Threat Modeling

Frank Piessens (Frank.Piessens@cs.kuleuven.be)
Overview

• Introduction

• Key Concepts
  – Threats, Vulnerabilities, Countermeasures
  – Example

• Microsoft’s Threat Modeling Process

• Conclusion
Threat modeling

• Threat modeling is an activity early in the software development lifecycle
  – Primary goal: get a good view on possible threats to the system being developed

• Threat modeling can be done on various levels of abstraction
  – System level
    • E.g. Threat modeling of the Internet e-mail system
  – Application or component level
    • E.g. Threat modeling of the e-mail client software
Overview

• Introduction

• Key Concepts
  – Threats, Vulnerabilities, Countermeasures
  – Example

• Microsoft’s Threat Modeling Process

• Conclusion
Owners wish to maximize availability/usefulness of assets that may be reduced by vulnerabilities that may be possessed by threat agents that increase to threats that give rise to risk that may increase to assets that may be aware of impose to reduce countermeasures that may be reduced by may be aware of wish to exploit and/or may damage owner's assets that may increase to risk that increases to
Threats versus Security Goals

• In a first approximation, threats and security goals are each others negation:
  – A security goal is a statement of intent to counter identified threats
  – A threat is the intention of a threat agent to break a security goal
Typical Threats

• Information disclosure
  – Threat: Information leaks to threat agents that should not be able to get access to that information
  – Corresponding Security Goals:
    • Data Confidentiality: protecting data against unauthorized reading
    • Access Control: preventing unauthorized access to software functions
  – Example: stealing of credit card numbers
Typical Threats

• Information tampering
  – Threat: threat agents tamper with data they should not be able to modify
  – Corresponding Security Goals:
    • **Data Integrity**: protecting data against unauthorized writing
    • **Data Origin Authentication**: verifying the claimed identity of the originator of a message
    • **Access Control**: preventing unauthorized access to software functions
  – Example: changing the price of purchased goods
Typical Threats

• Repudiation
  – Threat: threat agent denies involvement in an event
  – Corresponding Security Goals:
    • **Non-repudiation**: the provision of irrefutable evidence about what happened and who was involved
    • **Audit**: chronological record of system activities to enable the reconstruction of events
  – Example: denying placement of a stock order
Typical Threats

• Denial of Service
  – Threat: threat agent destroys the usefulness of the system for legitimate users
  – Corresponding Security Goals:
    • Availability: ensuring availability of the system to legitimate users
  – Example: Distributed Flood Attack
Typical Threats

• Elevation of privilege
  – Threat: threat agent gets more access to information/communication/processing resources than he is authorized for
  – Corresponding Security Goals:
    • **Access Control**: preventing unauthorized access to software functions
    • **Entity Authentication**: verifying the claimed identity of a party one is interacting with
  – Example: “getting root”
Typical Threats

• Spoofing
  – Threat: Threat agent pretends to be someone/something he is not
  – Corresponding Security Goals:
    • Entity Authentication: verifying the claimed identity of a party one is interacting with
    • Data Origin Authentication: verifying the originator of a message
  – Example: phishing
Vulnerabilities

• A vulnerability is an aspect of (a component of) the system that allows a threat agent to realize a threat (i.e. break a security goal)

• Hence, vulnerabilities are relative to the capabilities of the threat agent
  – E.g. the system can be vulnerable to insiders but not to outsiders
Vulnerabilities

- Vulnerabilities arise through failures in:
  - Requirements,
    - Failure to identify all relevant assets, or specific threats
    - E.g. protecting against the threat of downloaded malicious code was not recognized as a requirement for OS security in the eighties
  - Construction,
    - Vulnerabilities in security components
      - E.g. a weak cryptographic algorithm
    - Vulnerabilities in application functionality
      - E.g. buffer overflows, SQL injection, …
  - Operation
    - E.g. Setting an incorrect policy
Vulnerabilities

- Vulnerabilities can exist in the different layers of a software system
  - Application, e.g. SQL injection
  - Programming language runtime, e.g. type confusion
  - Middleware, e.g. open management interface
  - Operating system, e.g. FAT32 file system
  - Hardware, e.g. side-channels
Countermeasures

- Countermeasures are the mechanisms used to:
  - Reduce vulnerabilities, and hence:
    - Realize security goals
    - Counter threats
Countermeasures

• Countermeasures can be:
  – Preventive: avoid vulnerability
  – Detective: detect vulnerability exploitation
  – Reactive: handle incidents

• Countermeasures can be taken by:
  – Software engineers, programmers who develop software
  – Administrators, system managers who deploy software
  – …
Software Engineer Countermeasures

• For vulnerabilities introduced in the requirements phase:
  – Security reqs engineering, threat analysis and modeling

• To address threats collected during requirements engineering
  – Security technologies
    • cryptography, access control mechanisms, authentication mechanisms, …

• For vulnerabilities introduced during construction:
  – Secure programming, static analysis, safe languages, …

• For vulnerabilities introduced during operation:
  – Documentation, operational procedures, secure defaults, …
Administrator Countermeasures

• Preventive countermeasures:
  – deployment of additional protection: Firewalls, VPN’s, …
  – patching weaknesses where possible: CERT advisories, Windows Update, …

• Detective countermeasures:
  – Intrusion Detection software or Fraud Detection software
  – Virus scanning

• Security solutions should be managed, supporting reactive countermeasures
Overview

• Introduction
• Key Concepts
  – Threats, Vulnerabilities, Countermeasures
  – Example
• Microsoft’s Threat Modeling Process
• Conclusion
Simplified e-mail system

User u1

1

mail storage server

2

mail transfer server

mail client

domain1.com

3

mail storage server

mail transfer server

mail client

domain2.com

4

User u2

5

6

7

8
Assets

• E-mail messages
  – header
  – body
• Address book / contact information
• Client and Server machines
  – Storage space
  – Computational resources
  – Mail delivery service
  – Infrastructural software (OS,...)
• Network infrastructure
• …
Threats

• E-mail message
  – Unauthorized reading of message body
    • Define what is “authorized”!
  – Tampering with message
    • In transit
    • In storage
  – E-mail spoofing (tampering with the from: field)
  – Repudiation
    • Of sending
    • Of receipt
  – …
Threats

- **Client and Server machines**
  - Denial of service
    - E.g. Inbox storage space
  - Elevation of privilege
    - Server is a “trusted software layer”, making a limited functionality (sending/receiving mail) available to clients
    - If you can break the server out of this limited functionality (e.g. because of bugs), you can attack the server machine
  - E-mail viruses
    - E.g. through executable attachments
  - Unauthorized use of mail delivery service
    - E.g. Spam e-mail
  - …
Threats

• Not all “undesirable behavior” can easily be related to assets:
  – Detecting when somebody reads his mail
    • Asset = “privacy of the reader”?  
  – …
Vulnerabilities

• Requirements: Security was not a concern in the original design of Internet e-mail
  – As a consequence, plain e-mail system is vulnerable to all identified threats
  – Countermeasures have been designed and integrated over the past decades

• Construction:
  – Implementation bugs in all e-mail components have been exploited
  – Sometimes amplified because of vulnerabilities in OS

• Operation:
  – Insecure configuration of e-mail: open mail relays, sendmail debug mode, …
Possible Countermeasures

• Public key encryption and signing of e-mail, counters:
  – e-mail spoofing
  – tampering / reading messages in transit
  – repudiation of sending

• Password based authentication and access control
  – Counters tampering / reading messages in storage

• Sandboxing of attachments
  – Counters virus-spreading

• Code review or static analysis of server code
  – Counters elevation-of-privilege on server

• …
Vulnerabilities in Countermeasures

- Weak encryption algorithms
- Predictable generation of encryption keys
- Guessable passwords
- False negatives in static analysis for bugs
- ...

Overview

- Introduction
- Key Concepts
  - Threats, Vulnerabilities, Countermeasures
  - Example
- Microsoft’s Threat Modeling Process
- Conclusion
Introduction

• As part of it’s Secure Development Life Cycle, Microsoft has defined a threat modeling activity
  – Supported by documentation and tools
• Goal for the rest of this session:
  – Discuss the structure of this Threat Modeling process
  – Illustrate it with the tool
Overall Structure

• Understand the adversary’s view
  – Entry points
  – Assets
  – Trust Levels

• Characterize the security of the system
  – Use Scenarios
  – Assumptions and dependencies
  – Model the system

• Determine threats
  – Identify threats
  – Analyze threats
Understand the adversary’s view

• Entry points:
  – any location where data or control flow crosses the system’s boundary
  – E.g. sockets, RPC interfaces, GUI elements, …

• Assets:
  – Anything that needs protection
    • Information, resources, functionality

• Trust levels:
  – Categorization of possible threat agents according to “power”
Characterize the security of the system

• Define use scenarios
  – Define how the system will be used, and how it won’t be used
    • Bound the threat modeling discussion

• Identify assumptions and dependencies
  – External dependencies: requirements on systems outside the system being modeled
  – External security notes: security relevant information to users that interface with the system
  – Internal security notes: information for the reader of the threat model
  – Implementation assumptions: Assumptions under which the current threat model was made
Characterize the security of the system

• Model the system
  – Typically modeled using dataflow diagrams
    • External entities
    • Processes
    • Data stores
    • Data flow
    • Privilege boundaries
Example DFD
Example DFD

User via PSTN

1.1 Enable password entering mode

1.2 Read digit

Password ring buffer (stores 8 digits)

1.3 Check if buffer matches password

Successful completion

8-digit password

1.4 Enable admin mode

1.5 Administrative interface

Menu, data

Selection, data

Admin enable

User dials 9 after answering machine picks up

User enters digit in password entering mode

Interface enabled privilege boundary

Digit

Password enter enable
Determine threats

• Determine threats: elicit and categorize threats using the STRIDE categories:
  – Spoofing
  – Tampering
  – Repudiation
  – Information leakage
  – Denial-of-service
  – Elevation of Privilege

• Analyze threats
  – Threat trees decompose threats
  – Risk rating for vulnerabilities, e.g. using DREAD
Overview

• Introduction
• Key Concepts
  – Threats, Vulnerabilities, Countermeasures
  – Example
• Microsoft’s Threat Modeling Process
• Conclusion
Conclusion

• Threat modeling is an activity to be performed in the early phases of the software life cycle
  – In order to understand the “threat profile” of the software
  – In order to select countermeasures in an informed way

• Microsoft places Threat Modeling in the top 2 of most important security related activities
  – Other one is static analysis for implementation vulnerabilities