cigital

Software Security Requirements

The foundation for security

Paco Hope
Technical Manager
paco@cigital.com

Peter White
Managing Consultant
pwhite@cigital.com

cigital
Software Confidence. Achieved.

www.cigital.com
info@cigital.com
+1.703.404.9293

Agenda

- What are Requirements?
- What are Security Requirements?
- How do we put Security Requirements into Real Software?
- Examples:
  - Anti-requirements
  - Abuse Cases
- Bringing it home

cigital

What are Requirements?

- The IEEE Standard 729 defines requirements as:
  - *A condition or capability needed by a user to solve a problem or achieve an objective*
  - *A condition or capability that must be met or possessed by a system…to satisfy a contract, standard, specification, or other formally imposed document.*
- Three Types of Requirements
  - Functional (Behavioral) Requirements
    - Functions that the system must perform
  - Non-Functional Requirements
    - Properties system must possess
  - Derived Requirements
    - Functional/non-functional requirements implicit from stated requirements

Functional Requirements

- Inputs that are expected by the system
- Outputs that must be produced
- Relationships between those inputs and outputs

- Examples of Functional Requirements
  - There shall be four inputs. They shall be buttons, and are named B1, B2, B3, and B4.
  - B1 shall be the "Power On" button
  - B4 shall be the "Power Off" button
  - B2 and B3 shall be "action buttons"
  - After B1 has been pushed but before B4 is pushed, the system shall be termed "Powered On".

Non-functional Requirements

- Auditability
- Extensibility
- Maintainability
- Performance
- Portability
- Reliability
- Security
- Testability
- Usability
- etc.

- Example Non-Functional Requirements
  - The system shall run on Windows XP, Windows Vista, and MacOS X 10.4
  - User logins will take at most 20 seconds from submitting credentials to seeing first screen.
  - The system will require less than 10 Mbs network speed to handle 100 concurrent users.

Attributes of Good Requirements

Testable

Complete

Clear

Consistent

Unambiguous

Measurable

cigital

Adding Security to Requirements

cigital

Software Confidence. Achieved.

Security requirements, not security features

"An increasing number of software organizations recognize that developing security requirements is more important than designing protections because paying attention to security requirements in the early stages of the software lifecycle potentially saves millions of dollars." Qian Gao

- Security is not about features.
- It is typically difficult (or impossible) to patch bad software, and nearly always costly to do so.
- Early consideration of security makes it part of the standard SDLC, and places it on a par with functional requirements.
- **You can't test what you don't specify.**

"75% of all attacks today occur at the application layer and bypass traditional firewalls." Gartner, 2005

cigital

8

New and Old Vocabulary

- Functional security requirement
  - *A condition or capability needed in the system to control or limit the fulfillment of requirements*
- Non-functional security requirement
  - *An emergent property of the system required to ensure fulfillment of requirements in the face of abuse or misuse*
- Derived security requirements
  - From functional requirements
  - From other security requirements

Typical Security Properties

- Confidentiality
  - Access Control
  - Privacy
- Integrity
  - Anti-Corruption
  - Origin Authenticity
- Availability
- Accountability

- Access control
  - "System shall require passwords..."
- Confidentiality
  - "System shall only show docs to authorized users...."

Security non-functional Requirements

- **Audit logs shall be verbose enough to support forensics**
  - *All account modification events shall be logged. The event log shall contain date, time, user, action, object, prior value, new value*
  - *Audit logs shall have integrity protection...*
- **Application use of credit card data shall be PCI compliant. e.g. PCI 3.3:**
  - *Mask PAN when displayed (the first six and last four digits are the maximum number of digits to be displayed).*

 11

Deriving Security Requirements

- **Web App Req 1: All accounts have passwords**
- **Web App Req 2: 3 bad attempts == account lock**
- **Implication: Bad guy can DoS the App**
  - Try every account 3 times
  - All accounts locked
- **Derived requirement:**
  - Accounts should unlock after 5 minutes of no attempts
  - eBay attack

 12

Thinking backwards

- Think of abuse cases and misuse cases as "backward" use cases
- Consider grammatical negation
- Start with use cases
  - Think about what a system does
  - Continue at increasing levels of detail
- Once you know what a system does, look at it from the adversary's perspective.
  - How can they disrupt the system?
  - How can they profit from the system?

An Automated Teller Machine

- Scenario:
  1. Login
  2. Withdraw money
  3. Logout
- What are some example requirements
- How about security requirements?

Login, Withdraw, Logout

- Card required to login
- Correct PIN required to login
- Withdraw even dollar amounts in increments of $20
- Can't exceed account balance

- It's still not good enough
  - What will a bad guy do?

Security Requirements

- Shoulder-surfing
  - Don't display PIN
- Steal card
  - Don't allow lots of login attempts
- Guy behind you uses your forgotten card
  - Audible and visible alerts
  - Session timeout and logout

Anti-requirements: a useful construct

- Requirements generally have the form:

  The system shall [do something] given [inputs]
- To develop an anti-requirement:
  - Categorize the possible outcomes
  - Rank in order of severity from perfect to worst
  - Define a threshold – what outcomes are unacceptable
  - Explore the inputs and determine the outcome associated with each
  - Determine which are acceptable and which are not
  - Associate each input and outcome
- This exploration of the requirements from an "anti" perspective allows you to design security requirements to address unacceptable outcomes from the code that implements a requirement

An example of "anti-requirements"

*Requirement: The system **shall** produce a unique identifier valid for N days into the future **given** a time, an integer N, and a valid authorization token where $0 < N \leq 7$*

Consider Undesirable Outcomes

Crash system
Crash application
Non-unique identified produced
Identifier with incorrect validity period
etc.

- **Address undesirable outcomes in order of business impact**

- **Consider Inputs**

Time is negative
N ≤ 0
N > 7 etc.
N is non-numeric
etc.

- **Map inputs to outcomes**

Bad N = error
Bad time = error
Invalid auth = error
error = invalidate session

Formulate *Positive REQUIREMENTS* to mitigate unacceptable outcomes

Building an abuse case

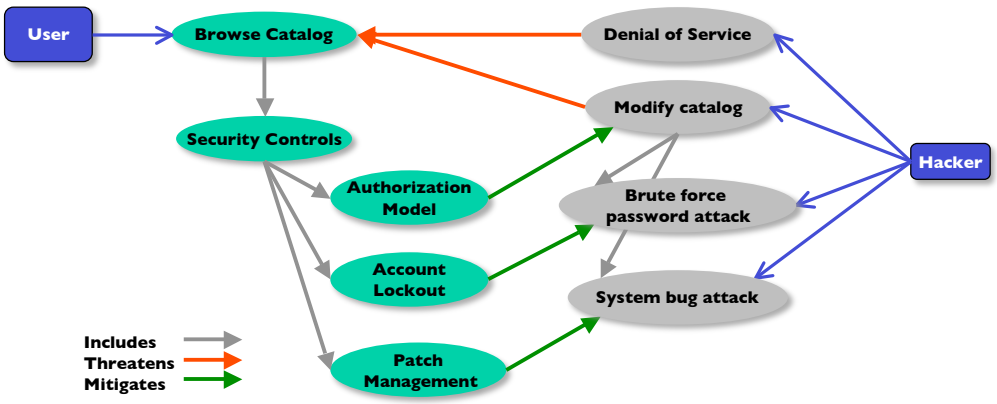
- Start with building an understanding of what the system does. If use cases are available, start with those.
- Consider how the process can be interdicted or corrupted.
- Tell stories, in simple narrative form, about how an adversary can misuse the system to their advantage
- Expand the story using simple graphics (UML is useful, but not necessary)

Move towards requirements:

- Consider how the misuse can be mitigated.
- Build increasingly more specific requirements building a bridge from the business to the technical.

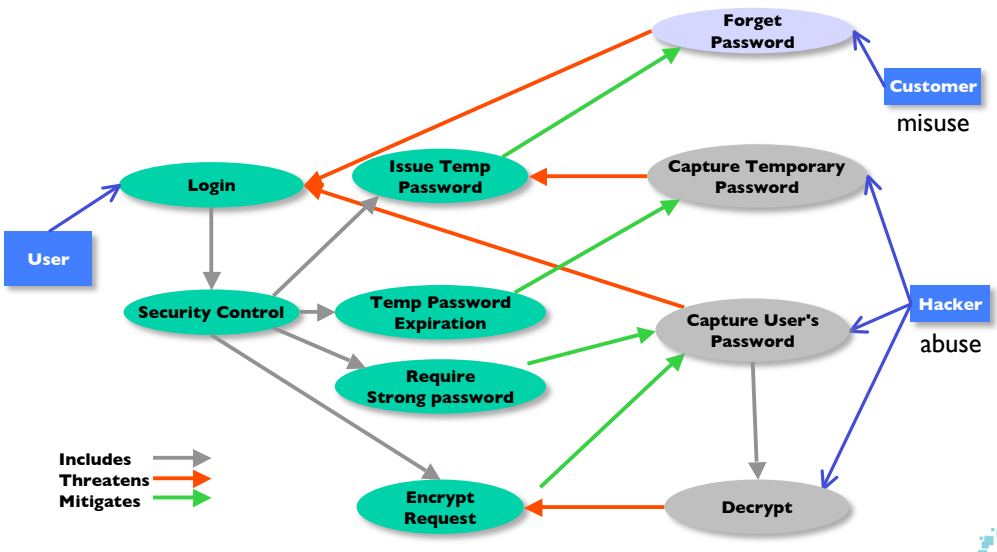
19

Example abuse case: Browse Web Catalog



20

Example abuse case: Login Process



How Do You Do It?

- Ideal: During Initial Requirements
- Next best thing: During Test Strategy
  - Include Security Test Plan as part of strategy
  - Balance security testing based on risks and impacts

The best time to plant an oak tree was twenty years ago. The next best time is now.

—Ancient Proverb

cigital

Questions and Answers

Thank you for your time

cigital

Software Confidence. Achieved.

www.cigital.com
info@cigital.com
+1.703.404.9293