Chapter 3: Security Risk

Raimundas Matulevičius
University of Tartu, Estonia, rma@ut.ee

Goal

- Define what is security risk and what are its main constituents:
  - Threat agent
  - Attack method
  - Threat
  - Vulnerability
  - Event
  - Impact
What is **Security engineering**?

[Firesmith, 2003]

Security engineering is concerned with lowering the risk of intentional unauthorized harm to valuable assets to level that is acceptable to the system’s stakeholders by preventing and reacting to malicious harm, misuse, threats, and security risks.

---

**Risk Analysis**

1. Identify explicitly what system assets are targeted
2. Use explicit knowledge and previous expertise to characterize potential vulnerabilities of the considered system assets
3. Security analyst should impersonate himself as the threat agent (motive, capabilities, means, opportunities)
4. Attack method should be explicitly stated including its major steps
5. Elicit and state what is the impact of the defined risk event
6. Once all components are gathered state what is the risk
Risk Analysis

1. Identify explicitly what **system assets** are targeted
2. Use explicit knowledge and previous expertise to characterize potential vulnerabilities of the considered system assets
3. Security analyst should impersonate himself as the threat agent (motive, capabilities, means, opportunities)
4. Attack method should be explicitly stated including its major steps
5. Elicit and state what is the impact of the defined risk event
6. Once all components are gathered state what is the risk

Security Risk Management
Domain Model
Processing of Information

Everything that IT does, reduces to six functions

- **Capturing information**
  - Keyboard, bar code reader, digital camera
- **Transmitting information**
  - Wired-, wireless-phone
- **Storing information**
  - Hard disk, memory card, internet
- **Retrieving information**
  - From any storage device
- **Manipulating information**
  - Calculations, combinations of data
- **Displaying information**
  - Monitor, printer
Functional Decomposition

- **User interaction**
  - Interfacing and/or interacting with users

- **Data/storage management**
  - Storing and management of applications or information

- **Resource management**
  - Resource allocation, global scheduling, process migration,
  - Dynamic configuration of active software components

- **Distribution control**
  - Component collaboration
  - Coordination of local/remote execution
  - Synchronization/concurrency control

- **Communication**
  - Network communication

- **Addressing**
  - Address, identifier and/or name allocation, distribution and discovery/lookup

Risk Analysis

1. Identify explicitly what **system assets** are targeted
2. Use explicit knowledge and previous expertise to characterize potential **vulnerabilities** of the considered system assets
3. Security analyst should impersonate himself as the threat agent (motive, capabilities, means, opportunities)
4. Attack method should be explicitly stated including its major steps
5. Elicit and state what is the impact of the defined risk event
6. Once all components are gathered state what is the risk
Security Risk Management Domain Model

- Common Vulnerabilities and Exposures: [https://cve.mitre.org/cve/](https://cve.mitre.org/cve/)
- National vulnerability database: [https://nvd.nist.org](https://nvd.nist.org)
- OWASP vulnerability database: [https://www.owasp.org/index.php/Vulnerability](https://www.owasp.org/index.php/Vulnerability)
Seven Pernicious Kingdoms

1. Input validation and Representation
2. API Abuse
3. Security Features
4. Time and State
5. Error Handling
6. Code Quality
7. Encapsulation

* Environment

1. Caused by
   - Meta-characters, alternate encoding, numeric representation

2. Problems result from
   - Lack of input validation
   - Representation issues
     - easy target
     - often-used point of attack

3. Resulting problems
   - Buffer overflows
   - Cross-site scripting attacks
   - SQL injection
   - Command injection
   - Setting Manipulation
   - String termination error
   - ...

[Tsipenyuk et al., 2005]
1. Input validation and Representation

2. API Abuse
   - An API is a contract between a *caller* and a *callee*

3. Security Features
   - Software security is not security software

4. Time and State

5. Error Handling

6. Code Quality

7. Encapsulation

* Environment

**Caused by**
- The callee failing to honor its end of the contract

**Case**
- If a program fails to call `chdir()` after calling `chroot()`, it violates the contract that specifies how to change the active root directory in a secure fashion

**Resulting problems**
- Dangerous functions
- Often misused exception handling
- Unchecked return value
- Directory restriction
- ...

**Caused by**
- Incorrect implementation
- Incorrect use of security features like:
  - Authenticity, Access control, Cryptography, Digital signatures, ...

**Resulting problems**
- Insecure randomness
- Missing access control
- Password management
- Privacy violation
- ...

© Springer International Publishing AG 2017
1. Input validation and Representation
2. API Abuse
3. Security Features
4. Time and State
5. Error Handling
6. Code Quality
7. Encapsulation

- **Caused by**
  - Distributed computing
  - Sharing state
  - Multithreading
  - A programmer’s belief that everything happens in one continuous sequence

- **Resulting problems**
  - Deadlock
  - Failure to begin a new session upon authentication
  - Insecure temporary file
  - File access race condition (TOCTOU)
  - ...

*Environment*

- **Caused by**
  - Unexpected input – unexpected behaviour
  - Concept of exceptions in modern programming languages
  - Insecure use and handling of exceptions

- **Resulting problems**
  - Catch NullPointerException
  - Empty catch block
  - Overly-broad catch block
  - Overly broad throw declaration
  - Unchecked return value

© Springer International Publishing AG 2017
1. Input validation and Representation  
2. API Abuse  
3. Security Features  
4. Time and State  
5. Error Handling  
6. Code Quality  
7. Encapsulation  
* Environment

**Caused by**
- Lack of compliance with design  
- Insecure code is buggy code  
- Code readability  
- Complexity of code vs. complexity the human brain is able to manage  
- Forgetting to remove old code

**Resulting problems**
- Inconsistent implementation  
- Memory leak  
- Obsolete code  
- Underfined behaviour  
- Uninitialised variable  
- Unreleased resource  
- Use after free  
- ...

---

© Springer International Publishing AG 2017
1. Input validation and Representation
2. API Abuse
3. Security Features
4. Time and State

5. Error Handling
6. Code Quality
7. Encapsulation

* Environment

• Caused by
  o The software you are developing running on a machine...
  o ... with an operating systems...
  o ... and other software...
  o ... quite possibly connected to other machines through a network...
  o ... all the stuff that is outside the code but is still critical to the security of the created software

• Resulting problems
  o Misconfiguration issues
  o Insecure compiler optimisation
  o ...

Risk Analysis

1. Identify explicitly what system assets are targeted
2. Use explicit knowledge and previous expertise to characterize potential vulnerabilities of the considered system assets
3. Security analyst should impersonate himself as the threat agent (motive, capabilities, means, opportunities)
4. Attack method should be explicitly stated including its major steps
5. Elicit and state what is the impact of the defined risk event
6. Once all components are gathered state what is the risk
Characteristics of Threat agent
[ben Othmane et al. 2014; Shostack 2014]

- **Opportunity** – time to perform a successful attack

<table>
<thead>
<tr>
<th>Object type</th>
<th>Capability examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware device</td>
<td>Physical access</td>
</tr>
<tr>
<td>Software component</td>
<td>Local access (e.g., copying); Communicate with the component from within the local domain; Communicate with the component from outside the local domain.</td>
</tr>
<tr>
<td>Database</td>
<td>Query the database and modify the data files.</td>
</tr>
<tr>
<td>Communication link</td>
<td>Intercept messages and modify messages, or remove messages.</td>
</tr>
</tbody>
</table>

- **Motivation**
  - Curiosity
  - Personal fame
  - Personal gain
  - National interest
  - Legitimate
  - Illegitimate

- **Expertise**
  - Script kiddy
  - Undergraduate
  - Experts
  - Specialist
**Threat Agent Expertise**

[Shostack, 2014]

- **Script kiddie**
  - Use the tools an applications created by others
  - No real system knowledge
  - Follow instructions and use different available tools

- **Undergraduate**
  - Mostly use tools and applications other write
  - Can do minor modification (to these tools)
  - Lack skills to do anything more than twist and adjust a few dials and settings

- **Expert**
  - Supply tools and applications
  - Write worms and viruses
  - Write applications that snoop networks for weaknesses
  - Comfortable by working in kernel, reading protocol traces

- **Specialist**
  - Specialised training
  - Access critical resources
  - Do not leave traces of their work
  - Work carefully and methodically
  - Snooping and breaking computer is their job

**Characteristics of Threat agent**

[ben Othmane ET AL, 2014; Shostack, 2014]

- National interest
- Personal gain
- Personal fame
- Curiosity

- Spy
- Thief
- Vandal / Trespasser / Author
- Vandal
- Author

© Springer International Publishing AG 2017
Risk Analysis

1. Identify explicitly what **system assets** are targeted?
2. Use explicit knowledge and previous expertise to characterize potential **vulnerabilities** of the considered system assets
3. Security analyst should impersonate himself as the **threat agent** (motive, capabilities, means, opportunities)
4. **Attack method** should be explicitly stated including its major steps
5. Elicit and state what is the impact of the defined risk event
6. Once all components are gathered state what is the risk

Security Risk Management Domain Model
Security Risk Management
Domain Model

STRIDE
(Microsoft)

- **Spoofing** – pretending to be someone/something than intended
- **Tampering** – modifying something on the disk/file/network/memory
- **Repudiation** – claiming that someone did not do something or were not responsible
- **Information disclosure** – providing information to someone not authorised to see it
- **Denial of service** – absorbing resources needed to provide the service
- **Elevation of privileges** – allowing someone to do something they are not authorised to do
Threats to (Distributed) System
(first level threats)

• Identify attacks
• Network communication attacks
• Network protocol attacks
• Passing illegal data attacks
• Stored data attacks
• Remote information inference
• Loss of accountability
• Uncontrolled operations

[Uzunov and Fernandez, 2014]

Threats to (Distributed) System
(first level threats)

• **Identify attacks**
  • Network communication attacks
  • Network protocol attacks
  • Passing illegal data attacks
  • Stored data attacks
  • Remote information inference
  • Loss of accountability
  • Uncontrolled operations

• Attacker attempts to fabricate or misuse identities in a system

• **Examples**
  – Identity spoofing
  – Advantageous identity allocation
Threats to (Distributed) System
(first level threats)

• Identify attacks
• **Network communication attacks**
  • Network protocol attacks
  • Passing illegal data attacks
  • Stored data attacks
  • Remote information inference
  • Loss of accountability
  • Uncontrolled operations

• Threats to communication between distributed components

• **Examples**
  – Message secrecy violation
  – Message integrity violation
  – Message authenticity violation
  – Traffic analysis, protocol sniffing
  – Covert network channel
  – Session hijacking
  – Session state poisoning
  – Route poisoning
  – Message flooding

© Springer International Publishing AG 2017
Threats to (Distributed) System
(first level threats)

Threats to communication between distributed components

Examples

- Message secrecy violation
- Message integrity violation
- Message authenticity violation
- Traffic analysis, protocol sniffing
- Covert network channel
- Session hijacking
- Session state poisoning
- Route poisoning
- Message flooding

Remote information inference
- Loss of accountability
- Uncontrolled operations

Messages in transit
- intercepted and their contents read by an attacker
- intercepted and modified, replaced, corrupted or simply deleted by an attacker

© Springer International Publishing AG 2017
Threats to (Distributed) System

(first level threats)

- Identify attacks
- Network communication attacks
- Network protocol attacks
- Passing illegal data attacks
- Stored data attacks
- Remote information inference
- Loss of accountability
- Uncontrolled operations

Threats to communication between distributed components

Examples
- Message secrecy violation
- Message integrity violation
- Message authenticity violation
- Traffic analysis, protocol sniffing
- Covert network channel
- Session hijacking
- Session state poisoning

Secure communication
- message encryption
- message hashing, error detection codes
Threats to (Distributed) System
(first level threats)

- Identify attacks
- Network communication attacks

**Network protocol attacks**
- Passing illegal data attacks
- Stored data attacks
- Remote information inference
- Loss of accountability
- Uncontrolled operations

- Threats specifically to the network protocols used for communication

**Examples**
- Message replay
- Message reuse
- Protocol field modification
- Use of abnormal packet size
- Use of abnormal package sequencing
- Use of reserved protocol packet

Threats to (Distributed) System
(first level threats)

- Identify attacks
- Network communication attacks
- Network protocol attacks

**Passing illegal data attacks**
- Stored data attacks
- Remote information inference
- Loss of accountability
- Uncontrolled operations

- Input data is manipulated by attacker for some malicious purpose

**Examples**
- Injection
Threats to (Distributed) System
(first level threats)

- Identify attacks
- Network communication attacks
- Network protocol attacks
- Passing illegal data attacks
- **Stored data attacks**
  - Remote information inference
  - Loss of accountability
  - Uncontrolled operations

- **Threats on storage data**
  - **Examples**
    - Corruption

- Extracting information from a component or remotely, i.e., over network
  - **Examples**
    - Scanning (information gathering)
    - Probing (vulnerability checking)
    - Output information disclosure
    - Data inference
Threats to (Distributed) System
(first level threats)

- Identify attacks
- Network communication attacks
- Network protocol attacks
- Passing illegal data attacks
- Remote information inference
- 
  Loss of accountability
- Uncontrolled operations

- Impact accountability attributes

- Examples
  - Track erasing
  - Repudiation

- Exploits existing system functionality in ways that would not normally be allowed (e.g., race conditions, access to data)

- Examples
  - Unauthorized access
  - Invoking unauthorized operations
  - Spoofing privileged processes
  - Unsafe code execution
  - Exploitation of tight component coupling
  - Process overflow attack
  - Exploiting concurrency flaws
  - Resource exhaustion
  - Targeted process crashing
Threats to Security Infrastructure
(second level threats)

- Cryptography attacks
- Countermeasure design
- Configuration/administration
- Network protocol threats

[Uzunov and Fernandez, 2014]
Threats to Security Infrastructure
(second level threats)

• Cryptography attacks
• Countermeasure design
• Configuration/administration
• Network protocol threats

• Threats to countermeasures using cryptography

• Examples
  – Forging cryptographic credentials
  – Abuse of weak algorithm
  – Exploiting vulnerable security protocol
  – Password attacks (guessing, brute force, rainbow tables)

Threats to Security Infrastructure
(second level threats)

• Cryptography attacks
• Countermeasure design
• Configuration/administration
• Network protocol threats

• Threats to the way certain countermeasures are (or may be) designed

• Examples
  – Use of default credentials
  – Bypassing controls
  – Leveraging authorization model
Threats to Security Infrastructure  
(second level threats)

- Cryptography attacks
- Countermeasure design
- **Configuration/administration**
- Network protocol threats

- Threats related to configuration and/or administration of the security system
  
  - **Examples**
    - Exploiting bad policies
    - Unauthorized modification of rights

- Also at the first level
  - Threats applicable to secure protocol design
Risk Analysis

1. Identify explicitly what **system assets** are targeted?
2. Use explicit knowledge and previous expertise to characterize potential **vulnerabilities** of the considered system assets
3. Security analyst should impersonate himself as the **threat agent** (motive, capabilities, means, opportunities)
4. **Attack method** should be explicitly stated including its major steps
5. **Elicit and state what is the impact of the defined risk event**
6. Once all components are gathered state what is the risk

Security Risk Management
Domain Model

---

© Springer International Publishing AG 2017
Malicious software
https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101/

• Software used to cause harm to the computer
  – Compromise computer functions
  – Steal data
  – Bypass access controls
  – ...

• Symptoms:
  – Increased CPU usage
  – Slow computer or web browser speeds
  – Problems connecting to networks
  – Freezing or crashing
  – Modified or deleted files
  – Appearance of strange files, programs, or desktop icons
  – Programs running, turning off, or reconfiguring themselves
  – Strange computer behavior
  – Emails/messages being sent automatically and without user’s knowledge
Malicious software

[https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101/]

• Symptoms:
  – Increased CPU usage
  – Slow computer or web browser speeds
  – Problems connecting to networks
  – Freezing or crashing
  – Modified or deleted files
  – Appearance of strange files, programs, or desktop icons
  – Programs running, turning off, or reconfiguring themselves
  – Strange computer behavior
  – Emails/messages being sent automatically and without user’s knowledge

<table>
<thead>
<tr>
<th>Malware types</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101/">https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101/</a></td>
</tr>
</tbody>
</table>

• **Adware**
  (advertising supported software)
  – automatically delivers advertisements
  – Pop-up ads on Websites
  – Come with spyware
    • to track-user activities
    • to steal information

• **Spyware**
  – spy on user activity without their knowledge
    • activity monitoring
    • collecting keystrokes
    • data harvesting
      (account information, logins, financial data)

© Springer International Publishing AG 2017
Malware types

https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101/

• **Bot**
  – automatically perform specific operations
  – used in botnets – collections of computers to be controlled by third parties
    • DDoS attacks
    • spambots

• **Bug**
  – A flaw that produces an undesired outcome
  – Security bugs are the most severe type of bugs
    • Attackers can
      – bypass user authentication
      – override access privileges
      – steal data

• **Ransomware**
  – holds a computer system captive while demanding a ransom
  – restricts user access to the computer
    • encrypting files
    • locking down the system and displaying messages to force the user to pay the malware creator

• **Rootkit**
  – remotely access or control a computer without being detected
    • remotely execute files
    • access/steal information
    • modify system configurations
    • alter software (especially security software that could detect the rootkit)
    • install concealed malware
    • control the computer as part of a botnet
Malware types
https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101/

• **Virus**
  – copying itself and spreading to other computers
    • by attaching themselves to various programs, executing code when a user launches those programs
    • through script files, documents, and cross-site scripting vulnerabilities in web apps
  – used to
    • steal information, money
    • harm host computers and networks
    • create botnets
    • render advertisements

• **Worm**
  – spread over computer networks by exploiting operating system vulnerabilities
  – harm to host networks by consuming bandwidth and overloading web servers
  – Have the ability to self-replicate and spread independently
  – Worms often spread by sending mass emails with infected attachments to users’ contacts

© Springer International Publishing AG 2017

• **Trojan Horse**
  – disguises itself as a normal file or program to trick users into downloading and installing malware
  – can give a malicious party remote access
  – once an attacker has access to an infected computer
    • to steal data
    • install more malware
    • modify files
    • monitor user activity
    • use the computer in botnets
    • ...

• **Spam**
  – electronic sending of mass unsolicited messages
    • email, instant messages, texting, blogs, web forums, search engines, social media
  – it is very common for malware to spread through spamming
    • when infected computers (with viruses, worms, ...) are used to distribute spam messages containing more malware

© Springer International Publishing AG 2017
Risk Analysis

1. Identify explicitly what **system assets** are targeted?
2. Use explicit knowledge and previous expertise to characterize potential **vulnerabilities** of the considered system assets
3. Security analyst should impersonate himself as the **threat agent** (motive, capabilities, means, opportunities)
4. **Attack method** should be explicitly stated including its major steps
5. Elicit and state what is the **impact** of the defined risk event
6. Once all components are gathered state what is the **risk**

Define Risk Explicitly

© Springer International Publishing AG 2017
Risk Modelling Techniques

- **KAOS extensions to security**
  - Goal modelling
  - Anti-models, Anti-goals and Anti-requirements

- **Abuse frames**
  - Extension of problem frames
  - Anti-requirements be fulfilled by a threat agent

- **Attack trees**
  - Definition and refinement of potential attack

---

**Summary**

1. Identify explicitly what system assets are targeted
2. Use explicit knowledge and previous expertise to characterize potential vulnerabilities of the considered system assets
3. Security analyst should impersonate himself as the threat agent (motive, capabilities, means, opportunities)
4. Attack method should be explicitly stated including its major steps
5. Elicit and state what is the impact of the defined risk event
6. Once all components are gathered state what is the risk