#23, Improving Attention to Security in Software Design with Analytics and Cognitive Techniques

Jim Whitmore  
(former) IBM Distinguished Engineer  
Carlisle, PA  
jjwhitmore@ieee.org

Will Tobin  
Lead Developer, Secure Engineering, IBM  
Lexington, MA  
wctobin@us.ibm.com
Disclaimers

The content and the comments in the conference paper and this presentation represent the personal views of the authors and do not reflect the policies, practices or offerings of any business or organization.
The elements of a security focused software engineering process can be found in standards and reference documents.

Essential practices are mapped to Security Controls...

- NIST SP800-53R4 S&P Controls for US Govt [2]
- ISO/IEC 20243:2015 Trusted Technology Provider
- ISO/IEC 27034-1:2011 Application Security
- NIST CSF V1.1 Cyber Security Framework

“At scale” secure development requires automation for consistency, accountability, continual improvement, and forensics.

There is a widening chasm between the ease of creating software and difficulty of "building the right security in".

1. **Evolving threats** to systems, vulnerabilities in software and risks to businesses

2. **Expanding supply chain** of Open Source Software

3. **Emphasis on speed** over rigor in new styles of development
   - Traditional Security work products are not easily consumed or acted upon
   - Fail-fast-and-often development maximizes innovative function, but does not account for security risks
What if **Security Design** and **Security Analysis** was a transaction?

**Scenario #1**: "We need to build a..."

**Scenario #2**: "What is the root cause of..."

**Scenario #3**: "What is the attack surface for..."

---

**User Story**

A user signs up for a cloud service by completing the online registration form, providing personal information...

**Incident Description**

A bad actor inserts special characters into an html form to bypass input validation...

---

Project Artefacts:

- Threats & Risks
- Attack Vectors
- Test Cases
- Tutorials & Videos
- Exemplar Code

---

Secure Design Assistant
Prototype... Search of indexed Security Information Catalogs (CAPEC, CWE, CVE, etc.) using crafted natural language input... returns tabular data with prioritized attacks, weakness dependencies, descriptive information and metadata.

A user signs up for a cloud service by completing the online registration form, providing personal information...

Secure Design Assistant

Security Information Indexes and Catalogs

Prototype

Target

Crafted Narrative

Search indexed corpus

Metadata Analytics

Interactive Visualization

Artefact Generation

Cognitive NL dialog

Search indexed corpus

Cognitive Review & Refinement

Interactive Visualization

Artefact Generation
A user signs up for a cloud service by completing the online registration. The user enters identifying information into the HTML form and submits the form to the enrollment application.

The application performs input validation, stores personal data in the account database.

The application updates the identity management system with username, password and security questions. The application sends a confirmation response to the user email account. By clicking on the link in the email the user successfully completes account activation.

Related terms: browser, web server, identity, credentials, SSL, TLS, certificates, sensitive information, private information, regulated data, HTML, HTTP, HTTPS, REST, API, XML, JAVA, JavaScript, JSON, Struts, Tomcat, SQL, DBMS, RDMS
(1/3) Web App: Enrollment Use Case Narrative

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th># Abuse Cases</th>
<th>Skill</th>
<th>SANS25</th>
<th>OWASP10</th>
<th>Industrial</th>
<th>Appliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAP0037</td>
<td>Retrieve Embedded Sensitive Data</td>
<td>3</td>
<td>Medium</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>CAP0077</td>
<td>Manipulating User-Controlled Variables</td>
<td>3</td>
<td>Low</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Web App: Enrollment Use Case Narrative

<table>
<thead>
<tr>
<th>Order</th>
<th>ID</th>
<th>Title</th>
<th># Abuse Cases</th>
<th>Skill</th>
<th>SANS25</th>
<th>OWASP10</th>
<th>Industrial</th>
<th>Appliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>CAP0018</td>
<td>Embedding Scripts in Non-Script Elements</td>
<td>13</td>
<td>Low</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>6</td>
<td>CAP0063</td>
<td>Cross Site Scripting</td>
<td>12</td>
<td>Low</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>13</td>
<td>CAP0076</td>
<td>Manipulating Web Input to File System Calls</td>
<td>11</td>
<td>Low</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>9</td>
<td>CAP0086</td>
<td>Embedding Script (XSS) in HTTP Headers</td>
<td>9</td>
<td>Low</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>10</td>
<td>CAP0085</td>
<td>AJAX Fingerprinting</td>
<td>8</td>
<td>Medium</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>11</td>
<td>CAP0001</td>
<td>Accessing Functionality Not Properly Constrained by ACLs</td>
<td>6</td>
<td>Low</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>CAP0062</td>
<td>Cross Site Request Forgery</td>
<td>4</td>
<td>Medium</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>5</td>
<td>CAP0084</td>
<td>XQuery Injection</td>
<td>3</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CAP0089</td>
<td>Pharming</td>
<td>3</td>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CAP0470</td>
<td>Expanding Control over the Operating System from the Database</td>
<td>2</td>
<td>High</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>1</td>
<td>CAP0066</td>
<td>SQL Injection</td>
<td>1</td>
<td>Low</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>CAP0126</td>
<td>Path Traversal</td>
<td>1</td>
<td>Low</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CAP0110</td>
<td>SQL Injection through SOAP Parameter Tampering</td>
<td>1</td>
<td>Medium</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>12</td>
<td>CAP0101</td>
<td>Server Side Include (SSI) Injection</td>
<td>1</td>
<td>Medium</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Pie Chart**:
- **Critical**: 2
- **Very High**: 14
- **High**: 26
- **Medium**: 7
- **Medium Low**: 7
- **Low**: 1
- **Unspecified**: 20
- **High Medium**: 23

**Bar Chart**:
- **75%**: SW Related
- **25%**: Net SW

---

IEEE SecDev 9/25/2017
(3/3) Web App: Enrollment Use Case Narrative

CAP0086: Embedding Script (XSS) in HTTP Headers

CAP0084: XQuery Injection

CAP0076: Manipulating Web Input to File System Calls

CAP0126: Path Traversal

CAP0089: Pharming

CAP0063: Cross Site Scripting

CAP0085: Ajax Fingerprinting

CAP0018: Embedding Scripts in Non-Script Elements

CAP001: Accessing Functionality not properly Constrained by ACLs

CAP0101: Server Side Include (SSI) Injection

CAP0062: Cross Site Request Forgery

CAP0110: SQL Injection through SOAP Parameter Tampering

CAP0066: SQL Injection

CAP0470: Expanding Control over OS from Database

IEEE SecDev 9/25/2017
Comparison of Web App narrative vs. Cloud Container Narrative

**Web App**

- Critical: 2
- Very High: 14
- High: 26
- Medium: 6
- Medium Low: 8
- Low: 3
- Unspecified: 19
- High Medium: 22

**Cloud Container**

- Critical: 4
- Very High: 14
- High: 17
- Medium: 5
- Medium Low: 5
- Low: 1
- Unspecified: 34
- High Medium: 20

Bar charts show the distribution of vulnerabilities by category and relationship to SW-related and SW-related incidents.
In our opinion...

Analysis of Natural Language User Stories and Component Descriptions has the potential to:

- **reduce time** for security design / review
- **augment skills** with just-in-time security knowledge and artefacts
- **enable continual improvement** for security across projects and components
- **elevate and mediate the security discussion** between developers, security professionals and risk managers

Further investigation is required for:

- **integrating** cognitive capabilities
- **compiling and curating** Security Information Catalogs
- **developing the strategies** for Service Delivery
References


Cloud Container Narrative

Containers provide lightweight isolated operating environment called containers. A container is a set of functions that may include basic operating system, local file system, runtime, middleware and application enablers. These functions and services operate within trust boundaries, which means that the processes and data for one container are private and not accessible to or from any other user or container.

With containers, resources can be isolated, services restricted, and processes provisioned to have an almost completely private view of the operating system with their own process ID space, file system structure, and network interfaces. Multiple containers share the same kernel, but each container can be constrained to only use a defined amount of resources such as CPU, memory and I/O. Containers rely on the kernel's functionality and uses resource isolation (CPU, memory, block I/O, network, etc.) and separate namespaces to isolate the application's view of the operating system. Communication with external services are protected with encrypted channels with unique keys and digital certificates.

Users authenticate to an external identity management service by presenting credentials (username and password). Once authenticated the container is launched and the user can invoke the functions and services in the container. Container processes operate with privileges of end users. Users and container processes access local and remote services via APIs. The services may require elevated privileges for communications and data storage. Containers relies on infrastructure services: the OS Kernel, core processes, hypervisor, virtualization management functions or hardware devices or drivers that are not directly accessible to the user.

A single server or virtual machine can run several containers simultaneously. Containers may simplify the creation of highly distributed systems by allowing multiple applications, worker tasks and other processes to run autonomously on a single physical machine or across multiple virtual machines. This allows the deployment of nodes to be performed as the resources become available or when more nodes are needed, allowing a platform as a service of deployment and scaling. Containers also simplifies the creation and operation of task or workload queues and other distributed systems.

Related terms: Linux kernel (cgroups and namespaces), virtual machine, shared operating system, libcontainer library, libvirt, LXC (Linux Containers), systemd-nspawn, read, write, file, Cassandra, MongoDB or Riak, SSL, TLS