Standards in reporting Software Flaws: **SCAP**, CVE and CWE (Part 2)

Robin A. Gandhi, Ph.D.

University of Nebraska at Omaha (UNO)
College of Information Science and Technology (IS&T)
School of Interdisciplinary Informatics (Si2)
Nebraska University Center on Information Assurance (NUCIA)
Who am I?

• Job
  – Assistant Professor of Information Assurance at IS&T since Fall 2008

• Research highlights
  – Regulatory Requirements driven Risk Assessment
    – Using the semantic web to bridge the gap from high-level regulations to low-level technical evidence (Domain: SCADA)
  – Software Assurance in the Development Lifecycle
    – Building semantic templates for the most egregious software flaws
  – Cyber attack modeling and forecasting (CyCast)
    – Exploring disturbances in the human network to predict cyber attacks

• Teaching
  – Software Assurance (seniors/grad) New!
  – Foundations of Information Assurance (seniors/grad)
  – Introduction to Information Assurance (Freshmen) New!
  – Introduction to Computer Science II (Freshmen/Sophomore)
A two part talk (Recap Part 1)

• SCAP
  – What is it?
  – What does it do?
  – What will it take to realize its potential?
  – What do I need to do to start preparing for it?

• How can we better understand vulnerabilities
  – Research on semantic templates built from CWE and CVE enumerations
SCAP Philosophy (Recap Part 1)
Different Roles and Responsibilities

• **Information Assurance** professionals tend to focus on the protection of systems that they may NOT have built
  – Extrinsic and deployed view of the system
  – SCAP is geared towards improving the efforts of IA professionals (Vulnerability Assessment/ Hardening)

• **Software Assurance** professionals tend to focus on the development of software systems with security BUILT-IN
  – Intrinsic and functional view of the system
  – Weakness, attack and secure coding enumerations are geared towards improving the efforts of developers
Why Jonny Can’t write secure code?

• **Johnny, avoid these weaknesses…. Period!**
  – Common Weaknesses Enumeration (CWE)

• **Johnny…learn from your mistakes**
  – Common Vulnerabilities and Exposures (CVE)

• **Johnny…these are the ways of the bad guys**
  – Common Attack Patterns Enumeration and Classification (CAPEC)

• **Johnny…these are ways to develop secure code**
  – CERT secure coding guidelines
Poor Johnny!

Countless Do’s and Don’ts

42976 CVE Vulnerabilities

CWE v.19
668 Weaknesses
1043 Pages

CAPEC
311 Attack Patterns
Using Semantic Templates to Study Vulnerabilities Recorded in Large Software Repositories

Me
Harvey Siy
Yan Wu
Outline

• Information overload in the study of vulnerabilities
  – Large software repositories
  – Vulnerability databases
  – Weakness enumerations

• Our research efforts:
  – Building semantic templates to understand and categorize the information related to a vulnerability

• Ongoing progress

• Future work
The Paradox we face!

- Source Code Differences after the fix
- Log of Changes
- Mailing list Discussions
- Public Descriptions
- Vulnerability Databases
- Weakness Enumerations
- Bug tracking databases
Large Software Repositories

• Source code version control systems (CVS, SVN)
  – Support distributed development
  – Versioning, Merging and Backup functions
  – Huge!
• Log of changes
  – Brief descriptions of the change performed
  – who, when, what, why
• IDEs, Bug tracker databases (reporters, resolvers, discussions), Public websites
• Mailing list threads related to the changes
  – Stakeholders: Developers, Organizations
Vulnerability Databases

• Several databases available
  – IBM X-force
  – CERIAS
  – CERT
  – DARPA CIDF
  – BindView Hacker Shield
  – Many others...

• Common Vulnerability Enumeration (CVE)
  – 42976 Vulnerabilities as of 2010-07-21 09:22 CST
Learning from our mistakes

• The Landwehr Software Flaw Taxonomy (1993)
  – Genesis (How), Time of introduction (When), Location (Where)

• Several recent efforts have followed
  – Seven Pernicious Kingdoms, PLOVER, 19 Deadly Sins, OWASP top ten...

• The Common Weaknesses Enumeration (CWE) has tried to assimilate these efforts and bring consensus (http://cwe.mitre.org)
CWE Organization (rough)

Class Weaknesses
Base Weaknesses
Variant Weaknesses

Research View
Development View
OWASP View
Weaknesses in Software Written in C
Weakness Enumerations

• Common Weaknesses Enumeration (CWE)
  – (measurement) Unified, measurable set of software weaknesses
  – (communication) Effective sharing, description, selection, and use of software security tools and services
  – (management, prioritization) Better understanding and management of software weaknesses related to design and code
Problems, Problems, Problems...

- Most vulnerability related artifacts are in unstructured text
  - Makes aggregation of these artifacts harder
- No shortage of weakness enumerations and categorization
  - Adoption in projects is slow
    - many choices could be a factor
- Growing software complexity
  - Little or no effort to improve the mental model of the software developer to sense the possibility of a vulnerability
Reducing the Cognitive Overload

• Devil is in the Details
  – The details about vulnerabilities are enormous during the coding phases

• Simple guides can be more effective than a long checklist
  – The 3 golden questions to ask about each bug (1989)
    • Is this mistake somewhere else also?
    • What next bug is hidden behind this one?
    • What should I do to prevent bugs like this?
Reducing the Cognitive Overload

• Questions about security weaknesses
  – What are the **Software flaws** (commission, omission, operational) that lead to the weakness?
  – What are the defining characteristics of the **Weakness**?
  – What are the **Resources/Location** where the weakness is typically manifest?
  – What are the **Consequences** that the weakness precedes?
Tangling of information in the CWE

- **CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer**
  
  - The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.
  
  - Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data. As a result, an attacker may be able to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash.
Tangling of information in the CWE

- **CWE-119**: Failure to Constrain Operations within the Bounds of a Memory Buffer
  - The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.
  - Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data. As a result, an attacker may be able to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash.
Tangling of information in the CWE

• CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
  – The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.
  – A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer.
  – Buffer overflows often can be used to execute arbitrary code...
  – Buffer overflows generally lead to crashes
SOFTWARE-FAULT

BUFFER COPY WITHOUT CHECKING SIZE OF INPUT #120

WEAKNESS

ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786
ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124

IS-A
IS-A

FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119

RESOURCE/LOCATION

MEMORY-BUFFER #119

IS-A

BUFFER #119

PART-OF

INDEXABLE-RESOURCE #118

CONSEQUENCES

WRITE-WHAT-WHERE CONDITION #123

CAN-PRECEDE

OCCURS-IN

CAN-PRECEDE
Building a Semantic Template

• For each weakness type create a semantic template

**Concept Extraction (CE)**

• Exploration of the CWE structure to extract entries relevant to a weakness

**Template structuring (TS)**

• Software Fault, Weakness, Resources, Consequences

**Template refinement (TR)**

• Aggregation of Vulnerability Artifacts
• Annotation using Semantic template concepts
Concept Extraction

• CWE 1.6
• Development view
  – Suited for stakeholders in the SDLC
• Research view
  – Suited for research using the cwe; deep hierarchical structure
• Select a “Root entry”
  – CWE that provides the most abstract description of a weakness, that would be CWE 119 for BO
Concept Extraction

• Strategies:
  1. Navigate hierarchical relationships of the root entry
  2. Navigate non-taxonomical relationships such as “Can Precede”, “Can Follow”, “Peer-of”
  3. Visualization of the CWE XML specification
     o A graph is generated using graphviz
  4. Keyword search on the CWE hyperlinked document
     o Followed by exploration of parent, sibling and child categories of the discovered CWE, for relevance to the root entry
Template Structuring

• Each CWE identified in the previous step is analyzed for concepts along the conceptual unit of the semantic template

• Relationships among the CWE entries are then used to structure the identified concepts into a coherent semantic template

• CWE-120: Buffer Copy without Checking Size of Input (‘Classic Buffer Overflow’)
  – The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.
  – A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer.
  – Buffer overflows often can be used to execute arbitrary code...
  – Buffer overflows generally lead to crashes
Buffer Overflow

**WEAKNESS**
- ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786
- ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124
- FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119
- IMPROPER-ACCESS-OF-INDEXABLE-RESOURCE #118

**CAN-PRECEDE**

**RESOURCE/LOCATION**
- STACK-BASED #121
- STATIC #129
- HEAP-BASED #122
- MEMORY-BUFFER #119
- BUFFER #119
- INDEXABLE-RESOURCE #118

**CONSEQUENCES**
- WRITE-WHAT-WHERE CONDITION #123
- UNCONTROLLED MEMORY ALLOCATION #789
- INFORMATION LOSS OR OMISSION #199 #221

**SOFTWARE-FAULT**
- SIGN ERRORS #194 #195 #196
- INCORRECT-BUFFER-SIZE-CALCULATION #131
- IMPROPER-INPUT-VALIDATION #20
- OFF-BY-ONE #193
- INTEGER COERCION ERROR #192
- INTEGER OVERFLOW #190 #680
- INTEGER UNDERFLOW #191
- WRAP-AROUND ERROR #128
- POINTER ERRORS #467 #468
- IMPROPER HANDLING OF EXTRA VALUES #231
- RETURN OF POINTER VALUE OUTSIDE OF EXPECTED RANGE #466
- IMPROPER NULL TERMINATION #170
- IMPROPER VALIDATION OF ARRAY INDEX #129 #789
- BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ('CLASSIC BUFFER OVERFLOW') #120

**CAN-PRECEDE**
Apache HTTP Server

- Widely used web server
- Open Source project with a large software repository readily available
- Due to the project size and its complexity, various vulnerabilities have occurred and solved during its lifetime
CVE (CAN-2004-0492)

- National Vulnerability Database *(Vulnerability Database)*
  - **Heap-based buffer overflow** in `proxy_util.c` for `mod_proxy` in Apache 1.3.25 to 1.3.31 allows remote attackers to **cause a denial of service (process crash)** and possibly **execute arbitrary code** via a negative `Content-Length` HTTP header field, which causes a large amount of data to be copied.

- Apache Security Reports *(Public website)*
  - A **buffer overflow** was found in the Apache proxy module, `mod_proxy`, which can be triggered by receiving an **invalid Content-Length header**. In order to exploit this issue an attacker would need to get an Apache installation that was configured as a proxy to connect to a malicious site. This would cause the Apache child processing the request to crash, although this does not represent a significant Denial of Service attack as requests will continue to be handled by other Apache child processes. This issue may lead to remote arbitrary code execution on some BSD platforms.
Apache Log of Changes (Fix)

Files Fixed in Project:

Roles

Fix time
Source Code Differences

```
revision 103191, Mon Mar 29 17:47:15 2004 UTC

485  int ap_proxy_http_handler(request_rec *r
486    content_length = ap_table_get(resp_hdrs, "Content-Length"),
487    if (content_length != NULL) {
488      c->len = ap_strtol(content_length, NULL, 10);
489    }
490
491  }
492
493  }
494
495  }
496
497  }

revision 103896, Fri Jun 11 07:54:38 2004 UTC

485  int ap_proxy_http_handler(request_rec *r
486    content_length = ap_table_get(resp_hdrs, "Content-Length"),
487    if (content_length != NULL) {
488      c->len = ap_strtol(content_length, NULL, 10);
489    }
490
491    if (c->len < 0) {
492      ap_kill_timeout(r);
493      return ap_proxyerror(r, HTTP_BAD_GATEWAY, ap_pstrcat(r->pool,
494        "Invalid Content-Length from remote server", NULL));
495    }
496
497  }
498
499  }
```
Study of the Vulnerability

SOFTWARE-FAULT

WEAKNESS

ACCESS AND OUT-OF-BOUNDS WRITE #125, #126, #127, #786
ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124

CVE: .... possibly execute arbitrary code via a negative Content-Length HTTP header field...

Apache Website: buffer overflow .... can be triggered by receiving an invalid Content-Length header

Source Code: Added validation criteria to avoid negative input
if (c->len < 0) {

CVE: .... causes a large amount of data to be copied ...

Apache Website: This issue may lead to remote arbitrary code execution on some BSD platforms...

CVE: allows remote attackers to cause a denial of service (process crash) and possibly execute arbitrary code

Apache Website: cause the Apache child processing the request to crash ...
This issue may lead to remote arbitrary code execution on some BSD platforms...
Study of the Vulnerability

1

CVE: ....possibly execute arbitrary code via a negative Content-Length HTTP header field...

Apache Website: buffer overflow ....can be triggered by receiving an invalid Content-Length header

Source Code: Added validation criteria to avoid negative input

if (c->len < 0) {

2

CVE: ....causes a large amount of data to be copied

ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124

Can-Precede

OcCurs-In

Resource/Location

Stack-Based #121
Study of the Vulnerability

WEAKNESS

ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786

ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124

FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119

IMPROPER ACCESS OF INDEXABLE RESOURCE #118

IS-A

INDEXABLE RESOURCE #118

INDEX (POINTER #466 INTEGER #129)

PART OF

HTTP RESPONSE DATA STRUCTURE

BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ('CLASSIC BUFFER OVERFLOW') #120

WRITE WHAT WHERE CONDITION #123

CONSEQUENCES CAN PRECEDE RESOURCE/LOCATION STACK BASED #121 STATIC #129 HEAP BASED #122 MEMORY BUFFER #119 BUFFER INDEXABLE RESOURCE #118 IS A PART OF IS A IS A IS A INDEX

CAN-PRECEDE

ACCESS AND OUT-OF-BOUNDS WRITE

OCCURS-IN

RESOURCES/LOCATIONS

STACK-BASED #121

IS-A

CONSEQUENCES

CVE : ....possibly execute arbitrary code via a negative Content-Length HTTP header field...

Apache Website: buffer overflow ....can be triggered by receiving an invalid Content-Length header

Source Code: Added validation criteria to avoid negative input

if (c->len < 0) {

CVE : ....causes a large amount of data to be copied ...

Apache Website: This issue may lead to remote arbitrary code execution on some BSD platforms...

CVE : allows remote attackers to cause a denial of service (process crash) and possibly execute arbitrary code... Apache Website: cause the Apache child processing the request to crash ...

This issue may lead to remote arbitrary code execution on some BSD platforms
CVE: Heap-based buffer overflow in proxy_util.c for mod_proxy in Apache 1.3.25 to 1.3.31...

Apache Website: A buffer overflow was found in the Apache proxy module, mod_proxy ...

Apache Change History: ... buffer overflow in later code...

CVE: allows remote attackers to cause a denial of service (process crash) and possibly execute arbitrary code via a negative Content-Length header ...

...can be triggered by receiving an invalid Content-Length header ...

Validation criteria to avoid negative input
if (c->len < 0) {

CVE: causes a large amount of data to be copied...

Apache Website: This issue may lead to remote arbitrary code execution on some BSD platforms...

This issue may lead to remote arbitrary code execution on some BSD platforms...
**Study of the Vulnerability**

**WEAKNESS**

ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786

ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124

FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119

IMPROPER ACCESS OF INDEXABLE RESOURCE #118

IS-A

CONSEQUENCES

WRITE-WHAT-WHERE CONDITION #123

CAN-PRECEDE

CVE : Heap-based buffer overflow in proxy_util.c for mod_proxy in Apache 1.3.25 to 1.3.31...

Apache Website: A buffer overflow was found in the Apache proxy module, mod_proxy ...

Apache Change History: ... buffer overflow in later code...

CVE : allows remote attackers to cause a denial of service (process crash) and possibly execute arbitrary code

Apache Website: cause the Apache child processing the request to crash ... This issue may lead to remote arbitrary code execution on some BSD platforms
Ontology Representation

• Semantic web based representation
  – Allow inferences and queries over a large collection of semantically annotated vulnerability artifacts
  – Examples
    • “Show past vulnerabilities that related to buffer overflow weaknesses that precedes arbitrary code execution”
    • “Which software fault most often precedes the buffer overflow weaknesses?”
Semantic Web Visualization
Common Attack Pattern
Enumeration and Classification (CAPEC)

• A shared indexing standard for common attacks patterns used in exploits or malware

• Attack patterns
  – Capture and communicate an attackers perspective
    • Common vocabulary to express attack vectors
  – List of common methods to exploit vulnerabilities
  – A “destructive” way of thinking
    • Know your enemy. Defense alone is not enough.

• http://capec.mitre.org/
CAPEC-100: Overflow Buffers

**Description**

Buffer Overflow attacks target improper or missing bounds checking on buffer operations, typically triggered by input injected by an attacker. As a consequence, an attacker is able to write past the boundaries of allocated buffer regions in memory, causing a program crash or potentially redirection of execution as per the attacker's choice.

**Summary**

1. The attacker identifies a buffer to target. Buffer regions are either allotted on the stack or the heap, and the exact nature of attack would vary depending on the location of the buffer.
2. Next, the attacker identifies an injection vector to deliver the excessive content to the targeted buffer.
3. The attacker crafts the content to be injected. If the intent is to simply cause the software to crash, the content need only consist of an excessive quantity of random data. If the intent is to leverage the overflow for execution of arbitrary code, the attacker will craft a set of content that not only overflows the targeted buffer but does so in such a way that the overwritten return address is replaced with one of the attacker's choosing which points to code injected by the attacker.
4. The attacker injects the content into the targeted software.
5. Upon successful exploitation, the system either crashes or control of the program is returned to a location of the attacker's choice. This can result in execution of arbitrary code or escalated privileges, depending upon the exploited target.

**Attack Prerequisites**

- Targeted software performs buffer operations.
- Targeted software inadequately performs bounds-checking on buffer operations.
- Attacker has the capability to influence the input to buffer operations.
Ongoing Work (Injection Template)
**Ongoing Work (Injection Template)**

---

### Apache Website CVE-2007-5000

A cross-site scripting attack is possible.

### NVD CVE-2007-5000

A Cross-site scripting (XSS) vulnerability in the mod_imap module.

### Source Code Repository developer fix documentation

Fix cross-site scripting issue by escaping the URI.

### Source Code Repository Code Difference

- **File:** mod_imagemap.c
- **Line 482 modified to contain an explicit character set:**
  
  ```c
  ap_set_content_type(r, "text/html;charset=UTF-8");
  ```

### CAPEC-43: Exploiting Multiple Input Interpretation Layers

(Experimentation) Determine which character encodings are accepted by the application/system:

(Exploit) Perform XSS attacks.

---

### Source Code Repository developer fix documentation

Ensure that a charset parameter is sent in the content-type.

### Source Code Repository Code Difference

- **File:** mod_imagemap.c
- **Line 485 and 490 modified to escape html in URI:**
  
  ```c
  ap_escape_html(r->pool, r->uri)
  ```

---

### CAPEC-63: Simple Script Injection

(Experimentation) Use a list of XSS probe strings to inject script into resources accessed by the application.

(Exploit) Develop malicious JavaScript that is injected through vectors identified during the Experiment Phase.

---

### Apache Website CVE-2007-5000

A cross-site scripting attack is possible.

### NVD CVE-2007-5000

Allows remote attackers to inject...
Ongoing Work (Injection Template)

**Source Code Repository developer fix documentation:**
...ensure that a charset parameter is sent in the content-type ...

**Source Code Repository Code Difference:**
File: mod_imagemap.c
Line 482 modified to contain an explicit character set:
ap_set_content_type(r, "text/html;charset=ISO-8859-1");

**CAPEC-43: Exploiting Multiple Input Interpretation Layers:**
(Experimentation) Determine which character encodings are accepted by the application/system:
(Exploit) Perform XSS attacks

---

**Website CVE-2007-5000:** allows attackers to inject...

**Source Code Repository Code Difference:**
File: mod_imagemap.c
Line 485 and 490 modified to contain a character set:
```
Line 485 and 490 modified to contain an explicit character set:
ap_set_content_type(r, "text/html;charset=ISO-8859-1");
```

---

**Source Code Repository developer fix documentation:**
...allow remote attackers to inject...

**Source Code Repository Code Difference:**
File: mod_imagemap.c
Line 482 modified to contain an explicit character set:
ap_set_content_type(r, "text/html;charset=ISO-8859-1");

---

**Consequences:**
This vulnerability allows remote attackers to inject...
CAPEC-63: Simple Script Injection:
(Experimentation) Use a list of XSS probe strings to inject script into resources accessed by the application
(Exploit) Develop malicious JavaScript that is injected through vectors identified during the Experiment Phase

Apache Website CVE-2007-5000: allows remote attackers to inject arbitrary user-controlled data

NVD CVE-2007-5000: allows remote attackers to inject script or HTML...

Source Code Repository developer fix documentation: Fix cross-site-scripting issue by escaping the URI...

Source Code Repository Code Difference:
File: mod_imagemap.c
Line 485 and 490 modified to escape html in URI:
ap_escape_html(r->pool, r->uri)
CAPEC-43: Exploiting Multiple Input Interpretation Layers:
(Experimentation) Determine which character encodings are accepted by the application/system:
(Exploit) Perform XSS attacks

Source Code Repository developer fix documentation: Fix cross-site-scripting issue by escaping the URI...

Source Code Repository Code Difference:
File: mod_imagemap.c
Line 485 and 490 modified to escape html in URI:
ap_escape_html(r->pool, r->uri)

NVD CVE-2007-5000: Allows remote attackers to inject...

NVD CVSS CVE-2007-5000 impact type: Allows unauthorized modification
Future Work

• Integrate with existing static and dynamic analysis tools to enhance reporting capabilities
  – Provide layers of guidance to a developer upon detection of a software flaw
  – Organize and retrieve knowledge of past vulnerabilities
  – Verify patch submissions
• Investigate project/developer specific coding errors and vulnerability fix patterns
• Other usage scenarios in the SDLC
Some take aways...

• Ask Johnny (or your software vendor):
  – How many CWEs have you attempted to explicitly avoid in your software?
  – What CWEs can our Threats take advantage of?
    • I want you to build a shopping cart, while avoiding those CWEs...
  – What CAPECs do your testing efforts map to?
  – What CWEs do the vulnerabilities in your code typically map to? Have you taken any training for them?
    • Have you looked at the semantic templates by being developed at UNO/NUCIA for those CWEs?
      – http://faculty.ist.unomaha.edu/rgandhi/st/
CERT Secure Coding Guidelines

https://www.securecoding.cert.org/

CERT Secure Coding Standards

The CERT Oracle Secure Coding Standard for Java

The CERT C Secure Coding Standard

The CERT C++ Secure Coding Standard
Acknowledgement

• This research is funded in part by Department of Defense (DoD)/Air Force Office of Scientific Research (AFOSR), NSF Award Number FA9550-07-1-0499, under the title “High Assurance Software”
Thank you for your Attention