Uni App Security Notes

Felix Pojtinger

October 12, 2021

```
Contributing
   License
Organization
Overview
   Elements of a Secure Development Process
   Support Hierarchy
Basics
   What is Secure Software?
   What is Security?
   CISSP Domains/Certificates
   Why Security?
   Common Terms
   Threat Agents
```





Contributing

These study materials are heavily based on professor Heuzeroth's "Anwendungssicherheit" lecture at HdM Stuttgart.

Found an error or have a suggestion? Please open an issue on GitHub (github.com/pojntfx/uni-appsecurity-notes):



Figure 1: QR code to source repository

If you like the study materials, a GitHub star is always appreciated :)



License



Figure 2: AGPL-3.0 license badge

Uni App Security Notes (c) 2021 Felix Pojtinger and contributors

SPDX-License-Identifier: AGPL-3.0



Organization

- ▶ 60 Minutes of test at the end
- Will have practical examples
- ▶ Threat detection plays a fundamental role in tests





Elements of a Secure Development Process

Primary purpose: Analysis of the data flow; data is both protected by the GDPR and represents value of the corportation

- Requirements
 - Security-Requirements
 - Anti-Requirements
 - Abuse cases
 - Protection poker
 - ▶ → Security analysis/architecture analysis
- Draft
 - AuthN/AuthZ
 - Drafting concepts
 - Risk modelling
- Implementation
 - Secure implementation guidelines
 - Code review, dynamic analysis
- Tests
 - Security testing plans
 - Security testing cases
 - Ethical hacking, pentesting, dynamic analysis



Support Hierarchy

- ▶ Level 1: Direct support with customers; call center, non-technical
- ▶ Level 2: People who know about typical problems with the software
- **Level 3**: Developers of the software

Basics



What is Secure Software?

- ► Software which is protected against intentional attacks
- Every participant in the software development process should be interested in this objective
- Software must be hardened against all known attacks (and future, unknown attacks)



What is Security?

- $ightharpoonup Risk = rac{Cost\ of\ breach}{Probability\ of\ breach}$
- A system is protected against threats compromising valuable data using measures which lead to a reduced, accepted risk.
- ▶ Accepted risk is defined by context of use (i.e. nuclear power: very low accepted risks)
- ► Safety: Protection of the environment from the functional effects a system
- ➤ **Security**: Protection of the system from threats from the environment
- Concrete definitions: uni-itsec-notes#security-objectives; most importantly ("CIA objectives"):
 - Confidentiality
 - Integrity
 - Availability
- ▶ If there are contractions between the security objectives (anonymity vs. accountability): The context defines which objectives dominate over others



CISSP Domains/Certificates

- Security Engineering: Engineering and Management of Security
- ➤ Security Assessment and Testing: Designing, Performing and Analyzing Security Testing
- ➤ **Security Operations**: Foundational Concepts, Investigations, Incident Management and Disaster Recovery
- ➤ Software Development Security: Understanding, Applying and Enforcing Software Security
- ightharpoonup This course strives for 80% of TPSSE compliance



Why Security?

- Security is context dependent: On localhost and unprotected UNIX socket isn't an issue, but forward it with socat and it becomes a massive security vulnerability!
- With every change every test needs to be run again (regression testing)
- ▶ Typically ~30 errors in every 1000 lines of code
- Growing application complexity
- Devices are more and more connected which reduces the need for physical access
- Extensible architectures



Common Terms

- Exploit/Proof of Concept
- Attack
- Vulnerability
- Threat
- Error
- 1. Threat agent gives rise to threat
- 2. Threat exploits vulnerability
- 3. Vulnerability leads to risk
- 4. Risk can damage asset and causes exposure
- 5. Exposure can be countermeasured by a safeguard
- 6. Safeguard directly affects threat agent



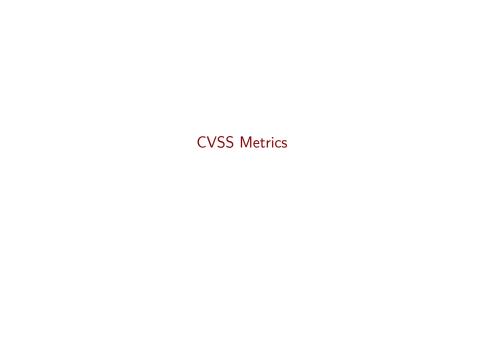
Threat Agents

- Virus (i.e. infection)
- Hacker (i.e. unauthorized access)
- User (i.e. wrong config, data loss)
- Fire (i.e. damage to computers)
- Worker (i.e. leaking)
- Other corporations (i.e. industrial espionage)
- Black hats (i.e. buffer overflows, DoS)
- Intruders (i.e. physically stealing drives)



Researching Vulnerabilities

- Classifying vulnerabilities by severity (low, middle, high)
- Classifying vulnerabilities by exploit range (local or remote)
- Intents to find trends and attacks
- Intents to find vulnerabilities before they can be exploited
- Intents to find countermeasures



CVSS Metrics

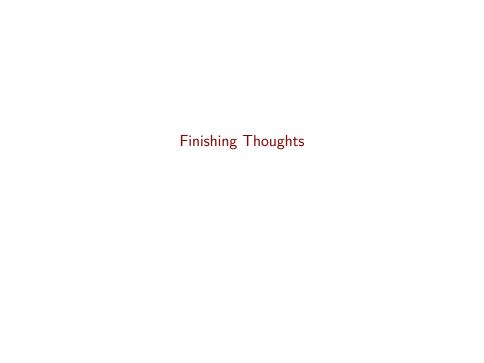
Results in a number which can be used to classify the vulnerability.

- **▶** Base Score Metrics
 - Exploitabilility Metrics
 - ► AV: Attack Vector: Network, Adjacent Network, Local, Physical
 - ► AC: Attack Complexity: Low, High
 - **PR: Privileges Required**: None, Low, High
 - UI: User Interaction: None, Required
 - **S: Scope**: Unchanged, Change
 - Impact Metrics (CIA Metrics)
 - **C: Confidentiality Impact**: None, Low, High
 - I: Integrity Impact: None, Low, High
 - ► A: Availability Impact: None, Low, High
- ► Temporal Score Metrics
 - **E:** Exploit Code Maturity: Not defined, unproven that exploit exists, proof of concept code, functional exploit exists, high
 - RL: Remediation Level: Not defined, official fix, temporary fix, workaround, unavailable
 - ▶ RC: Report Confidence: Not defined, unknown, reasonable,



Balancing Security

- Security is always a balance between functionality and usability
- Security often means to have restrictions in terms of features



Finishing Thoughts

- Systems are only secure if all elements of the system are secure
- Perimeter and infrastructure security can not make the entire system secure
- Applications are always connected
- Development of secure systems is not a choice, but a must!