





Welcome to

13. Software Assessment

KEA Competence OB2 Software Security

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse  

Slides are available as PDF, kramse@Github
13-software-assessment.tex in the repo security-courses

Plan for today



Subjects

- Review using the red book, similarities to green book
- Repetition, common problems, how to improve software security
- How to do Review and audit of software
- Attack and Response
- Attack graphs
- Attack surfaces, and reducing them
- MITRE ATT&CK framework

Exercises

- Layout a plan for securing the Juice Shop

Goals: Software Assessment Introduction



We have learned a lot about insecure coding, in some detail
Often we will not review and assess code on the lowest levels

Photo by NESAs Makers on Unsplash

Software Vulnerability Fundamentals



Vulnerability, bugs, exploiting with an exploit

- AoSSA Chapter 1 refer to a book - which now is available in updated version
Computer Security: Art and Science, 2nd edition 2019! Matt Bishop ISBN: 9780321712332
We use this book in the course *Computer Systems Security*
- Security policies are introduced with a few examples: formal specifications, formal written policy or informal ambiguous collection
- Real organisations have policies from high level mission-type statements down to detailed instructions for operational procedures
- Often organized in a information security management system (ISMS)

Security policy



A security policy defines *secure* for a system or a set of systems.

Matt Bishop, Computer Security 2019

Secure states

Transitions between states, what is allowed

Breach of security - system enters an unauthorized state

Is it possible to return from insecure to a secure state?

Book also defines Confidentiality, Integrity and Availability more precisely

Origin integrity authentication

Military security policy (coinfidentiality) vs commercial security policy (integrity)

Assumptions



Any security policy, mechanism, or procedure is based on assumptions that, if incorrect, destroy the super-structure on which it is built.

Matt Bishop, Computer Security 2019

Example, vendor patches

Important points:

- Is patch correct? Example Spectre and heartbleed
- Vendor test environments equal to intended environments
- Installed correctly - including operator skills

Types of Access Control



Definition 4-13. If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a *discretionary access control (DAC)*, also called an *identity-based access control (IBAC)*

Definition 4-14. When a system mechanism controls access to an object and an individual user cannot alter that access, the control is a *mandatory access control (MAC)*, occasionally called a *rule-based access control*

Quote from Matt Bishop, Computer Security 2019

Examples from real life systems



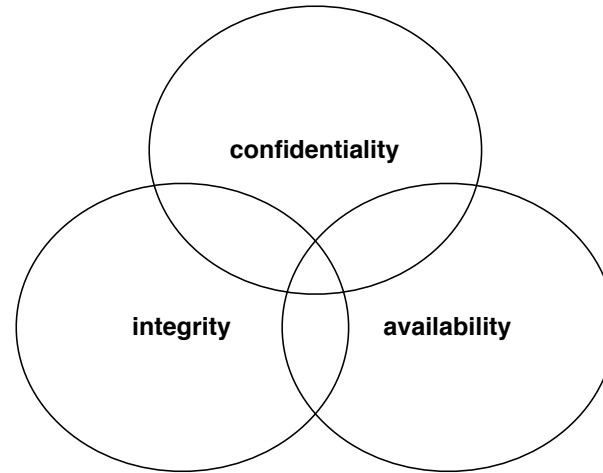
Example systems implementing DAC/MAC:

- Unix file permissions - DAC
- SELinux - Mandatory Access Control architecture to the Linux Kernel
- Sun's Trusted Solaris uses a mandatory and system-enforced access control mechanism

See also: https://en.wikipedia.org/wiki/Discretionary_access_control

https://en.wikipedia.org/wiki/Mandatory_access_control

Confidentiality, Integrity and Availability



We want to protect something

Confidentiality - data kept a secret

Integrity - data is not subjected to unauthorized changes

Availability - data and systems are available when needed

Security is a process



Remember:

- what is information and security?
- Data kept electronically
- Data kept in physical form
- Don't forget the human element of security

Incident Response and Computer Forensics reaction to incidents

Good security is the result of planning and long-term work

Security is a process, not a product, Bruce Schneier

Source for quote: https://www.schneier.com/essays/archives/2000/04/the__process_of__secur.html

Work together



Team up!

We need to share security information freely

We often face the same threats, so we can work on solving these together

How to do Review and audit of software



Auditing an application is the process of analyzing application code (in source or binary form) to uncover vulnerabilities that attackers might exploit

Source: AoSSA chapter 1

- Identify and close security holes
- Auditing vs black box testing
- Black box testing is quick to get started, like fuzzing
- Example in book, a certain input mode triggers a special code path with `sprintf`
- Incorporate Code Auditing into the Secure Software Development Lifecycle (SDLC)

Software Development Lifecycle



A full lifecycle approach is the only way to achieve secure software.

–Chris Wysopal

- Often security testing is an afterthought
- Vulnerabilities emerge during design and implementation
- Before, during and after approach is needed

Secure Software Development Lifecycle



- SSDL represents a structured approach toward implementing and performing secure software development
- Security issues evaluated and addressed early
- During business analysis
- through requirements phase
- during design and implementation

Design vs Implementation



Software vulnerabilities can be divided into two major categories:

- Design vulnerabilities
- Implementation vulnerabilities

Even with a well-thought-out security design a program can contain implementation flaws.

Common Secure Design Issues



- Design must specify the security model's structure
Not having this written down is a common problem
- Common problem AAA Authentication, Authorization, Accounting (book uses audited)
- Weak or Missing Session Management
- Weak or Missing Authentication
- Weak or Missing Authorization

Operational Vulnerabilities



- Security Problems that arise through the operational procedures
- General use of a piece of software in a specific environment
- How the software interacts with its environment
- Recommendation, automate deployment!
- Taking days to install, configure and deploy software is a **red flag**

Design Review



- Design review is a useful tool for identifying vulnerabilities in application architecture and prioritizing components for implementation review
- Optimize the process, which parts are most security critical
- Algorithms, problem domain logic, key algorithms
- Trust relationships and trust boundaries

Enforcing Security Policy



- AoSSA repeats the same problems:
- Authentication
- Untrustworthy credentials
- Insufficient Validation
- Authorization
- Accountability



- Encryption algorithms
- Book describes the common terms used in this area
- Use the *IT Security Guidelines for Transport Layer Security (TLS)* from the dutch National Cyber Security Centre
<https://english.ncsc.nl/publications/publications/2019/juni/01/it-security-guidelines-for-transport-layer-security-tls>
- and Cisco *Next Generation Cryptography*
https://tools.cisco.com/security/center/resources/next_generation_cryptography

Application Architecture Modelling



- Unified Markup Language (UML)
- Class diagrams
- Component diagrams
- Use cases
- Data flow diagram (DFD)
- Processes
- Data stores
- External entities
- Data flow
- Trust boundary
- Threat identification is the process of determining an applications security exposure based on your knowledge of the system

Microsoft Secure Development Lifecycle



There are five major threat modeling steps:

- Defining security requirements.
- Creating an application diagram.
- Identifying threats.
- Mitigating threats.
- Validating that threats have been mitigated. Threat modeling should be part of your routine development lifecycle, enabling you to progressively refine your threat model and further reduce risk.

Sources:

<https://www.microsoft.com/en-us/securityengineering/sdl>

<https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling>

Risk-Based Security Testing



Focus testing on areas where difficulty of attack is least and the impact is highest.

–Chris Wysopal

Time and resources are constrained

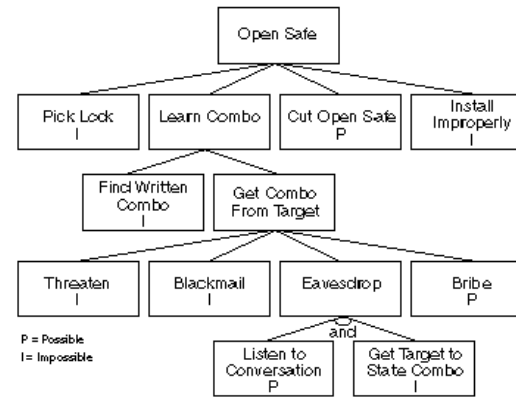
Software development must be prioritized

Threat modelling / risk modelling exist to help this

- Identify threat paths
- Identify threats
- Identify vulnerabilities
- Rank/prioritize the vulnerabilities

Sounds easy enough, harder to do

Attack trees



- Attacks can be said to be based on a chain of dependencies, or graphs
- To achieve goal, need to achieve sub goal x, y, and z – Break the chain and the attack fails!
- Simple example, installing updates remove a dependency for a vulnerability
- Attack trees, picture from Bruce Schneier Attack Trees article December 1999:

https://www.schneier.com/academic/archives/1999/12/attack_trees.html

Exposure, Attack surfaces, and reducing them



- Incident prevention
- Real-time intrusion detection systems (IDS/IPS)
- **Definition 27-7** An *attack surface* is the set of entry points and data that attackers can use to compromise a system.
- Reducing the chance of success also helps, randomization
- Use stack and heap protection
- Address space layout randomization (ASLR) is a host-level moving target defense.
- OpenBSD even randomizes the kernel on install – kernel address randomized link (KARL)
- Limit number of listening services, change insecure defaults, implement access control and firewalls
- Remove anything but the necessary request methods on web servers GET, HEAD and POST
- Restrict access to administrative interfaces
- Implement network segmentation

MITRE ATT&CK framework



MITRE ATT&CK™ is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community.

With the creation of ATT&CK, MITRE is fulfilling its mission to solve problems for a safer world — by bringing communities together to develop more effective cybersecurity. ATT&CK is open and available to any person or organization for use at no charge.

ATT&CK™

Great resource for attack categorization <https://attack.mitre.org/>

Security Assessment Frameworks



- Structured approach to testing, finding and eliminating security flaws
- Information Systems Security Assessment Framework ISSAF
- Penetration Testing Execution Standard (PTES)
- PCI Penetration testing guide, Payment Card Industry Data Security Standard (PCI DSS)
- Technical Guide to Information Security Testing and Assessment (NIST800-115) (GISTA)
- Open Source Security Testing Methodology Manual (OSSTMM)
- CREST Penetration Testing Guide

Which one to choose?

From the book Bishop and https://www.owasp.org/index.php/Penetration_testing_methodologies

Application Review Process



- Rationale - to be successful, any process you adopt must be pragmatic, flexible, and results driven
- Code review is a fundamentally creative process, is that true?
- Code review is a skill
- Preassessment, application review, documentation/analysis, remediation support
- Scoping, information collection

Code-Auditing Strategies



- Code comprehension (CC) strategies, analyze source code directly to discover vulnerabilities
- Candidate point (CP) strategies, two-step create list of potential issues, and examine the source code to determine the relevance of these issues
- Design generalization (DG) strategies, analyzing potential medium- to high-level logic and design flaws

Code Auditor's Toolbox



- Source Code Navigators - good editors and Integrated development environment (IDEs)
- Cross referencing, text searching, multiple language support, syntax highlighting, graphing capabilities, scripting capabilities
- Note: new tools are being developed all the time, book examples are older

Exercise



Now lets do the exercise

Securing the JuiceShop

which is number **43** in the exercise PDF.

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools