The CWE/SANS Top 25: Towards Minimum Due Care in Software Security

Steve Christey
March 13, 2009

http://cwe.mitre.org/top25

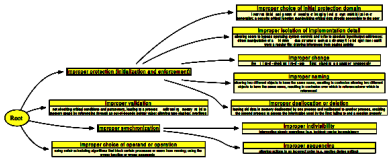
MITRE

Outline

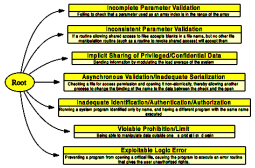
- Context
- Contributors and construction
- Considerations and caveats
- Clarifications

Making Security Measurable™

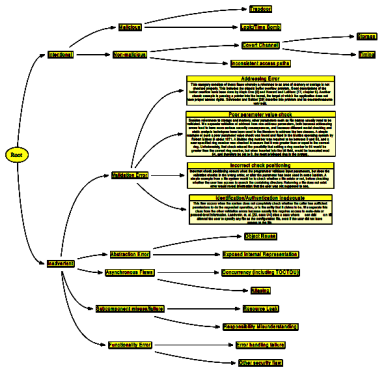
Protection Analysis



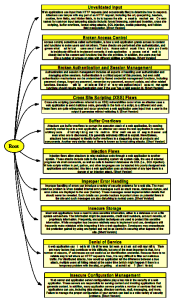
RISOS



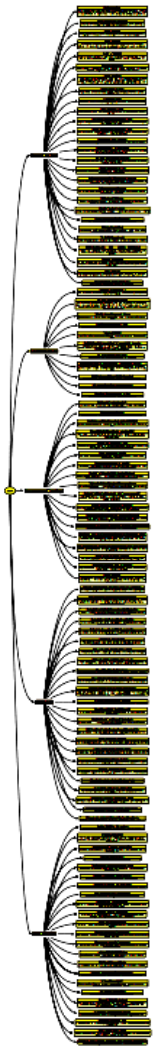
Weber



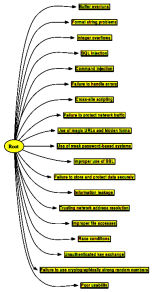
OWASP



CLASP

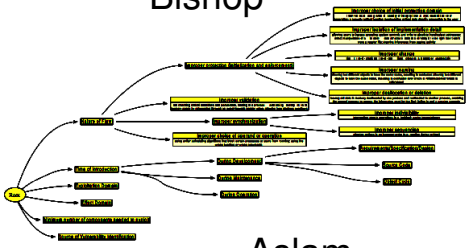


Microsoft

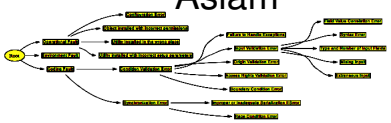


PLOVER

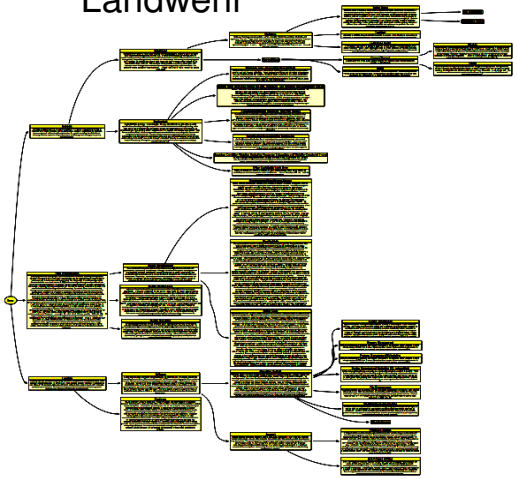
Bishop



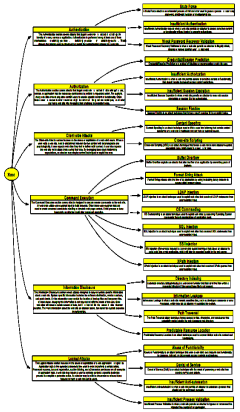
Aslam



Landwehr

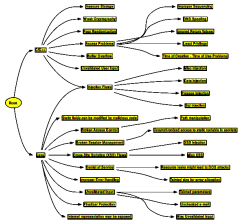


WASC

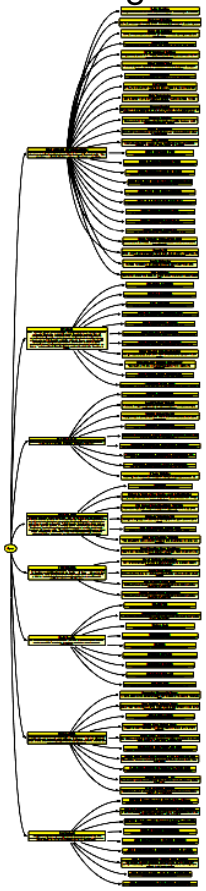


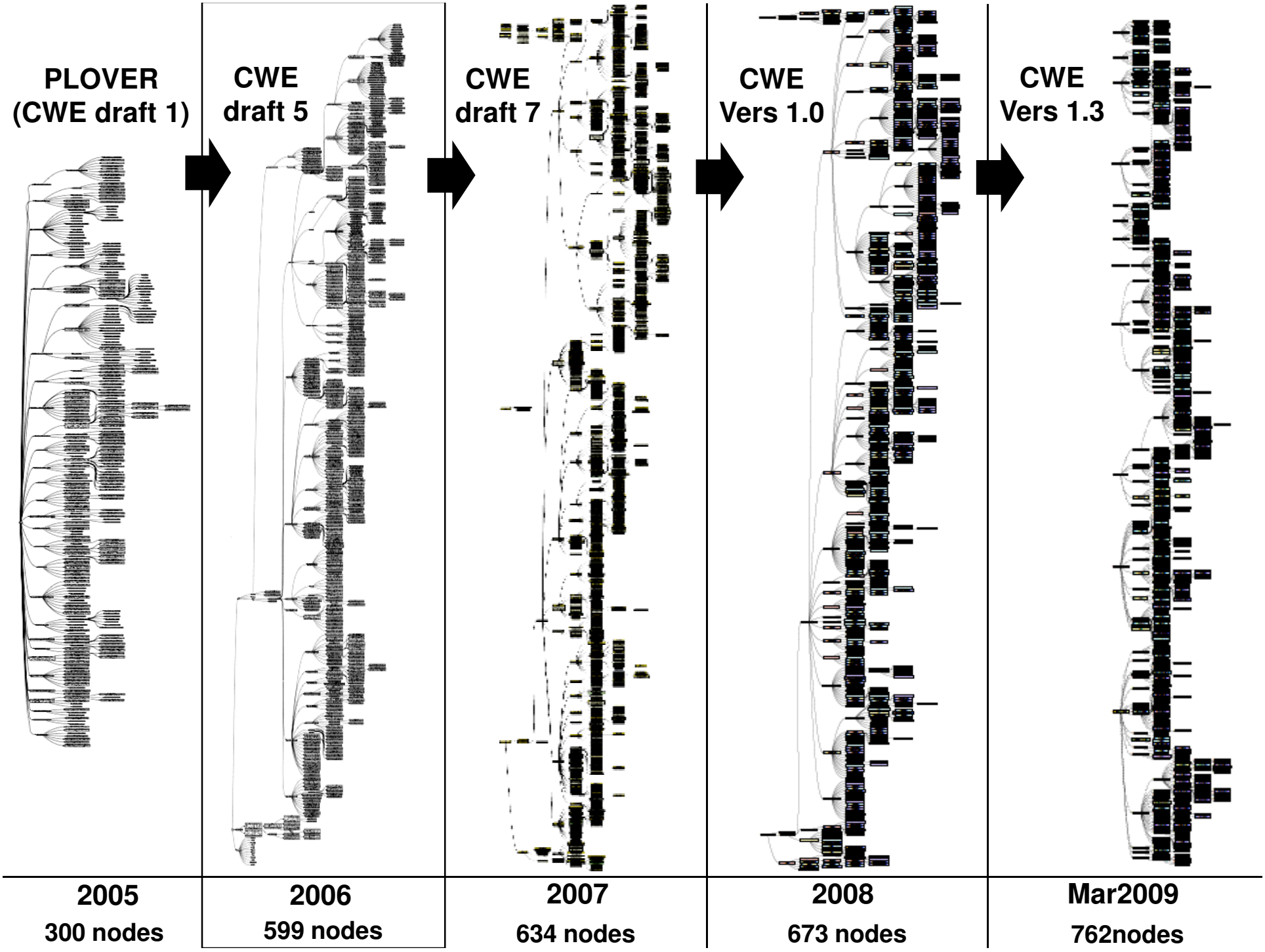
Tool B

Tool A



7 Kingdoms




PLOVER
(CWE draft 1)
CWE
draft 5
CWE
draft 7
CWE
Vers 1.0
CWE
Vers 1.3
2005
300 nodes
2006
599 nodes
2007
634 nodes
2008
673 nodes
Mar2009
762nodes

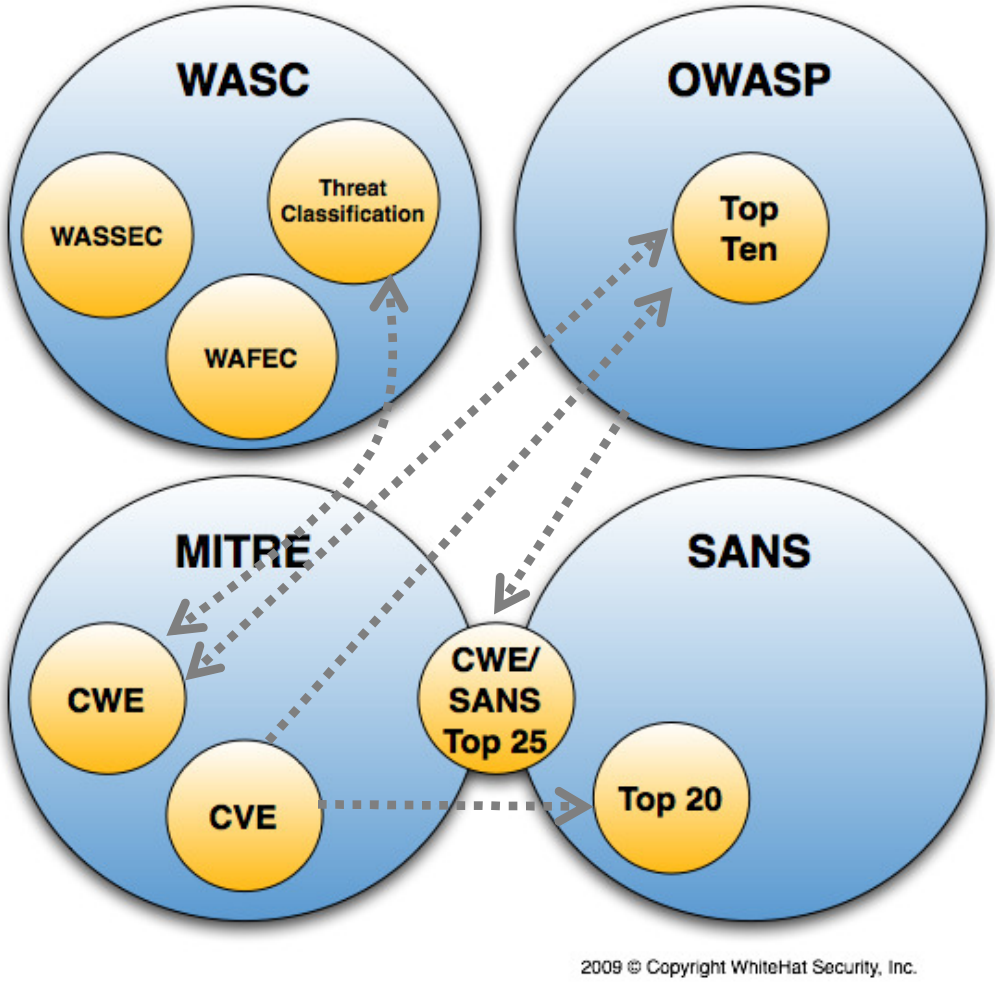
Current Community Contributing to the Common Weakness Enumeration

- AppSIC
- Apple
- Aspect Security
- Booz Allen Hamilton Inc.
- Cenzic
- CERIAS/Purdue University
- CERT/CC
- Cigital
- Codenomicon
- Core Security
- Coverity
- DHS
- Fortify
- Gramma Tech
- IPA/JPCERT
- IBM
- Interoperability Clearing House
- JHU/APL
- JMU
- Kestrel Technology
- KDM Analytics
- Klocwork
- McAfee
- Microsoft
- MIT Lincoln Labs
- MITRE
- North Carolina State University
- NIST
- NSA
- OMG
- Oracle
- Ounce Labs
- OSD
- OWASP
- Palamida
- Parasoft
- PolySpace Technologies
- proServices Corporation
- SANS Institute
- SecurityInnovation
- Security University
- Semantic Designs
- SofCheck
- SPI Dynamics
- SureLogic, Inc.
- Symantec
- UNISYS
- VERACODE
- Watchfire
- WASC
- Whitehat Security, Inc.

Making Security Measurable™

To join send e-mail to cwe@mitre.org

Some Context



Making Security Measurable™

From Jeremiah Grossman, WhiteHat Security

CWE/SANS Top 25 Programming Errors

- Sponsored by:
  - **National Cyber Security Division (DHS)**
  - **Information Assurance Division (NSA)**
- List was selected by a group of security experts from 35 organizations including:
  - **Academia: Purdue, Univ. of Cal., N. Kentucky Univ.**
  - **Government: CERT, NSA, DHS**
  - **Software Vendors: Microsoft, Oracle, Red Hat, Apple**
  - **Security Vendors: Veracode, Fortify, Cigital, Symantec**

Making Security Measurable™

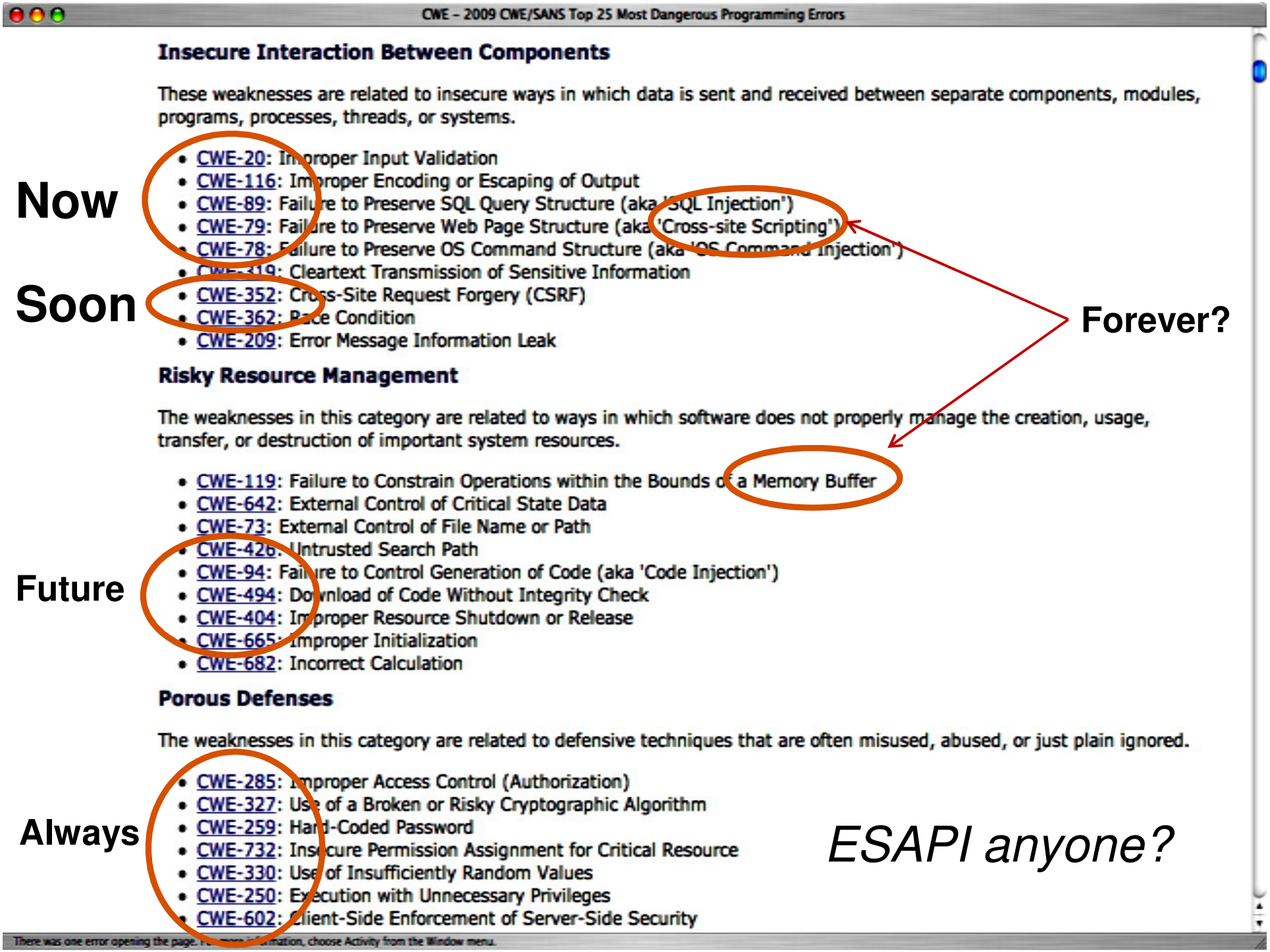
Robert C. Seacord	CERT
Pascal Meunier	CERIAS, Purdue University
Matt Bishop	University of California, Davis
Kenneth van Wyk	KRvW Associates
Masato Terada	Information-Technology Promotion Agency (IPA) (Japan)
Sean Barnum	Cigital, Inc.
Mahesh Saptarshi	Symantec Corporation
Cassio Goldschmidt	Symantec Corporation
Adam Hahn	MITRE
Jeff Williams	Aspect Security and OWASP
Carsten Eiram	Secunia
Josh Drake	iDefense Labs at VeriSign, Inc.
Chuck Willis	MANDIANT
Michael Howard	Microsoft
Bruce Lowenthal	Oracle Corporation
Mark J. Cox	Red Hat Inc.
Jacob West	Fortify Software
Djenana Campara	Hatha Systems
James Walden	Northern Kentucky University
Frank Kim	ThinkSec
Chris Eng	Veracode, Inc.
Chris Wysopal	Veracode, Inc.
Ryan Barnett	Breach Security
Antonio Fontes	New Access SA (Switzerland)
Mark Fioravanti II	Missing Link Security Inc.
Ketan Vyas	Tata Consultancy Services (TCS)
Lindsey Cheng	Secured Sciences Group, LLC
Ian Peters	Secured Sciences Group, LLC
Tom Burgess	Secured Sciences Group, LLC
Hardik Parekh	RSA - Security Division of EMC Corporation
Matthew Coles	RSA - Security Division of EMC Corporation
Mouse	
Ivan Ristic	
Apple Product Security	
Software Assurance Forum for Excellence in Code (SAFECode)	
Core Security Technologies Inc.	
Depository Trust & Clearing Corporation (DTCC)	
The working group at the first OWASP ESAPI Summit	
National Security Agency (NSA) Information Assurance Division	
Department of Homeland Security (DHS) National Cyber Security Division	

Special thanks to Alan Paller and Mason Brown (SANS), and Janis Kenderdine and Conor Harris (MITRE)

Main Goals

- Raise awareness for developers
- Help universities to teach secure coding
- Empower customers who want to ask for more secure software
- Provide a starting point for in-house software shops to measure their own progress

Making Security Measurable™


CWE – 2009 CWE/SANS Top 25 Most Dangerous Programming Errors
Insecure Interaction Between Components
These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.
CWE-20: Improper Input Validation
CWE-116: Improper Encoding or Escaping of Output
CWE-89: Failure to Preserve SQL Query Structure (aka 'SQL Injection')
CWE-79: Failure to Preserve Web Page Structure (aka 'Cross-site Scripting')
CWE-78: Failure to Preserve OS Command Structure (aka 'OS Command Injection')
CWE-319: Cleartext Transmission of Sensitive Information
CWE-352: Cross-Site Request Forgery (CSRF)
CWE-362: Race Condition
CWE-209: Error Message Information Leak
Risky Resource Management
The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.
CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer
CWE-642: External Control of Critical State Data
CWE-73: External Control of File Name or Path
CWE-426: Untrusted Search Path
CWE-94: Failure to Control Generation of Code (aka 'Code Injection')
CWE-494: Download of Code Without Integrity Check
CWE-404: Improper Resource Shutdown or Release
CWE-665: Improper Initialization
CWE-682: Incorrect Calculation
Porous Defenses
The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.
CWE-285: Improper Access Control (Authorization)
CWE-327: Use of a Broken or Risky Cryptographic Algorithm
CWE-259: Hard-Coded Password
CWE-732: Insecure Permission Assignment for Critical Resource
CWE-330: Use of Insufficiently Random Values
CWE-250: Execution with Unnecessary Privileges
CWE-602: Client-Side Enforcement of Server-Side Security
Now
Soon
Future
Always
Forever?
ESAPI anyone?

Example 3:

This example takes user input, passes it through an encoding scheme and then creates a directory specified by the user.

Perl Example:

```
sub GetUntrustedInput {
  return($ARGV[0]);
}

sub encode {
  my($str) = @_;
  $str =~ s/\&/\&amp;/gs;
  $str =~ s/\"/\&quot;/gs;
  $str =~ s/\'/\&apos;/gs;
  $str =~ s/\</\&lt;/gs;
  $str =~ s/\>/\&gt;/gs;
  return($str);
}

sub doit {
  my $uname = encode(GetUntrustedInput("username"));
  print "<b>Welcome, $uname!</b><p>\n";
  system("cd /home/$uname; /bin/ls -l");
  }
```

The programmer attempts to encode dangerous characters, however the blacklist for encoding is incomplete (CWE-184) and an attacker can still pass a semicolon, resulting in a chain with command injection (CWE-77).

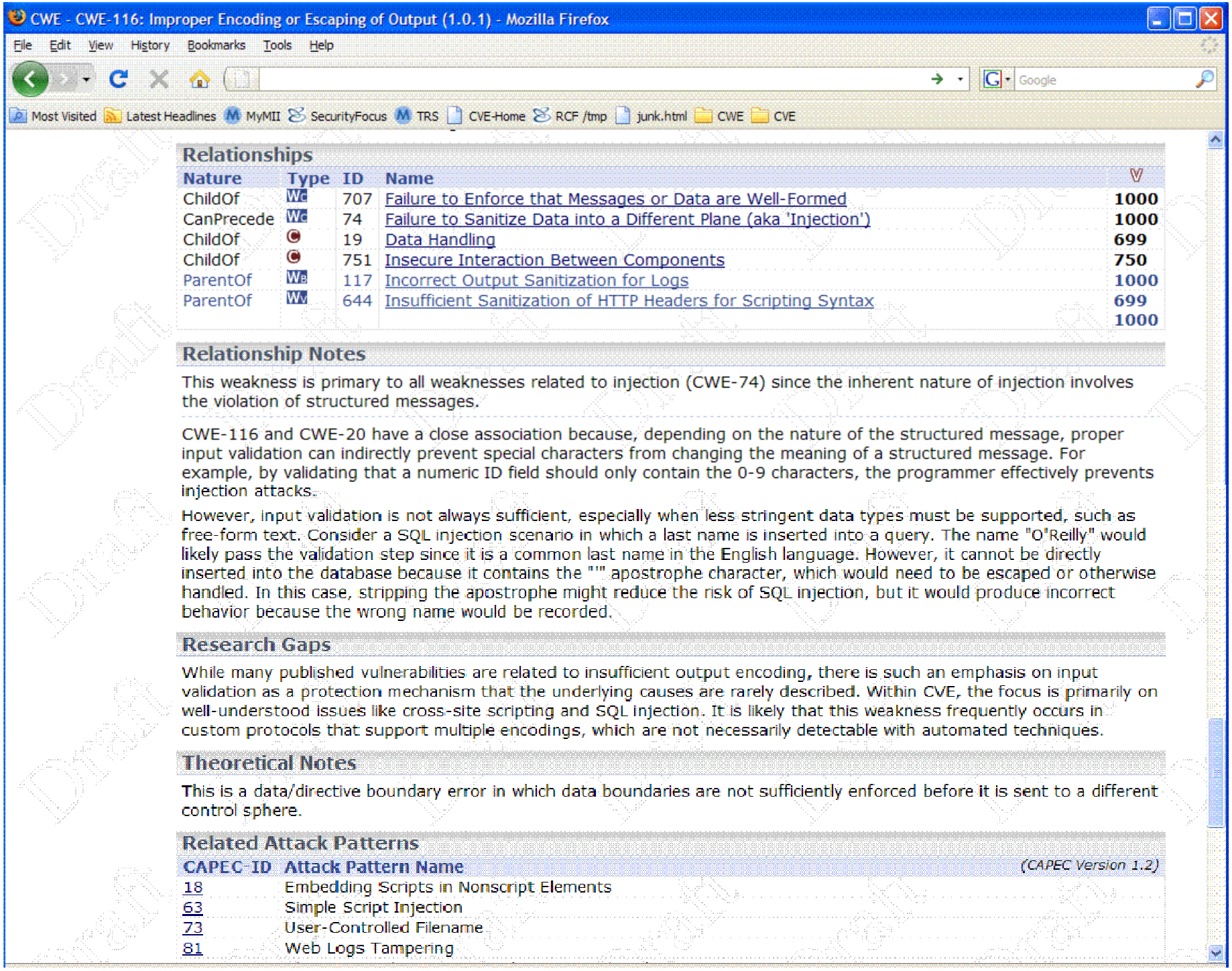
Additionally, the encoding routine is used inappropriately with command execution. An attacker doesn't even need to insert their own semicolon. The attacker can instead leverage the encoding routine to provide the semicolon to separate the commands. If an attacker supplies a string of the form:

```
pwd
```

then the program will encode the apostrophe and insert the semicolon, which functions as a command separator when passed to the system function. This allows the attacker to complete the command injection.

Observed Examples

Reference	Description
CVE-2008-0005	Program does not set the charset when sending a page to a browser, allowing for XSS exploitation when a browser chooses an unexpected encoding.
CVE-2008-0757	Cross-site scripting in chat application via a message, which normally might be allowed to contain arbitrary content.
CVE-2008-0769	Web application does not set the charset when sending a page to a browser, allowing for XSS exploitation when a browser chooses an unexpected encoding.
CVE-2008-3773	Cross-site scripting in chat application via a message subject, which normally might contain "&" and other XSS-related characters.
CVE-2008-4636	OS command injection in backup software using shell metacharacters in a filename; correct behavior would require that this filename could not be changed.



Relationships

Nature	Type	ID	Name	V
ChildOf	WC	707	Failure to Enforce that Messages or Data are Well-Formed	1000
CanPrecede	WC	74	Failure to Sanitize Data into a Different Plane (aka 'Injection')	1000
ChildOf	C	19	Data Handling	699
ChildOf	C	751	Insecure Interaction Between Components	750
ParentOf	WB	117	Incorrect Output Sanitization for Logs	1000
ParentOf	WV	644	Insufficient Sanitization of HTTP Headers for Scripting Syntax	699
				1000

Relationship Notes

This weakness is primary to all weaknesses related to injection (CWE-74) since the inherent nature of injection involves the violation of structured messages.

CWE-116 and CWE-20 have a close association because, depending on the nature of the structured message, proper input validation can indirectly prevent special characters from changing the meaning of a structured message. For example, by validating that a numeric ID field should only contain the 0-9 characters, the programmer effectively prevents injection attacks.

However, input validation is not always sufficient, especially when less stringent data types must be supported, such as free-form text. Consider a SQL injection scenario in which a last name is inserted into a query. The name "O'Reilly" would likely pass the validation step since it is a common last name in the English language. However, it cannot be directly inserted into the database because it contains the "'" apostrophe character, which would need to be escaped or otherwise handled. In this case, stripping the apostrophe might reduce the risk of SQL injection, but it would produce incorrect behavior because the wrong name would be recorded.

Research Gaps

While many published vulnerabilities are related to insufficient output encoding, there is such an emphasis on input validation as a protection mechanism that the underlying causes are rarely described. Within CVE, the focus is primarily on well-understood issues like cross-site scripting and SQL injection. It is likely that this weakness frequently occurs in custom protocols that support multiple encodings, which are not necessarily detectable with automated techniques.

Theoretical Notes

This is a data/directive boundary error in which data boundaries are not sufficiently enforced before it is sent to a different control sphere.

Related Attack Patterns

CAPEC ID	Attack Pattern Name	*(CAPEC Version 1.2)*
18	Embedding Scripts in Nonscript Elements	
63	Simple Script Injection	
73	User-Controlled Filename	
81	Web Logs Tampering	

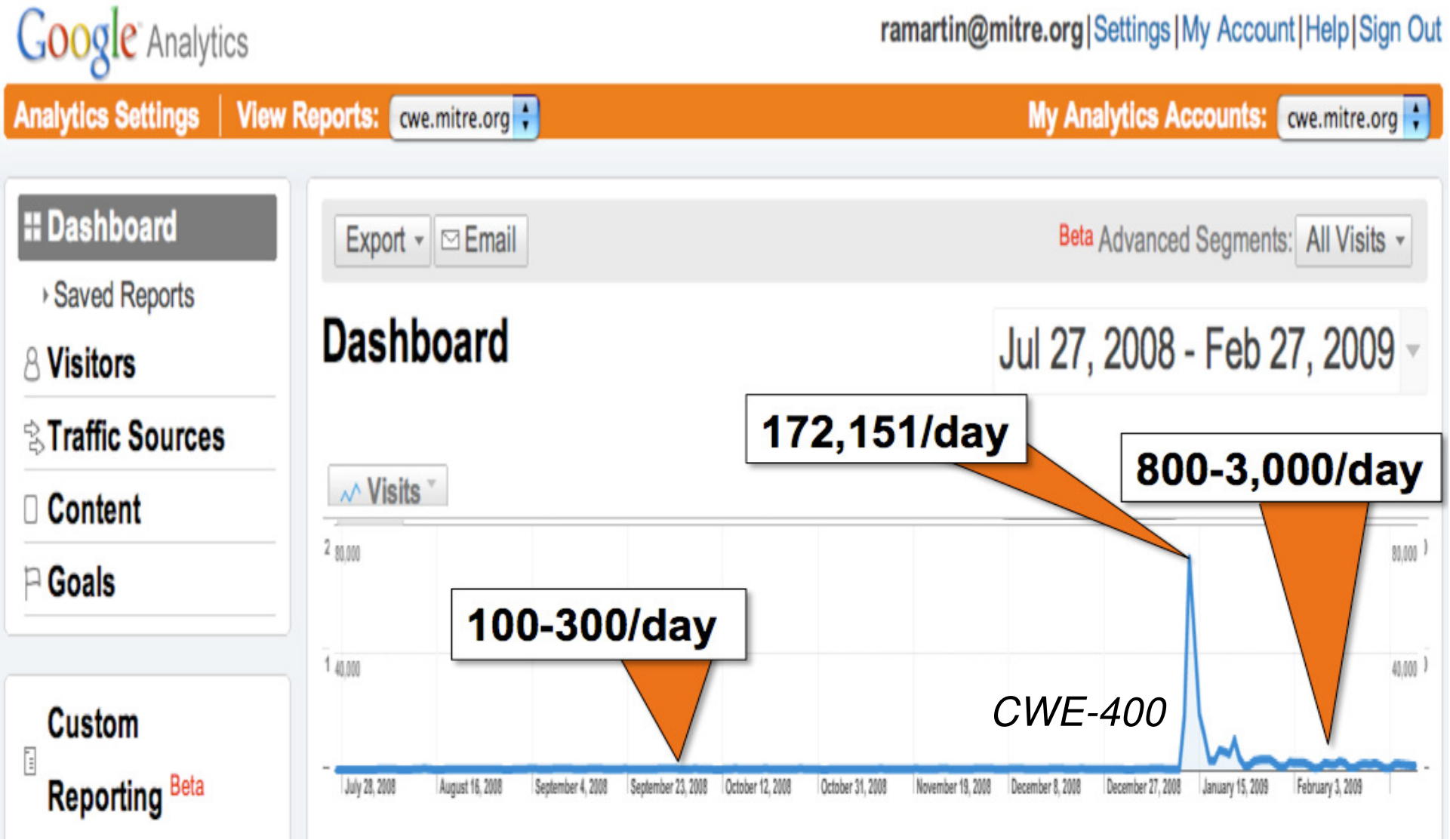
Who Did We Reach and Where?

- News: USA Today, Forbes, BBC
- Trade Publications
- Blogs, tweets, bookmarks
- Podcasts & Radio

- Developers
- Friends, Romans, Countrymen

Making Security Measurable™

People are Starved for Simplicity



Some Reactions (Paraphrased)

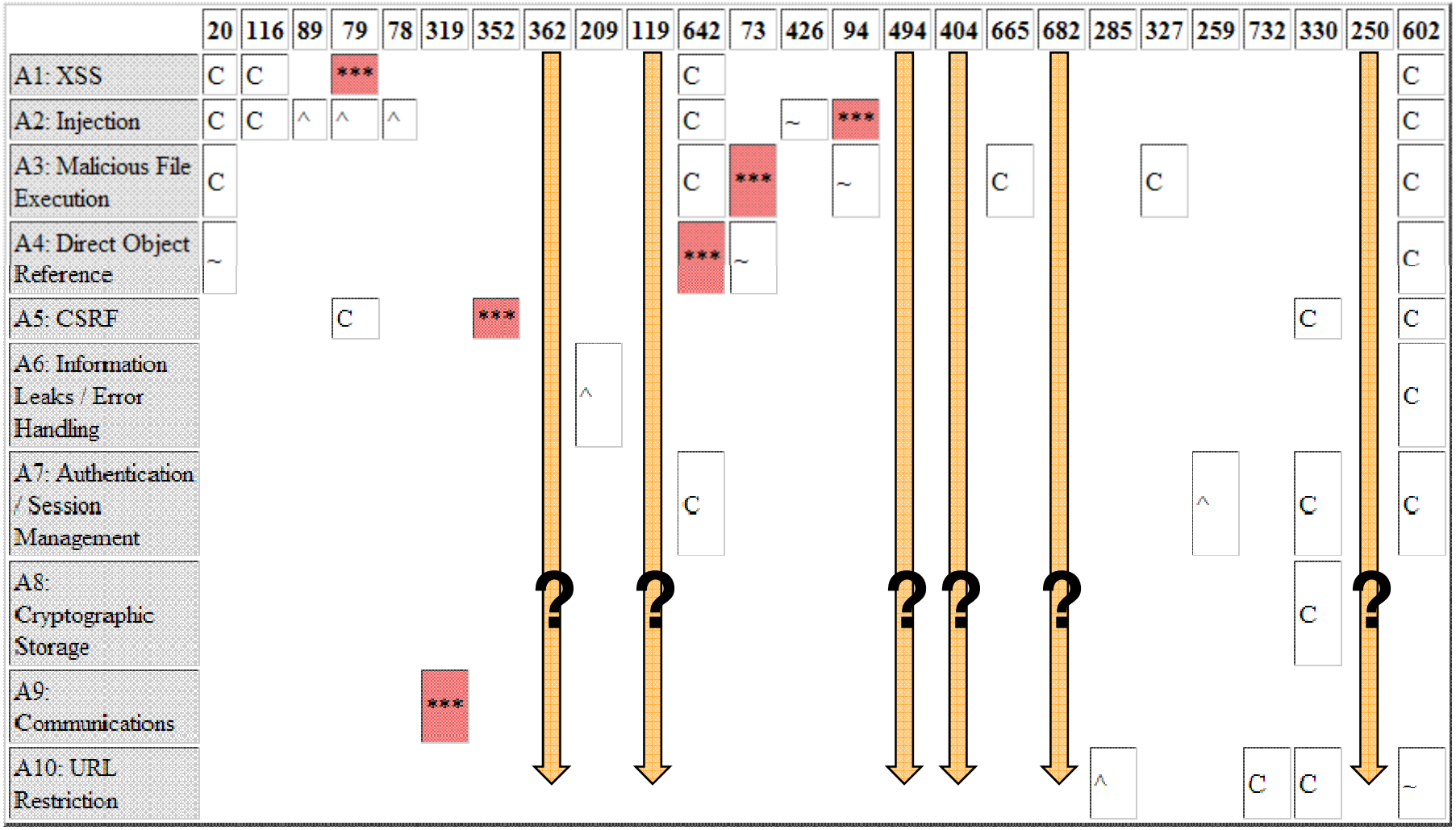
- “I never heard of any of these. Thanks!”
- “I have a feeling I’ll be busy this weekend.”
- “It’s convenient to have these all in one place”
- Blog title: “NSA flames N00bs”
- “You forgot #1: managers force us to meet deadlines.”
- “My boss asked what I thought about this.”
- “This complicates my job as a consultant”
- “This one is easy to fix.” “No it’s not!” “Oh, yeah.”
- “These are all just (web problems|injection|bugs)”
- [in vendor forum]
  - **Customer: “How have you protected against these?”**
  - **Vendor: <silence>**

Making Security Measurable™

Top 25 and OWASP Top 10

C	Causal (Chain)	***	Exact Match
~	Indirect Relationship	^	Abstraction Relationship

also see: http://securityninja.co.uk/blog/?p=132



Input Validation is Not The Answer for Everything

- Challenge: using only input validation, place this message into a database and present it on a web page, without any loss of data

Happy St. Patrick's Day!
I <3 green beer

Making Security Measurable™

Making the Top 25

- 12 weeks from start to finish
- Mailing list and private comments
- Multiple drafts
  - **Extensive feedback**
  - **Change summaries to contributors**
- Prioritization
  - **Severity and Prevalence**
  - **Not considered: frequency of exploit or occurrence, ease of fix, etc.**

Making Security Measurable™

Prevalence Based on 2008 CVE Data

Category	Count	%
SQL Injection	?	?%
XSS	?	?%
Buffer Overflow	?	?%
Directory Traversal	?	?%
PHP Include	?	?%
Symbolic Link	?	?%
Authorization Bypass	?	?%
DoS Malformed Input	?	?%
Information Leak	?	?%
Integer Overflow	?	?%
CSRF	?	?%
Bad Permissions	?	?%
Unnecessary Privile	?	?%

Hard cod rd	?	?%
Uploa	?	?%
We	?	?%
Form ring	?	?%
ient Randomness	?	?%
Meta haracter Injection	?	?%
Search Path	?	?%
Memory Leak	?	?%
Sensitive data root	?	?%
Race Condition	?	?%
DoS Flood	?	?%
CRLF Injection	?	?%
Eval Injection	?	?%
Numeric Error	?	?%

INCOMPLETE DATA

4855 total flaws tracked by CVE in 2008 (as of November 2008)

Prevalence Based on 2008 CVE Data

Category	Count	%
SQL Injection	?	??%
XSS	?	??%
Buffer Overflow	?	??%
Directory Traversal	?	??%
PHP Include	?	??%
Symbolic Link	?	??%
Authorization Bypass	?	??%
Dot-dot-slash	?	??%
Information Leakage	?	??%
Integer Overflow	?	??%
CSRF	?	??%
Bad Permissions	?	??%
Unnecessary Privileges	?	??%

Hard coded Password	?	??%
Upload of code	?	??%
Weak Crypto	?	??%
Format String	?	??%
Insufficient Randomness	?	??%
Metacharacter Injection	?	??%
Race Condition	?	??%
Memory Leak	?	??%
Sensitive data read	?	??%
Input Validation	?	??%
DoS Flood	?	??%
CRLF Injection	?	??%
Eval Injection	?	??%
Numeric Error	?	??%

Plus Info from Various Consultants Regarding "Internally Developed Code"

Making Security Measurable™

But I Want Numbers!

- Vulnerabilities != Weaknesses
- No 1-To-1 relationships between vulnerabilities, weaknesses, and attacks
  - **Send a long input… name 3 non-overflow weaknesses, quick!**
- Weakness counts don't exist
  - **… or, people aren't sharing**
- Severity is in the eye of the beholder
  - **Availability assumed to be less severe**

Making Security Measurable™

Threat Agent: The Skilled, Determined Attacker

- Skill
  - **Solid technical understanding of well-documented vulnerabilities**
  - **Knows white-box and black-box methods**
  - **Can combine multiple attacks**
- Determination
  - **Seeks to steal confidential data or take over an entire system**
  - **May collaborate with others**
  - **Willing to invest 20 hours**

It's not perfect… but it's a start

Making Security Measurable™

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

- CWE-20: Improper Input Validation
- CWE-116: Improper Encoding or Escaping of Output
- CWE-89: Failure to Preserve SQL Query Structure (aka 'SQL Injection')
- CWE-79: Failure to Preserve Web Page Structure (aka 'Cross-site Scripting')
- CWE-78: Failure to Preserve OS Command Structure (aka 'OS Command Injection')
- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-352: Cross-Site Request Forgery (CSRF)
- CWE-362: Race Condition
- CWE-209: Error Message Information Leak

Risky Resource Management

The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

- CWE-119: ***I don't care about buffer overflows...***
- CWE-642: External Control of Critical State Data
- CWE-73: External Control of File Name or Path
- CWE-426: Untrusted Search Path
- CWE-94: Failure to Control Generation of Code (aka 'Code Injection')
- CWE-494: Download of Code Without Integrity Check
- CWE-404: Improper Resource Shutdown or Release
- CWE-665: Improper Initialization
- CWE-682: Incorrect Calculation

What's #26?

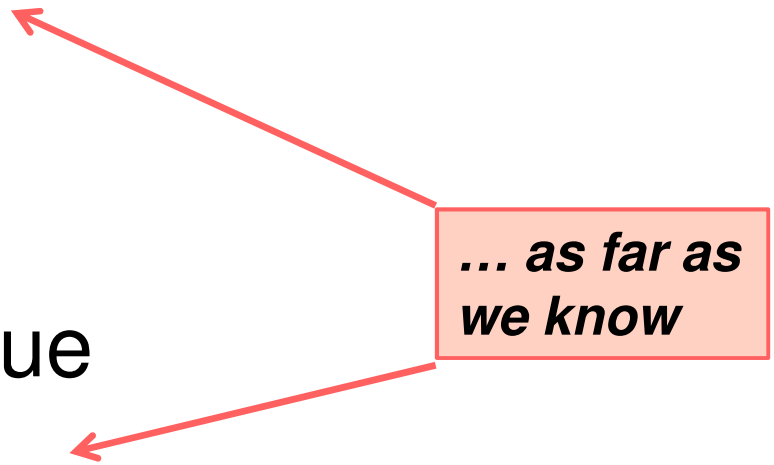
Porous Defenses

The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

- CWE-285: Improper Access Control (Authorization)
- CWE-327: Use of a Broken or Risky Cryptographic Algorithm
- CWE-259: Hard-Coded Password
- CWE-732: Insecure Permission Assignment for Critical Resource
- CWE-330: Use of Insufficiently Random Values
- CWE-250: Execution with Unnecessary Privileges
- CWE-602: Client-Side Enforcement of Server-Side Security

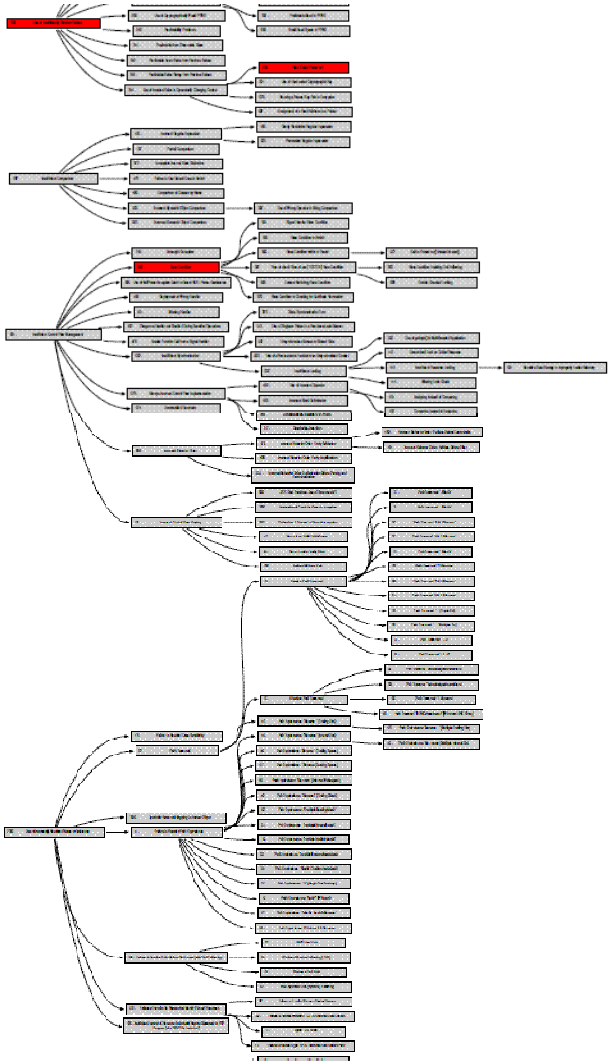
Fear #26… both of 'em

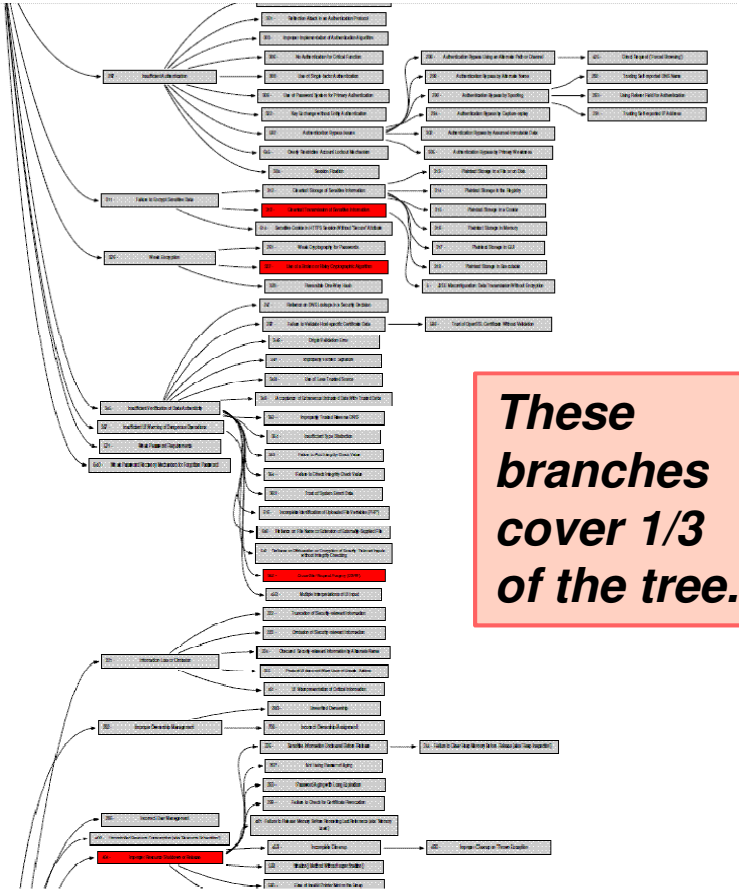
- Resource Exhaustion
  - **Not prevalent enough**
  - **Not severe enough**
    - Based on T25's criteria
- Unchecked Return Value
  - **Not prevalent enough**
  - **Rarely severe enough**
- Your own #26 is special



Making Security Measurable™

Fear the Rest:
The Top 25 compared to all CWE





These branches cover 1/3 of the tree.

Making Security Measurable™

Recent News Items Outside the Top 25

- BIND, OpenSSL: incorrect return value check and certificate spoofing (CWE-253)
- Twitter: lack of throttling or lockout for brute force password guessing (CWE-307)
- D.J. Bernstein djbdns: insufficient comparison allows DNS cache poisoning (CWE-697)

Making Security Measurable™

Some other entries on the Cusp

- CWE-93: CRLF Injection
- CWE-287: Improper Authentication
- CWE-573: Failure to Follow Specification
- CWE-749: Exposed Dangerous Method
- CWE-180: Validating before Canonicalizing
- CWE-470: Unsafe Reflection
- CWE-95: Eval Injection

Making Security Measurable™

http://cwe.mitre.org/top25/cusp.html

From Bad to Good: Mitigations

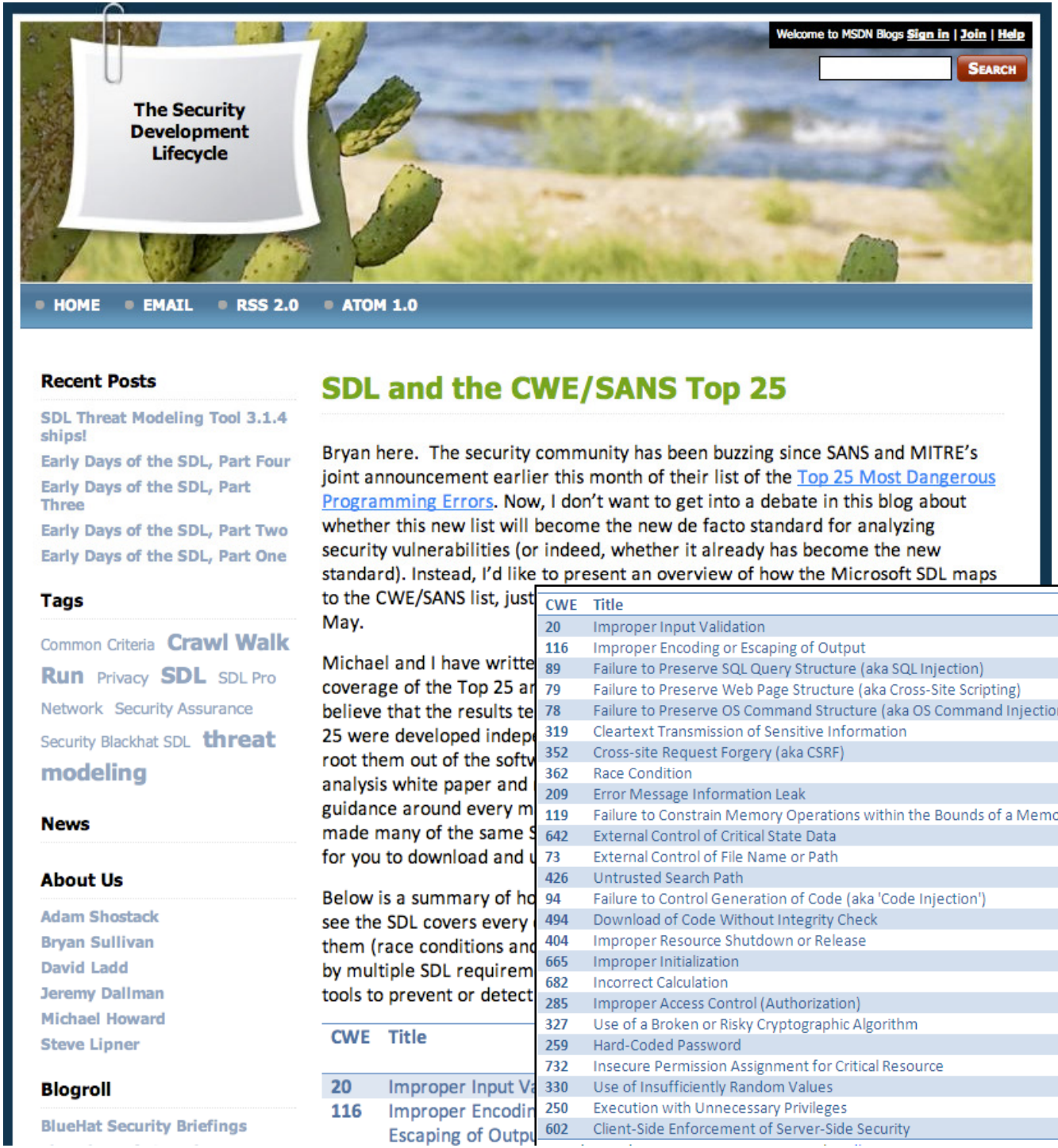
- Specific guidance to developers
- Extensive input from reviewers
- More improvements in CWE 1.3
  - **Mitigation enumeration anyone?**
- Longer term: if you use mitigation X, what issues does it really prevent?
  - **ESAPI**
  - **CERT Coding Standards**

Making Security Measurable™

Top-N Lists and CWE within the BSI Maturity Model (BSIMM)

T1.1	Provide awareness training
T2.1	Offer role-specific advanced curriculum (tools, technology stacks, bug parade)
SFD2.1	Build secure-by-design middleware frameworks/common libraries
SR1.4	Create secure coding standards
CR1.1	create top N bugs list (real data preferred)
CR2.1	use automated tools along with manual review

Making Security Measurable™



CWE	Title	Education?	Manual Process?	Tools?	Threat Model?
20	Improper Input Validation	Y	Y	Y	Y
116	Improper Encoding or Escaping of Output	Y	Y	Y	
89	Failure to Preserve SQL Query Structure (aka SQL Injection)	Y	Y	Y	
79	Failure to Preserve Web Page Structure (aka Cross-Site Scripting)	Y	Y	Y	
78	Failure to Preserve OS Command Structure (aka OS Command Injection)	Y		Y	
319	Cleartext Transmission of Sensitive Information	Y			Y
352	Cross-site Request Forgery (aka CSRF)	Y		Y	
362	Race Condition	Y			
209	Error Message Information Leak	Y	Y	Y	
119	Failure to Constrain Memory Operations within the Bounds of a Memory Buffer	Y	Y	Y	
642	External Control of Critical State Data	Y			Y
73	External Control of File Name or Path	Y	Y	Y	
426	Untrusted Search Path	Y		Y	
94	Failure to Control Generation of Code (aka 'Code Injection')	Y	Y		
494	Download of Code Without Integrity Check				Y
404	Improper Resource Shutdown or Release	Y		Y	
665	Improper Initialization	Y		Y	
682	Incorrect Calculation	Y		Y	
285	Improper Access Control (Authorization)	Y	Y		Y
327	Use of a Broken or Risky Cryptographic Algorithm	Y	Y	Y	
259	Hard-Coded Password	Y	Y	Y	Y
732	Insecure Permission Assignment for Critical Resource	Y	Y		
330	Use of Insufficiently Random Values	Y	Y	Y	
250	Execution with Unnecessary Privileges	Y	Y		Y
602	Client-Side Enforcement of Server-Side Security	Y			Y

Imagine if customers got this kind of data from all their vendors...

Background Details to Check Out

- Contributors
- Process description
- Changelog for each revision
- On the Cusp – weaknesses that almost made it
- Appendices
  - **Selection Criteria and Supporting Fields**
  - **Threat Model for the Skilled, Determined Attacker**

Making Security Measurable™

2009 CWE/SANS Top 25 - Final Draft Changes and Discussion

Changes in Final Week

Date: January 11, 2009

- Improved readability and understandability of discussion text
- Added remainder of suggested mitigations to CWE entries
- Made significant updates to CWE entries on the Top 25, focusing on demonstrative examples, mitigations, consequences, references, and extended descriptions.
- Finished additions to the "On the Cusp" list of CWEs that did not make it to the Top 25
- Created a CWE Top 25 view (CWE-750) and generated supporting PDF graphs for visualization
- Collected final supporting quotes
- Wrote process document
- Finalized contributor list
- Reorganized main document

BACK TO TOP

Summary of Received Comments for Draft 3

Date: January 8, 2009

- We again received many comments from about a dozen people, so we cannot individually respond to them all. Each draft has had approximately 5 to 8 new reviewers.
- Many of the comments were related to specific mitigations for individual entries. The CWE entries are being updated to reflect these changes.
- Some people provided substantive commentary on the threat model, which was a new addition. Much of the feedback centered around apparent contradictions and other issues with the description. As a result, this was cleaned up a bit in this final draft.
- Many contributors made final requests for entries that they thought were important for inclusion. In some cases, there were conflicting recommendations from different people, especially with respect to prevalence. The same entry would be seen extensively by one person but much less frequently by another person. This made the final decisions more difficult. A separate "On the Cusp" document will be published that will cover the issues that did not make it to the final list.

The Top 25 is not…

- A silver bullet
- A guarantee of software health
- A perfect match for your unique needs
- As simple as it seems
- The only thing to include in contract language
- Completely found by tools

Making Security Measurable™

The Top 25 is…

- A mechanism for awareness
- A trigger for deeper questions
- A place for understanding mitigations
- A conversation starter
- An early step on the long road to software assurance

Making Security Measurable™

Contact Us

top25@sans.org

cwe@mitre.org

cwe.mitre.org/top25

Public discussion list coming soon

Making Security Measurable™