SECURE BY DESIGN

Security Design Principles for the Working Architect

Eoin Woods
Endava
@eoinwoodz
• Eoin Woods
  • **CTO at Endava** (technology services, ~5000 people)
  • **10 years in product development** - Bull, Sybase, InterTrust
  • **10 years in capital markets applications** - UBS and BGI
  • Software dev engineer, then architect, now CTO

• Author, editor, speaker, community guy
• **What is security** and why do we care?

• What are **security principles**, why are they **useful**?

• **Security design principles**
  • 10 important principles useful in practice

• **Improving application security** in real teams
REVISITING SECURITY
We all know security is important - but why?
- protection against malice, mistakes and mischance
- theft, fraud, destruction, disruption

Security is a risk management business
- loss of time, money, privacy, reputation, advantage
- insurance model - balance costs against risk of loss
ASPECTS OF SECURITY PRACTICE

- Secure Application Design
- Secure Application Implementation
- Secure Infrastructure Design
- Secure Infrastructure Deployment
- Secure System Operation
DATA BREACHES 2005 - 2007

http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/
System **interfaces on the Internet**

Introspection of **APIs**

Attacks being **“weaponised”**

Today’s **internal app** is tomorrow’s **“digital channel”**

https://cybermap.kaspersky.com/
SECURITY PRINCIPLES
What is a “principle”?  

a fundamental **truth or proposition** serving as the foundation for **belief or action** [OED]

We define a **security design principle** as ….

a declarative **statement** made with the intention of **guiding security design decisions** in order to meet the goals of a system.
SECURITY DESIGN PRINCIPLES

- There are many sets of security design principles
  - Viega & McGraw (10), OWASP (10), NIST (33), NCSC (44), Cliff Berg (185) …
  - Many similarities between them at fundamental level

- I have distilled 10 key principles as a basic set
  - these are brief summaries for slide presentation
  - www.viewpoints-and-perspectives.info
A SYSTEM TO BE SECURED
10 KEY SECURITY PRINCIPLES
TEN KEY SECURITY PRINCIPLES

- Assign the **least privilege** possible
- Separate **responsibilities**
- **Trust cautiously**
- **Simplest** solution possible
- **Audit** sensitive events
- **Fail securely** & use **secure defaults**
- Never rely upon **obscurity**
- Implement **defence in depth**
- **Never invent** security technology
- Find the **weakest link**
<table>
<thead>
<tr>
<th><strong>Why?</strong></th>
<th>Broad privileges allow malicious or accidental access to protected resources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principle</strong></td>
<td>Limit privileges to the minimum for the context</td>
</tr>
<tr>
<td><strong>Tradeoff</strong></td>
<td>Less convenient; less efficient; more complexity</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Run server processes as their own users with exactly the set of privileges they require</td>
</tr>
</tbody>
</table>
## 2 - SEPARATE RESPONSIBILITIES

<table>
<thead>
<tr>
<th>Why?</th>
<th>Achieve control and accountability, limit the impact of successful attacks, make attacks less attractive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principle</strong></td>
<td>Separate and compartmentalise responsibilities and privileges</td>
</tr>
<tr>
<td><strong>Tradeoff</strong></td>
<td>Development and testing costs; operational complexity: troubleshooting more difficult</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>“Payments” module administrators have no access to or control over “Orders” module features</td>
</tr>
</tbody>
</table>
2 - SEPARATE RESPONSIBILITIES
### 3- TRUST CAUTIOUSLY

<table>
<thead>
<tr>
<th>Why?</th>
<th>Many security problems caused by inserting malicious intermediaries in communication paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principle</td>
<td>Assume unknown entities are untrusted, have a clear process to establish trust, validate who is connecting</td>
</tr>
<tr>
<td>Tradeoff</td>
<td>Operational complexity (particularly failure recovery); reliability; some development overhead</td>
</tr>
<tr>
<td>Example</td>
<td>Don't accept untrusted RMI connections, use client certificates, credentials or network controls</td>
</tr>
</tbody>
</table>
### Why?
Many security problems caused by inserting malicious intermediaries in communication paths.

### Principle
Assume unknown entities are untrusted, have a clear process to establish trust, validate who is connecting.

### Tradeoff
Operational complexity (particularly failure recovery), reliability; some development overhead.

### Example
Don't accept untrusted RMI connections, use client certificates, credentials or network controls.
<table>
<thead>
<tr>
<th><strong>Why?</strong></th>
<th>Many security problems caused by inserting malicious intermediaries in communication paths</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principle</strong></td>
<td>Assume unknown entities are untrusted, have a clear process to establish trust, validate who is connecting</td>
</tr>
<tr>
<td><strong>Tradeoff</strong></td>
<td>Operational complexity (particularly failure recovery); reliability; some development overhead</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Reject untrusted RPC connections, authenticate clients, check 3rd party components, scan your open source</td>
</tr>
</tbody>
</table>
3 - TRUST CAUTIOUSLY

https://www.aspectsecurity.com/research-presentations/the-unfortunate-reality-of-insecure-libraries
3 - TRUST CAUTIOUSLY

Average Days Before Vulnerability is Exploited

Sources: Gartner, IBM, Sonatype

Sonatype 2018 State of the Software Supply Chain Report
3 - TRUST CAUTIOUSLY

Who are you? How do we know?

What is connecting to our services?

What libraries do we use? From where?

What are we connecting to?

What can access our database?
<table>
<thead>
<tr>
<th>Why?</th>
<th>Security requires understanding of the design - complexity rarely understood - simplicity allows analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principle</td>
<td>Actively design for simplicity - avoid complex failure modes, implicit behaviour, unnecessary features, ...</td>
</tr>
<tr>
<td>Tradeoff</td>
<td>Hard decisions on features and sophistication; Needs serious design effort to be simple</td>
</tr>
<tr>
<td>Example</td>
<td>Does the system really need dynamic runtime configuration via a custom DSL?</td>
</tr>
</tbody>
</table>

The **price** of **reliability** is the pursuit of the utmost **simplicity** - C.A.R. Hoare
## 5 - AUDIT SENSITIVE EVENTS

<table>
<thead>
<tr>
<th>Why?</th>
<th>Provide record of activity, deter wrong doing, provide a log to reconstruct the past, provide a monitoring point</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principle</strong></td>
<td>Record all security significant events in a tamper-resistant store</td>
</tr>
<tr>
<td><strong>Tradeoff</strong></td>
<td>Performance; operational complexity; dev cost</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Record changes to &quot;core&quot; business entities in an append-only store with (user, ip, timestamp, entity, event)</td>
</tr>
</tbody>
</table>
5 - AUDIT SENSITIVE EVENTS
| Why?          | Default passwords, ports & rules are “open doors”  
|              | Failure and restart states often default to “insecure” |
| Principle    | Force changes to security sensitive parameters  
|              | Think through failures - to be secure but recoverable |
| Tradeoff     | Convenience |
| Example      | Don’t allow “SYSTEM/MANAGER” logins after installation  
|              | On failure don’t disable or reset security controls |
## 7 - NEVER RELY ON OBSCURITY

<table>
<thead>
<tr>
<th><strong>Why?</strong></th>
<th>Hiding things is difficult - someone is going to find them, accidentally if not on purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principle</strong></td>
<td>Assume attacker with perfect knowledge, this forces secure system design</td>
</tr>
<tr>
<td><strong>Tradeoff</strong></td>
<td>Designing a truly secure system takes time and effort</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Assume an attacker will guess a &quot;port knock&quot; network request sequence or a password obfuscation technique</td>
</tr>
</tbody>
</table>
## 8 - DEFENCE IN DEPTH

<table>
<thead>
<tr>
<th><strong>Why?</strong></th>
<th>Systems do get attacked, breaches do happen, mistakes are made - need to minimise impact</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principle</strong></td>
<td>Don’t rely on single point of security, secure every level, stop failures at one level propagating</td>
</tr>
<tr>
<td><strong>Tradeoff</strong></td>
<td>Redundancy of policy; complex permissioning and troubleshooting; can make recovery difficult</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Access control in UI, services, database, OS</td>
</tr>
</tbody>
</table>
8 - DEFENCE IN DEPTH
9 - NEVER INVENT SECURITY TECH

<table>
<thead>
<tr>
<th>Why?</th>
<th>Security technology is difficult to create - avoiding vulnerabilities is difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principle</td>
<td>Don’t create your own security technology - always use a proven component</td>
</tr>
<tr>
<td>Tradeoff</td>
<td>Time to assess security technology; effort to learn it; complexity</td>
</tr>
<tr>
<td>Example</td>
<td>Don’t invent your own SSO mechanism, secret storage or crypto libraries … choose proven components</td>
</tr>
</tbody>
</table>
9 - NEVER INVENT SECURITY TECHNOLOGY
9 - NEVER INVENT SECURITY TECHNOLOGY
## 10 - Secure the Weakest Link

<table>
<thead>
<tr>
<th>Why?</th>
<th>&quot;Paper Wall&quot; problem - common when focus is on technologies not threats</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principle</strong></td>
<td>Find the weakest link in the security chain and strengthen it - repeat! (Threat modelling)</td>
</tr>
<tr>
<td><strong>Tradeoff</strong></td>
<td>Significant effort required; often reveals problems at the least convenient moment!</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Data privacy threat =&gt; encrypted communication but with unencrypted database storage and backups</td>
</tr>
</tbody>
</table>
TEN KEY SECURITY PRINCIPLES

• Assign the **least privilege** possible
• Separate **responsibilities**
• **Trust cautiously**
• **Simplest** solution possible
• **Audit** sensitive events

• **Fail securely** & use secure defaults
• Never rely upon **obscurity**
• Implement **defence in depth**
• **Never invent** security technology
• Find the **weakest link**
SECURITY IN REAL TEAMS
SOME COMMON CONCERNS

Where do we start?

Who is involved?

What tools do we use?

Can we do this with agile?

Won't this slow everything down?

Will this cost a lot?
SOME OBSERVATIONS

• Some **individuals** will find it **fascinating**, some will **hate** it

• Teams will need **guidance and inspiration**

• Teams need to **own their security process**
  • But a clearly defined **starting point** and **standards** very valuable

• A clear **roadmap** helps to avoid overload
SOME USEFUL TACTICS

• Form a group of **security champions** - invest in them
  • involve many roles (BA, developer, tester, architect, …)

• **Communicate importance** of security from the top
  • and from the customer

• Make the **right thing the easy thing**
  • checklists and templates, clear guidance, packaged tools

• Be prepared for the process to **take time**
USUALLY A GRADUAL PROCESS

- EXPERT APPLICATION SECURITY TEAM
- COMPETENT APPLICATION SECURITY TEAM
- INFORMED APPLICATION SECURITY TEAM
- SECURITY AWARE TEAM
- NO SECURITY PRACTICE
EXAMPLE CAPABILITY PLAN

- **EXPERT**
  - Active Threat Assessment
  - Attck Surface Analysis
  - Dynamic Analysis
  - Fuzz Testing
  - Red Teams
  - Continual Improvement

- **COMPETENT**
  - Threat Modelling
  - Sec Code Reviews
  - Secure Design
  - Incident Simulations
  - Risk Assessment
  - OSS Mgmt
  - Basic Secure Design

- **INFORMED**
  - Security Requirements
  - Release Criteria
  - Secure Coding
  - Static Scanning

- **AWARE**
  - Security Principles
  - OWASP “Top 10”
  - Basic Sec Coding
  - Pen Testing

**Security Principles**
OWASP SAMM

SAMM Overview

Software Development

Business Functions

Governance
Construction
Verification
Deployment

Security Practices

Strategy & Metrics
Education & Guidance
Security Requirements
Design Review
Security Testing
Environment Hardening

Policy & Compliance
Threat Assessment
Secure Architecture
Code Review
Vulnerability Management
Operational Enablement

http://www.opensamm.org
MICROSOFT SDL

Training
- Establish Security Requirements
- Create Quality Gates / Bug Bars
- Security & Privacy Risk Assessment

Core Security Training
- Establish Design Requirements
- Analyze Attack Surface
- Threat Modeling

Requirements
- Use Approved Tools
- Deprecate Unsafe Functions
- Static Analysis

Design
- Dynamic Analysis
- Fuzz Testing
- Attack Surface Review

Implementation
- Incident Response Plan
- Final Security Review
- Release Archive

Verification
- Execute Incident Response Plan

Release
- Response

https://www.microsoft.com/en-us/sdl/
TO RECAP …
TEN KEY SECURITY PRINCIPLES

- Assign the **least privilege** possible
- Separate **responsibilities**
- **Trust cautiously**
- **Simplest** solution possible
- **Audit** sensitive events
- **Fail securely** & use secure defaults
- Never rely upon **obscurity**
- Implement **defence in depth**
- **Never invent** security technology
- Find the **weakest link**
Continuous Process

Towards Secure SDLC
REFERENCES

• UK Government NCSC Security Principles:
  https://www.ncsc.gov.uk/guidance/security-design-principles-digital-services-main

• NIST Engineering Principles for IT Security:

• Short intro to McGraw’s set:
  http://www.zdnet.com/article/gary-mcgraw-10-steps-to-secure-software/

• OWASP Principles set:
  https://www.owasp.org/index.php/Category:Principle
BOOKS
THANK YOU

QUESTIONS?

Eoin Woods
Endava
eoin.woods@endava.com
@eoinwoodz