Building Secure and Usable Systems with AEGIS

Ivan Flechais
University of Oxford
Software Engineering Programme
ivan.flechais@comlab.ox.ac.uk
Overview

- Common design strategies in Information Security
- *Philosophy!* Functionalism and Interpretivism in security
- What goes into building secure software and where do we start?
- One design methodology: AEGIS
The basic process of building a (secure) system consists of:
- eliciting requirements
- specifying a design that fulfils these
- implementing the design

Depending on the engineering methodology, these steps are not always separate (e.g. agile methods vs waterfall development vs iterative development)
Common Security Design Strategies

Three types of security design approaches:

- **Ad Hoc**
  - No structure
  - Unpredictable
  - Highly reliant on the ability of the individuals
Common Security Design Strategies

- **Baseline**
  - Checklists & Standards (e.g. ISO 17799, Common Criteria ISO 15408)

**Pros:**
- Specifying a common level of security
- Evaluating different systems against a common level
- Easy & relatively cheap

**Cons:**
- Too generic
- Box ticking exercise
- Doesn’t allow reasoning about security
Common Security Design Strategies

- **Risk Analysis**
  - Looking at Threats, Vulnerabilities and Risk

**Pros:**
- Supports cost benefit reasoning about security
- Suitable for securing a system where baselines don’t exist

**Cons:**
- Time consuming, difficult and complex
- Information about risk is inherently hard to get
- High levels of expertise necessary
Functionalist vs Interpretive Security

- What is functionalist\(^1\) security design?
- What is interpretive security design?
- Some research in interpretive security
- AEGIS design methodology

\(^1\) From *Current directions in IS security research: towards socio-organizational perspectives* - Dhillon & Backhouse 2001
Functionalist Security

- Rationalist
- Objective & expert viewpoint
- Formal & automated parts of information system

Security Literature:
- Checklists
- Evaluation
- Most Risk Analyses
Characteristics of Functionalism

- Objectivist, regulatory & controlling
- Organisations & Information
  Systems have strict boundaries
- Focus on input, throughput, output & feedback
- Security is the result of analysing the constituent elements
Problems of Functionalist Security

- Narrow focus
- Expertise comes from outside
- Prescriptive/Constraining
- Real-world issues of:
  - Motivation
  - Compliance
  - Understanding
Interpretive Security

- Philosophy: individual consciousness & subjectivity
- Social implications of security are considered
- Longer history in information systems research than computer security
Interpretive Security Design - 1

- Straub & Welke (1998) ¹
- Principle: Provide security know-how to managers
- Aim: Give managers a sound basis for security decisions -> better security
- Intervention: risk planning tool, security training, countermeasure matrix
- Results: Improvements in practical security in the organisations

Helen James (1996) ¹: Orion Strategy for security design (based on Soft Systems Methodology - Checkland)

Principle: Involve stakeholders (managers) in the design process

Aim: Improve awareness of security

Intervention: Soft Systems Methodology for security

Results: Raised awareness and understanding of security. Improved motivation & ownership of security too.

Why is this important?

Historically, security has arisen from the military (functionalist):

- Very specific **environment**
  - People are trained to follow orders
  - Strong hierarchy
- Very specific **goals**
  - Secrets *must* be protected
  - Cost of breach is national security
- How comparable is the military to business?
Why is this important?

- Businesses have different environments:
  - Employees are not trained to follow orders
  - Hierarchies in businesses are not as important
  - Enforcing security is harder than in the military

- Goals:
  - Businesses may not perceive security as obviously important
  - Motivation for having good security is harder than in the military
Security requirements

- What are they?
- Functional vs. non-functional?
- Elicitation
Motivation for requirements

- Lack of satisfactory definition of security requirements
- Why is it important?
  - Ross Anderson’s\(^1\) report gives a view of the security goals of a Clinical Information System from the point of view of the doctors

\(^1\) The DeCODE Proposal for an Icelandic Health Database - Ross Anderson
http://www.cl.cam.ac.uk/~rja14/iceland/iceland.html
Motivation for requirements - 2

- Consider two hypothetical sets of candidate security requirements for a Clinical Information System:
  1. proposed by the doctors, some actions are considered legitimate for doctors, but prohibited for administrators
  2. proposed by the administrators, the situation is reversed; some actions that would have been legitimate by the standards of report 1 are security violations, and vice versa

- Need security requirements to know what the violations are!
Cost of security errors

Source: Secure Coding by Graph & van Wyk
Business continuity

- One of the key goals of security is to support business processes.
- Can be done by analysing the business process itself.
- More commonly done by analysing the assets that support the business process.
- The value of the assets is then proportional to how crucial they are to the business.
- Business continuity extends into design through mitigation strategies.
What is needed for secure design?

- Up-to-date knowledge of security threats and countermeasures
- Good software engineering practice
- Risk assessment on which to base security decisions
- Security mechanisms that are usable by, and acceptable to, the intended users
Problem

- Gathering security requirements is not easy - need multiple sources of expertise
- Developers need a process for building secure, functional, usable, etc. systems
- Developers need a *usable* secure development process
There are many approaches to security
- Interpretivism, functionalism, checklists, risk analysis, standards...
- Most do not address all the issues raised here
- AEGIS is one methodology designed to attempt to reconcile these issues
- Two main parts to AEGIS:
  - Requirements phase
  - Design phase
Appropriate and Effective Guidance for Information Security

Bridge functionalist & interpretive approaches

Aim:
- Simplify secure design process
- Sound security design basis
- Integrate with existing software design techniques
  - risk analysis and choice of countermeasures
  - requirements capture and specification document
  - context of use
- Decision-making integrating user needs
Contextual Design

- Method aimed at basing design decisions on user context
- Identify and model as much as possible about the user and his environment
- Design is an ultimately creative discipline, but should be grounded in the data gathered about the context of the user
Gather Participants
Gather Participants

- First step is to secure the right stakeholders to participate in AEGIS development process
- Owners
- Developers
- Users
- Field Experts

Important in order to gather the most pertinent information. This has proven helpful in improving communications and clarifying the security role and expectations of people within the projects.
Identify Assets

- People
  - Users
  - Administrators
  - Developers

- Hardware
  - Network
  - Processing nodes
  - Storage

- Data
  - Applications
  - Information

- Security Measures
Gather Participants

Identify Assets

Model System Assets and Context
Asset Modelling in Context

- High-level view of system in UML compatible notation
- Re-create context
  - Physical i.e. rooms, organisations
  - Cultural i.e. job/responsibility, security culture
- Represent all major assets (include different people and their tasks)

This has proven useful in explicitly stating implicit assumptions made about people and their tasks in the system. It is also the basis on which security requirements can be elicited and captured.
Gather Participants

Identify Assets

Model System Assets and Context

Value Assets according to security properties
Asset valuation

- Assess value of assets according to:
  - Confidentiality
  - Integrity
  - Availability
  - Other properties (e.g. dependability, accountability, non-repudiation)
- Rate them intuitively and hierarchically
  - i.e. low, medium, high, essential, etc.
  - Identifies the most important assets
  - Can use natural language
Modelling Notation

- AEGIS uses a specific modelling notation to capture:
  - Assets
  - Context
  - Security Requirements
- Simple notation designed to integrate with UML
Simple Example

Main Office

Accountant

Secretary

Workstation

Salary Database
Confidentiality: High
Integrity: High
Availability: Low

Word Processor
Confidentiality: Low
Integrity: Low
Availability: High

Internet
Gather Participants

Identify Assets

Model System Assets and Context

Value Assets according to security properties

Identify Threats

Identify Vulnerabilities

Identify Risks
Threat/Risk Analysis

- Threats are potential sources of attacks
- Vulnerabilities are areas of the system that are amenable to exploitation
- Risk is the likelihood of a threat successfully exploiting one or a sequence of vulnerabilities
- Gather information from:
  - Expert advice
  - Competition (if possible)
  - Participants’ past experience
## Example Threats

<table>
<thead>
<tr>
<th>Threat</th>
<th>ScriptKiddie_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker</td>
<td>Script Kiddie</td>
</tr>
<tr>
<td>Motive</td>
<td>Fun</td>
</tr>
<tr>
<td>Target</td>
<td>Salary database</td>
</tr>
<tr>
<td>Resources</td>
<td>Low – found an attack script on the internet, single workstation, not much time to devote to breaking in.</td>
</tr>
<tr>
<td>Risk-Aversion</td>
<td>High – does not want to be caught or prosecuted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Threat</th>
<th>Employee_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker</td>
<td>Irate/Disgruntled Employee</td>
</tr>
<tr>
<td>Motive</td>
<td>Spite</td>
</tr>
<tr>
<td>Target</td>
<td>Salary database</td>
</tr>
<tr>
<td>Resources</td>
<td>High – has physical access to the workstation, is prepared to go to some lengths to find out about/change/delete the salary database in order to cause trouble.</td>
</tr>
<tr>
<td>Risk-Aversion</td>
<td>Low – does not care about being caught, as he is happy to leave.</td>
</tr>
<tr>
<td>Vulnerability</td>
<td>Buffer Overflow in Windows Workstation Service</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Asset affected</td>
<td>Workstation running Microsoft Windows 2000 with Internet access</td>
</tr>
<tr>
<td>Dependent assets</td>
<td>Salary database, word processor</td>
</tr>
<tr>
<td>Impact</td>
<td>Remote attacker can execute arbitrary code on the workstation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Social engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset affected</td>
<td>Accountant, Secretary</td>
</tr>
<tr>
<td>Dependent assets</td>
<td>Salary database, word processor, workstation, main office, security measures</td>
</tr>
<tr>
<td>Impact</td>
<td>Can circumvent/compromise security measures to provide operator level access to a third party. Can disclose information about salary database.</td>
</tr>
</tbody>
</table>
## Example Risk

<table>
<thead>
<tr>
<th>Risk Priority</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat</td>
<td>Script Kiddie_1</td>
</tr>
<tr>
<td>Vulnerability</td>
<td>Buffer Overflow in Windows Workstation Service, Internet connected machine</td>
</tr>
<tr>
<td>Likelihood</td>
<td>Medium</td>
</tr>
<tr>
<td>Impact</td>
<td>Very High: The confidentiality, availability and integrity of all assets located on the workstation stand to be compromised</td>
</tr>
<tr>
<td>Justification</td>
<td>Attack tool is readily available on the internet and the database resides on a machine that is directly connected to the internet</td>
</tr>
</tbody>
</table>
Gather Participants

Identify Assets

Model System Assets and Context

Value Assets according to security properties

Identify Threats

Identify Vulnerabilities

Identify Risks

Design Countermeasures

Assess Cost of Countermeasure in context

Assess Benefit of Countermeasure in context

Cost too high, Benefit too low

Cost, benefit and residual risk are acceptable
Cost-benefit design

- Security design using:
  - Asset model
  - Requirements specification
  - Risk analysis

- Evaluate Cost and Benefit:
  - Financial cost
  - User cost in context
  - Effectiveness of countermeasure
  - Residual risk
Case Studies

- AEGIS has been applied to a number of research projects:
  - EGSO, eDiaMond, CLEF, BioSimGrid, climateprediction.net, DCOCE

- Grid applications
  - Are distributed computation platforms
  - Unknown security needs
  - Novel area and requirements
  - Academic and Industrial involvement
General Findings

- Development environment is distributed
  - Highlights the importance of clear communication

- Scenarios
  - Means of reasoning about security
  - Means of communicating security concepts
  - Useful in eliciting security needs

- Anecdotes
  - Frequently used as source of security knowledge
General Findings

- **Clear responsibility is key to strong security motivation**
  - Legal/ethical requirements -> clear responsibility
  - No responsibility -> motivation can be a problem
  - Reputation & trust appear to be significant motivators for security

- **Diffusion of responsibility**
  - Perception that no control -> no responsibility
  - Lack of communication between stakeholders
Many projects have no one in charge of security

**Diffusion of responsibility**
- Many people assume that security will be taken care of
- Little coordination between the project and the institutions that will run the project regarding their security requirements.

**Responsibility for security is not generally with implementors of security**

**Perceived boundaries of responsibility for security affect motivation to apply security and need for usability.**
Motivation for security varies
+ Need to be trusted (by customers)
+ Legal requirement (medical grids)
- Functionality priority
- Pressure from users:
  • Don’t understand the need for software engineering methods
  • Want prototypes as fast as possible
Details – Knowledge

- **Knowledge of security:**
  - Focus mainly on confidentiality - availability and integrity not really considered (especially medical grids)
  - Incorrect assumptions made of some security technologies (Middleware will take care of PKI)
  - Preventative attitude, little though given to detection, reaction, deterrence, etc.
  - Security can be very confusing
    - Terminology is very precise
    - Need for a moderator in the process
    - Resolving disagreements is an open problem

- **Knowledge of system**
  - Participants usually learn more about the system in discussions
During development human actions are “automated”, i.e. assumed to function in the same way as the rest of the system.

Security controls require specific human behaviour that is frequently assumed to be taken care of (backup, key management, monitoring, etc.).

Need for usability apparent for areas of loss of direct control:
- Want low buy-in from (paying) customers (cannot enforce security)
- Need users to behave securely
- Administrators can be vetted and hired to maintain the more complex security – but this requires organisational practice
Summary

- Differences in common security design approaches
- What is important in security design
- AEGIS
- Outcomes from practical application