that security aspects will be considered in every phase of the development cycle. These frameworks present measurable activities that development teams can adopt as parts of the development process.

Security Frameworks divide software development cycle into business level domains that are used to organize related activities. All three well-knows security frameworks that where introduced during the lecture (Microsoft Security Development Life-cycle, Software Assurance Maturity Model, BSIMM-V Software Security Framework) use similar structuring.

Organizations should aim to demonstrate that the development process works, not that the software is not broken from the tested part.
Security activities in software project

Activities introduced by the frameworks ensure the realization of security in software product. These activities should be present during every phase of software development.

Microsoft Security Development Life-cycle presents following structuring and recommends to perform the activities in chronological order to ensure build in security.

<table>
<thead>
<tr>
<th>Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Security Training</td>
</tr>
</tbody>
</table>

**Requirements Finding and Product Management**

- Avoidance of architecture and design level security issues
- Finding functional & non-functional requirements

**Activities**

- Establish Security Requirements
- Create Quality Gates / Bug Bars
- Perform Security and Privacy Assessments

<table>
<thead>
<tr>
<th>Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avoidance of architecture and design level security issues mainly through creating and updating of an attack model</td>
</tr>
</tbody>
</table>

**Activities**

- Use Threat Modeling

<table>
<thead>
<tr>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation level bugs and using security features properly</td>
</tr>
</tbody>
</table>

**Activities**

- Use Approved Tools
- Perform Attack Surface Analysis / Reduction
- Perform Static Analysis

**Verification**

- Automated testing: static code analysis and security testing

**Activities**

- Perform Dynamic Analysis
- Perform Fuzz Testing
- Conduct Attack Surface Review

<table>
<thead>
<tr>
<th>Release and Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug management and support operations</td>
</tr>
</tbody>
</table>

**Activities**

- Create an Incident Response Plan
- Conduct Final Review
- Certify Release and Archive

**Response and Decommissioning**

- Data retention and erasure

**Activities**

- Execute Incident Response Plan
Security Testing

It is very typical that organizations begin adopting security work as a part of the development process from implementing security testing. Although security testing by nature only tries to find parts of the software that is already broken.

*Security testing demonstrates exploitability, but does not deliver value to the project which implies that a change in mentality is needed in security work.*

Typical Security Tools

- Linters – Pylint, JLint
- Static Code analyzers - FindBugs
- Dynamic code analyzers - Valgrind, Avalanche
- Fuzzers
- Penetration testing tools

Sources


http://bsimm.com/online/

[3] Valtiovarainministeriö, Sovelluskehityksen tietoturvaohje, VAHTI.  
https://www.vm.fi/vm/fi/04_julkaisut_ja_asiakirjat/01_julkaisut/05_valtionhallinnon_tietoturvalisuuks/20130207Sovell/name.jsp

[4] OpenSAMM.  
http://www.opensamm.org/

https://wiki.helsinki.fi/download/attachments/127960693/Session3.pdf?version=1&modificationDate=1395841684183&api=v2