How to Improve Software Security, Now is the time to change the Approaching

Special thanks to
Andrew Cushman, Sr. Director Microsoft TWC Security, and
Mike Convertino, Sr. Director Microsoft network security for the Cyber Terrain Concept

Jason Lee
Representative / Asia
Immunity, Inc.
Because we do security,

We are COMPANY!!
Approaching to build Secure Applications (but you’d never concern…)  
SOMETHING YOU KNEW ABOUT
Worst-case Scenarios: Weaken Software Security

What can be Happened?

- Sensitive Data Leakage
  - Customers (privacy), Business Partners or Company Data
- Identity Thefts
  - Bad guy can impersonating as trusted user
- Defacement – Content Modification
  - Damage to Brand, Mislead Customers etc
- Application Shutdown (Site Unavailable)
  - Lack of Access can cause major loses
## From IT, Application Security: 3rd Parties & Customized Applications

<table>
<thead>
<tr>
<th></th>
<th>Infrastructure Vulnerabilities</th>
<th>Application Specific Vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause of Defect</strong></td>
<td>Insecure development or deployment of 3rd party SW</td>
<td>Insecure development of <strong>your own applications</strong></td>
</tr>
<tr>
<td><strong>Location of Vulnerability</strong></td>
<td>3rd party <strong>infrastructure</strong> (web server, OS, etc.)</td>
<td><strong>Application Code</strong>, often resides on Application Server</td>
</tr>
<tr>
<td><strong>Method of Exploits</strong></td>
<td>Known vulnerabilities</td>
<td>Probing hacks, suspicious content, information leakage</td>
</tr>
<tr>
<td><strong>Detection</strong></td>
<td>Patch Management system</td>
<td>Application Security Scanners Source Code Analysis tool</td>
</tr>
<tr>
<td></td>
<td>Internal/External Audits, Automated Scanners</td>
<td></td>
</tr>
<tr>
<td><strong>What to do</strong></td>
<td>Update patches, use trusted 3rd party software</td>
<td>Training, Scanners &amp; Fix – across the Development Life Cycle</td>
</tr>
</tbody>
</table>
Why Software Security issues happen

• **THE FASTEST-GROWING TYPE OF HACK ATTACKS TODAY**

• **“IMPERFECT” DEVELOPMENT**
  - Application developers usually not trained and/or don’t’ care about security
  - IT security people are usually from the network/infrastructure side, have little experience about programming

• **“TRADITIONAL” SECURITY SOLUTIONS DO NOT PROTECT AGAINST Software ATTACKS**
  - Firewalls, IDS/IPS (ISS), Anti-Virus, Anti-Spam, etc are all network/infrastructure (IP Traffic) security solutions
  - Software and Application attacks are HTTP or SOAP traffic or other traffic and protocol – totally different

• **CHANGING TREND OF HACK ATTACKS TODAY**
  - Hackers today are focusing attacks on Software and applications to get in or through Software and application to approach
  - Most of security spending and assets are network/infra-focused.
Why Software Security is Continuously required

- Most biggest vulnerability in Network Environment of Company
  - Vast-range accessibility to the Application existing in Network
- Application Vulnerabilities
  - Defects or flaw in Web Applications
  - It may be a cause of unexpected results or malicious controls
- Using Web based technologies could simplify the behavior of attack itself
- Learning and using New technologies
- Application has been constantly updated and renewal

“Service using Applications through Internet may be a Major Vulnerability and Hole of Networks and Systems in Organizations.”
Why Integrating Security testing into SDLC

• Managing Security flaws in stage-by-stage may help to:
  • Reduce High Work Load Centered to Specific Team *(but what happened now?)*
  • Have a Control, Visibility and Metric to Threats through Keep Managing up *(Is it truly happened?)*
  • Find defects and flaws as early as possible during the SDLC *(Is it really worked on developer side?)*
  • Propose and determine most cost effective solution *(Cost Effective? Or Cheapest?)*
  • Give better protect to own assets

• Knowing and Understanding Security Threats and Vulnerabilities to a Applications can help to secure and maintain Services:
  • Since stakeholders can determine what security mechanism or technology would be an effective defender to protect floated vulnerabilities *(and they decide to have….)*
  • To solve issues, **better to use general countermeasure** than specific technology or solution.
  • Being aware of solved issues and issues in live can help to prevent and expect possible breakdowns or attacks as best as we can.
SDL Implementation & Key Stakeholder for Secure Application

Software Lifecycle Stakeholders

Will bring you the answer why key stakeholders of appl' should concern security

EXPLOITATION ECONOMICS
Machines Fight Back

- Machines do exactly what they are programmed to do
- Machines follow instructions – instructions are getting better…
- Defenders are ever better at telling….
- John Lambert - ROI of Exploit Economics
- How much cost find

The Golden Age of Exploit

- Pre-XP SP2
- Pre-Auth Remote ware all the rage
- Four Byte overwrites – Worked every time
- XPSP2 – August 2004
- XPSP2 – firewall on by default, authenticated endpoints
- BlueHat v2 – March 2005
Microsoft believes that delivering secure software

Executive commitment -> SDL a mandatory policy at Microsoft

- Core Training
  - Analyze Security and privacy risk
  - Define quality gates

- Requirement
  - Threat modeling
  - Attack surface analysis

- Implementation
  - Specify tools
  - Enforce banned functions
  - Dynamic/Fuzz testing
  - Verify threat models/attack surface

- Release
  - Response plan
  - Final security review
  - Release archive

- Response execution

Education -> Process -> Accountability

Ongoing Process Improvements
Software & Security Asymmetry Turned Upside-Down

- Ever increasing R&D needed for stealthy == 11 exploits
- Attackers face new challenges – margins on re-use of exploits go to the operator, or fast follower, not the author
- Lots of competition

Think about APT, how they re-use exploits effectively!!
Questions for Pwn2Own hacker Charlie Miller

By Ryan Naraine for Zero Day | March 19, 2009 -- 14:50 GMT (07:50 PDT)

Why Safari? Why didn’t you go after IE or Safari?

It’s really simple. Safari on the Mac is easier to exploit. The things that Windows do to make it harder (for an exploit to work), Macs don’t do. Hacking into Macs is so much easier. You don’t have to jump through hoops and deal with all the anti-exploit mitigations you’d find in Windows.

It’s more about the operating system than the (target) program. Firefox on Mac is pretty easy too. The underlying OS doesn’t have anti-exploit stuff built into it.

On a scale of 1-10, how impressive was the Nils’ sweep of exploiting all three main browsers?

I was surprised. For IE 8, I’d give him a 9 out of 10. For Safari, maybe a 2. It’s just too easy to pop Safari. For Firefox on Windows, I give him a 10. That was the most impressive of the three. It’s really hard to exploit Firefox on Windows.

Nils2Own: 'I want to see security flaws fixed'


Let me correct something. It was a Firefox on Mac OS X vulnerability and exploit. The bug does affect Windows but, honestly, it’s way harder to get the code to run reliably on Windows. That’s the reason I did my Firefox attack on the Mac. I’m not allowed to talk about it but, for that bug, to get real exploitation on Windows is difficult because of ASLR (Address Space Layout Randomization) and DEP (Data Execution Prevention). On the Mac, I could trigger it and exploit it easily.

IE 8 on Windows 7?

I came here with that vulnerability. It’s another nice bug but it was really, really difficult to write the exploit because of those ASLR and DEP. I had to use some techniques around those mitigations and make a lot of preparation to make it a reliable exploit. It was very, very hard.
Baron Daniel Crowley
@dan_crowley

Windows 8 memory corruption exploitation is now incredibly difficult. In other news: 'union all select cc_num from cards;--

Chaouki Bekrar VUPEN
@cBekrar

Win8 will make exploit dev harder but at the same time it will make exploit prices higher. At Win9, researchers will be billionaires

Tarjei Mandt
@kernelpool

Kernel pool quota pointer attack no longer works on Windows 8. Process pointer is XOR'ed with a random cookie (nt!ExpPoolQuotaCookie).
The Software Vulnerability Asymmetry Problem

• Defender must fix all vulnerabilities in all software, but attacker wins by finding and exploiting just one vulnerability

• Threat change over time – state-of-the-art in vulnerability finding and attack technique changes over time.

• Patch deployment takes time – vendor must offset risks to stability & compatibility, customer waits for servicing cycle

Result: Attackers only have to find one vulnerability, and they get to use it for a really long time.
Exploit Economics

\[ \text{ROI} = \frac{\text{Gain from Investment} - \text{Cost of Investment}}{\text{Cost of Investment}} \]

\[ \text{Attacker ROI} = \frac{\text{Attacker Gain} - \text{Attacker Cost}}{\text{Attacker Cost}} \]

\[ \text{Attacker Gain} = \frac{\text{Gain}}{\text{Opportunity}} \times \text{N Opportunities} \]

\[ \text{Attacker Cost} = \text{Vulnerability Cost} + \text{Exploitation Cost} \]

\[ \text{Attacker ROI} = \frac{\left( \frac{\text{Gain}}{\text{Opportunity}} \times \text{N Opportunities} \right) - \left( \text{Vulnerability Cost} + \text{Exploitation Cost} \right)}{\left( \text{Vulnerability Cost} + \text{Exploitation Cost} \right)} \]
Exploit Economics

\[
\text{Attacker ROI} = \left( \frac{\text{Gain}}{\text{Opportunity}} \times N \text{ Opportunities} \right) - (\text{Vulnerability Cost} + \text{Exploitation Cost}) \\
\frac{\text{Vulnerability Cost} + \text{Exploitation Cost}}{\text{Vulnerability Cost} + \text{Exploitation Cost}}
\]

- We can decrease Attacker ROI if we are able to...
  - **Increased attacker investment** – increased cost to find usable vulnerabilities
    - Varies by platform and vendor and technology
    - New tools and automation help w/bug mining, but on some platforms the watermelons are already harvested
  - Increased attacker investment required to write reliable (and stealthy) exploits
    - Exploit vulnerability and breakout of sandbox / defeat additional protections and mitigations
    - Boutique bespoke software development house w / ever expanding requirements
  - Decreased attacker opportunity to recover investment
    - Fewer opportunities via artificial diversity & improved updating
    - Ever improving detection of exploits & follow on actions
    - Fewer resale ? Reuse opportunities

Result: Stealthy, reliable attacks require significant engineering; working exploits become more scarce and valuable and shorter lived(?)
Exploit Economics

- Maturing Industry – Specialized & horizontal

  - Also now vertically reintegrated at state level
  - Squeezed from the bottom
    - $500 PC with IDA Pro & BinDiff
  - Squeezed from the top
    - Ever expanding list of cyber capable countries
    - $500M investment returns Tier1 capability
Countries w/ Cyber “Organization”

- **33 states include cyber-warfare in the world-wide planning and organization**
  - Range of Capabilities and Approaches
  - E.g., Argentina, Brazil, Canada, China, Denmark, Germany, India, Iran, Israel, North Korea, South Korea, Switzerland and the United States (now Japan want to have Cyber army…)

- **36 states w/ civilian agencies charged with internal security missions, computer security or law enforcement**
  - “Traditional”(1990s) approach to cyber-security – National CERT, special departments

What’s your Future

- Low skill exploits never go out of style
  - Expanding surface area & victim pool
- High Skill Exploits – barrier to entry growing
  - Sole proprietorship bespoke exploit house a dying breed (?)
- To be continued….
Defenders Ethos

- **Increase attacker investment** required to find usable vulnerabilities
  - Remove entire classes of vulnerabilities where possible
  - Focus on automation to scale human efforts

- **Increase attacker investment** required to write reliable exploits
  - Build mitigations that add brittleness
  - Make exploits impossible to write completely reliably

- **Decrease attacker’s opportunity to recover their investment**
  - Shrink windows of vulnerability
  - Work on rapid detection & suppression of exploit usage

- **Engineer for Scale**
  - Design & build solutions for billions of machines

- **Copy what works & keep plugging away**
  - Long term approach, incremental and continual progress
  - Apply best practices across domains e.g., “ASLR” for networks & web apps
Typical Defense in Depth approaching

IBM AppScan Source Edition ™

OSI Protocol Stack

1. Physical Layer
   - Fiber/Copper/Serial

2. Data Link Layer
   - Network Interface Device

3. Network Layer
   - IP, ICMP, RIP, IGMP, ARP

4. Transport Layer
   - TCP, UDP, SSL

5. Session Layer
   - NETBIOS, NFS, SQL, RPC

6. Presentation Layer
   - ASCII, JPEG, MPEG

7. Application Layer
   - DNS, Telnet, TFTP, HTTP, SMTP

Network (TCP/IP) Sack

1. PHYSICAL LAYER
   - ETHERNET, SWITCH

2. NETWORK INTERFACE CARD
   - MACHINE LANGUAGE 11001101110

3. NETWORK
   - HARDWARE CPU, GPU

4. TRANSPORT
   - MACHINE LANGUAGE 11001101110

5. SESSION
   - MACHINE LANGUAGE 11001101110

6. PRESENTATION
   - MACHINE LANGUAGE 11001101110

7. APPLICATION
   - MACHINE LANGUAGE 11001101110

Computing Stack

Security

- IDS/IPS & Firewalls
- Mail Guards
- Host Based Security System
- Anti-Virus System
- Database Security

Software Application

- Machine Language 11001101110

Platform & Application

Operations System (Windows, Unix)

Machine Language 11001101110

Operating System

- CPU, GPU

Platform & Application

- Operating System

- Operating System

- Operating System

- Operating System

- Operating System
Terrain matters on the ground

- Defense in Depth at the gateway is the right place to enforce policy control
  - Sees all traffic
  - Defines trust boundary

- BUT… Applications Have Changed
  - Ports ≠ Applications
  - IP Addresses ≠ Users
  - Packets ≠ Content

- HTTP == UFPP
  - Universal Firewall Penetrating Protocol
Manic or Visionary

- Defenses Improve – Machines do what they’re told 100% of time
  - Secrets and Assets can be protected
  - Attacks can be detected and thwarted

- Exploitation Direction
  - Highest value targets require ever increasing R&D investment
  - Ever increasing R&D w/ smaller margins & low units
  - No shortage of vulnerable targets, but…
  - Exploit economics worsen for the majority

- Back to the Future (?)
  - Create Social Engineering & Tradecraft Departments
How to change the approaching....

NOW YOU CAN SEE
SDL Implementation & Key Stakeholder for Secure Application

Software Lifecycle Stakeholders

- Top Management
- Business Unit Head
- IT Manager
- Security Specialist
- Application Owner
- Developers & Coders
- Project Managers
- Team Leads
- Quality Assurance Manager
- Business Analysts
- Technical Architects
- Auditors
- Client Side PM
- Industry Group Delivery Heads

Back to: Why Integrating Security testing into SDLC

- Managing Security flaws in **stage-by-stage** may help to:
  - Reduce High Work Load Centered to **Specific Team** *(you know who Centered Specific Team was, then who will be the team share the testing workload?)*
  - Have a **Control, Visibility and Metric to Threats** through Keep Managing up *(to happen, how where and who should do testing?)*
  - Find defects and flaws **as early as possible** during the SDLC *(who should have majority of test?)*
  - Propose and determine **most cost effective solution** *(did not mean cheaper, it means most cost effective! And time either.. Why not use secure coding consulting to learn practices)*
  - Give better protect to own assets

- Knowing and Understanding Security Threats and Vulnerabilities to a Web Applications can help to secure and maintain Services:
  - Since **stakeholders** can determine what security mechanism or technology would be a effective defender to protect floated vulnerabilities *(they have to know about exploitation economics and importance to fix defects)*
  - To solve issues, **better to use general countermeasure** than specific technology or solution.(*re you familiar with secure library?*)
  - Being aware of solved issues and issues in live can help to prevent and expect possible breakdowns or attacks as best as we can.
What you really need to know: Core Defense Mechanisms

- The Fundamental Security Problem with web applications
  - Rise to a number of security mechanisms that applications use to defend themselves against attack.
- Virtually all applications employ mechanisms that are conceptually similar.
- Core elements of Web Applications’ defense mechanisms:
  - **Handling user access** to the application’s data and functionality
    - Prevent user from gaining unauthorized access
  - **Handling user input** to the application’s function
    - Prevent malformed input from causing undesirable behavior
  - **Handling attackers**
    - Ensure that the application behaves appropriately when being directly targeted, taking suitable defensive and offensive measures to frustrate the attacker
  - **Managing the application itself**, by enabling administrators to monitor its activities and configure its functionality
Core Defense Mechanisms

- Handling User Accessing
  - Authentication
  - Session Management
  - Access Control

- Handling User Input
  - Various Input
  - Approaching Method to Handle Input
    - Reject Known Bad
    - Accept Known Good
    - Sanitization
    - Secure Data Processing
    - Semantic Check
  - Boundary Validation
  - Multi Step Verification and Canonicalization

- Handling Attacker
  - Error Handling
  - Maintain Audit Log
  - Alarm
  - Response for Attack

- Application Management
  - User Account
  - User Role
  - Monitoring and Auditing
  - Analyzing
  - Control Application Functionality side
# SDLC with Enabling Security Testing

## Requirement
- Define Security Requirements (System & Functionalities)
- Define Secure Coding Standard
- Define Secure Frameworks
- Mapping Security Requirements

## Design
- Review Security Design
- Define Security Lists
- Define Security Test for Develop. and QA
- Threat Modeling
- Security Design

## Coding
- Review Security Code
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing

## Integration
- Web Application Security Testing using Automated Black Box Testing
- Tracking Security Issues
- Periodical Testing and Meeting for Security Issues between QA/Support Teams
- Security Update, Patch Test, Patch Deploy

## QA
- Define Security Requirements (System & Functionalities)
- Define Secure Coding Standard
- Define Secure Frameworks
- Mapping Security Requirements

## Release
- Functional Lists
- Quality Guide Line
- Composition Docs
- Time Plan
- Detailed Functional Specification
- Develop New Codes
- Bugs Fix
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing

## Security Training
- Security Kickoff & Register with SWI
- Security Design Best Practice
- Threat Modeling
- Detailed Design Specification
- Detailed Functional Specification
- Use Security Development Tools & Security Best Dev & Test Practice
- Create Security Docs & Tools for Product
- Prepare Security Response Plan
- Penetration Testing
- Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Security Push
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools

## Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Notification
- Notification & Ownership
- Change Validation
- Application Retirement

## Change
- Change Validation
- Application Retirement

## Integration
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools

## QA
- Define Security Requirements (System & Functionalities)
- Define Secure Coding Standard
- Define Secure Frameworks
- Mapping Security Requirements

## Release
- Functional Lists
- Quality Guide Line
- Composition Docs
- Time Plan
- Detailed Functional Specification
- Develop New Codes
- Bugs Fix
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing

## Security Training
- Security Kickoff & Register with SWI
- Security Design Best Practice
- Threat Modeling
- Detailed Design Specification
- Detailed Functional Specification
- Use Security Development Tools & Security Best Dev & Test Practice
- Create Security Docs & Tools for Product
- Prepare Security Response Plan
- Penetration Testing
- Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Security Push
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools

## Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Notification
- Notification & Ownership
- Change Validation
- Application Retirement

## Change
- Change Validation
- Application Retirement

## Integration
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools

## QA
- Define Security Requirements (System & Functionalities)
- Define Secure Coding Standard
- Define Secure Frameworks
- Mapping Security Requirements

## Release
- Functional Lists
- Quality Guide Line
- Composition Docs
- Time Plan
- Detailed Functional Specification
- Develop New Codes
- Bugs Fix
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing

## Security Training
- Security Kickoff & Register with SWI
- Security Design Best Practice
- Threat Modeling
- Detailed Design Specification
- Detailed Functional Specification
- Use Security Development Tools & Security Best Dev & Test Practice
- Create Security Docs & Tools for Product
- Prepare Security Response Plan
- Penetration Testing
- Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Security Push
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools

## Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Notification
- Notification & Ownership
- Change Validation
- Application Retirement

## Change
- Change Validation
- Application Retirement

## Integration
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools

## QA
- Define Security Requirements (System & Functionalities)
- Define Secure Coding Standard
- Define Secure Frameworks
- Mapping Security Requirements

## Release
- Functional Lists
- Quality Guide Line
- Composition Docs
- Time Plan
- Detailed Functional Specification
- Develop New Codes
- Bugs Fix
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing

## Security Training
- Security Kickoff & Register with SWI
- Security Design Best Practice
- Threat Modeling
- Detailed Design Specification
- Detailed Functional Specification
- Use Security Development Tools & Security Best Dev & Test Practice
- Create Security Docs & Tools for Product
- Prepare Security Response Plan
- Penetration Testing
- Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Security Push
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools

## Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Notification
- Notification & Ownership
- Change Validation
- Application Retirement

## Change
- Change Validation
- Application Retirement

## Integration
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools

## QA
- Define Security Requirements (System & Functionalities)
- Define Secure Coding Standard
- Define Secure Frameworks
- Mapping Security Requirements

## Release
- Functional Lists
- Quality Guide Line
- Composition Docs
- Time Plan
- Detailed Functional Specification
- Develop New Codes
- Bugs Fix
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing
- Security System and Functionality Testing
- Unit and Integrated System Security Analysis and Testing through Black Box Testing
- Automated Security Code Testing

## Security Training
- Security Kickoff & Register with SWI
- Security Design Best Practice
- Threat Modeling
- Detailed Design Specification
- Detailed Functional Specification
- Use Security Development Tools & Security Best Dev & Test Practice
- Create Security Docs & Tools for Product
- Prepare Security Response Plan
- Penetration Testing
- Final Security Review
- Security Servicing & Response Execution
- Code Signing A Check Point Express Sign Off
- RTM
- Products Support Service packs and Security Updates to support QFE

## Security Push
- Security Arch & Attack Surface Review
- Testing and Verification
- Bug Fix
- Security Unit Testing
- Automated Security Implementation Testing
- Unit testing through Automated White Box testing and Black Box Testing tools
If you can map between working role and security role....

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Main Responsibilities</th>
<th>Mapping to ITIL</th>
<th>Mapping to CLASP</th>
<th>Pattern Occurrence</th>
<th>Mapping to 800-53</th>
<th>Controls actor is</th>
</tr>
</thead>
<tbody>
<tr>
<td>🧪</td>
<td>Developer</td>
<td>Follow security engineering principles</td>
<td>Application developer</td>
<td>Implementer</td>
<td>Public web server, Cloud computing, SCV publication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🧪</td>
<td>SW architect</td>
<td>Responsible that all application security controls are added to the application design. Coordinates security specialist and developers.</td>
<td>Applications analyst/Designer, Architect</td>
<td></td>
<td>Cloud computing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🧪</td>
<td>Infrastructure architect</td>
<td>Maintenance planning and prioritisation. Planning for security infrastructure such as directories and networks.</td>
<td>IT Architect</td>
<td></td>
<td>Client module, Server module,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🧪</td>
<td>Quality manager</td>
<td>Ensures that the products meets (security) specifications.</td>
<td>Test manager</td>
<td>Test Analyst</td>
<td>Secure SDLC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🧪</td>
<td>External auditor</td>
<td>Security accreditation</td>
<td>Security Auditor</td>
<td></td>
<td>Cloud computing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🧪</td>
<td>Internal auditor</td>
<td>Security assessments</td>
<td>Security Auditor</td>
<td></td>
<td>Cloud computing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🧪</td>
<td>End user</td>
<td>The user of an IT-Service, can be internal business person or customer.</td>
<td>User</td>
<td></td>
<td>Data security, Client module,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🧪</td>
<td>Data privacy officer</td>
<td>Responsible to ensure that customer and employee data is managed according to privacy regulations in the jurisdictions where the data is processed or accessed. These responsibilities may be performed by the information asset owner, or delegated to Data Privacy Officer in larger organisations.</td>
<td></td>
<td>IT facility manager</td>
<td>Server module,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🧪</td>
<td>Facility manager</td>
<td>Responsible for the physical security.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Literature
- [Link to source](http://homes.cicou.unimi.it/~ronggo/lib/ssetn0729300046.pdf)
- [Link to another source](https://buildsecurity.us-cert.gov/ds/ay/this/articles/best-practices/requirements/548-4851.html)
You can design Security Architectures for your own
Use your security architecture based on your own Control catalog

Controls are based on [NIST 800-53](https://csrc.nist.gov/publications/detail/nistpubs/800-53/sp800-53rev4pg202111/SP-800-53%20Volume%201%20Security%20Controls%20202111.pdf), and there is a mapping to [ISO17799](https://www.iso.org/standard/12647.html) and [COBIT 4.1](https://www.isaca.org/cobit).
Business vs. Security

- **Exploit Economics**: It’s all about a bad code
- **Safer Software**: Make bad guys hard to find defect
- **Software w/o defect**: There is nothing bad guy use for..

Where should we start From Kick-off till end of the Lifecycle of S/W
THANKS YOU!!