



CWE Coverage *Claims* Representation (CCR)

IT Security Automation Conference

October 31st 2011

a success story

682 CWE's defined

29 companies declaring compatibility

of **49** products & services





Coverity Coverage for Common Weakness Enumeration (CWE): Java

CWE ID	Coverity Static Analysis Checker
171	BAD_EQ
262	CHECKED_RETURN
366	GUARDED_BY_VIOLATION
	INDIRECT_GUARDED_BY_VIOLATION
	NON_STATIC_GUARDING_STATIC
	VOLATILE_ATOMCITY
382	DC.CODING_STYLE
	BAD_OVERRIDE
	DC.EXPLICIT_DEPRECATION
	DC.GC
	MUTABLE_COMPARISON
396	MUTABLE_HASHCODE



Coverity Coverage For Common Weakness Enumeration (CWE): C/C++

CWE ID	Coverity Static Analysis Checker	Checker Description	Type of Security Risk
	TAINTED_SCALAR	Use of untrusted scalar value	Alter control flow
		Untrusted value as an argument	
		Use of untrusted value	Arbitrary control of a resource
		Use of untrusted string value	Arbitrary code execution
		User pointer dereference	Arbitrary code execution
		Out-of-bounds access	
		Stray pointer arithmetic	
		COM bad conversion to BSFTR	
		Overflowed array index write	
		Overflowed pointer write	
		Using invalid iterator	
		Iterator container mismatch	
		Splice iterator mismatch	
		Allocation size error	
		Out-of-bounds access	
		Out-of-bounds write	Unauthorized code execution
		Out-of-bounds access	
		Out-of-bounds write	
		Out-of-bounds access	
		Out-of-bounds write	
		Argument cannot be negative	
		Copy into fixed size buffer	
		Destination buffer too small	
		Possible buffer overflow	
		Allocation too small for type	
		Buffer overflow	
		Copy into fixed size buffer	
		Destination buffer too small	
		Unbounded source buffer	

CWE IDs mapped to Klocwork Java issue types

From current

CWE IDs mapped to Klocwork Java issue types

See also Detected Java Issues.

CWE IDs mapped to Klocwork C and C++ issue types/ja

From current

< CWE IDs mapped to Klocwork C and C++ issue types
CWE IDs mapped to Klocwork C and C++ issue types/ja

その他の情報 Detected C and C++ Issues.

CWE ID	説明
20 (http://cwe.mitre.org/data/definitions/20.html)	ABV.TAINTED 未検証入力によるバッファ オーバーフロー SV.TAINTED.GENERIC 未検証文字列データの使用 SV.TAINTED.ALLOC_SIZE メモリ割り当てにおける未検証の整数の使用 SV.TAINTED.CALL_INDEX_ACCESS =関数呼び出しにおける未検証整数の配列インデックスとしての使用
22 (http://cwe.mitre.org/data/definitions/22.html)	SV.CUDS.MISSING_ABSOLUTE_PATH ファイルのロードでの絶対パスの不使用
73 (http://cwe.mitre.org/data/definitions/73.html)	SV.CUDS.MISSING_ABSOLUTE_PATH ファイルのロードでの絶対パスの不使用
74 (http://cwe.mitre.org/data/definitions/74.html)	SV.TAINTED.INJECTION コマンド インジェクション
77 (http://cwe.mitre.org/data/definitions/77.html)	SV.CODE_INJECTION.SHELL_EXEC シェル実行へのコマンド インジェクション
78 (http://cwe.mitre.org/data/definitions/78.html)	NNTS.TAINTED 未検証ユーザ入力があるバッファ オーバーフロー - 非 NULL 終端文字列 SV.TAINTED.INJECTION コマンド インジェクション
88 (http://cwe.mitre.org)	SV.TAINTED.INJECTION コマンド インジェクション NNTS.TAINTED 未検証ユーザ入力があるバッファ オーバーフロー



Cenzic Product Suite is CWE Compatible

Cenzic Hallstorm Enterprise ARC, Cenzic Hallstorm Professional and Cenzic ClickToSecure are compatible with the CWE standard or Common Weakness Enumeration as maintained by Mitre Corporation. Web security assessment results from the Hallstorm product suite are mapped to the relevant CWE ID's providing users with additional information to classify and describe common weaknesses found in Web applications.

For additional details on CWE, please visit: <http://cws.mitre.org/index.html>

The following is a mapping between Cenzic's SmartAttacks and CWE ID's:

	Cenzic SmartAttack Name	CWE ID/s
1	Application Exception	CWE-388: Error Handling
2	Application Exception (WS)	CWE-388: Error Handling
3	Application Path Disclosure	CWE-200: Information Leak (rough match)
4	Authentication Bypass	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection') (rough match)
5	Authorization Boundary	CWE-285: Missing or Inconsistent Access Control, CWE-425: Direct Request ('Forced Browsing')
6	Blind SQL Injection	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
7	Blind SQL Injection (WS)	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
8	Browse HTTP from HTTPS List	CWE-200: Information Leak
9	Brute Force Login	CWE-521: Weak Password Requirements
10	Buffer Overflow	CWE-120: Unbounded Transfer ('Classic Buffer Overflow')
11	Buffer Overflow (WS)	CWE-120: Unbounded Transfer ('Classic Buffer Overflow')
12	Check Basic Auth over HTTP	CWE-200: Information Leak
13	Check HTTP Methods	CWE-650: Trusting HTTP Permission Methods on the Server Side

tool vendors are beginning to advertise coverage

description
goes to native code
tampering action
Working Directory (Stored XSS)
g (Reflected XSS)
g (Stored XSS)
g (Reflected XSS)
g information from the ents
orms: validate method
orms: inconsistent validate
ie Splitting
ex used for array access

a (relatively) simple idea...

lightweight and
define a ^standard way

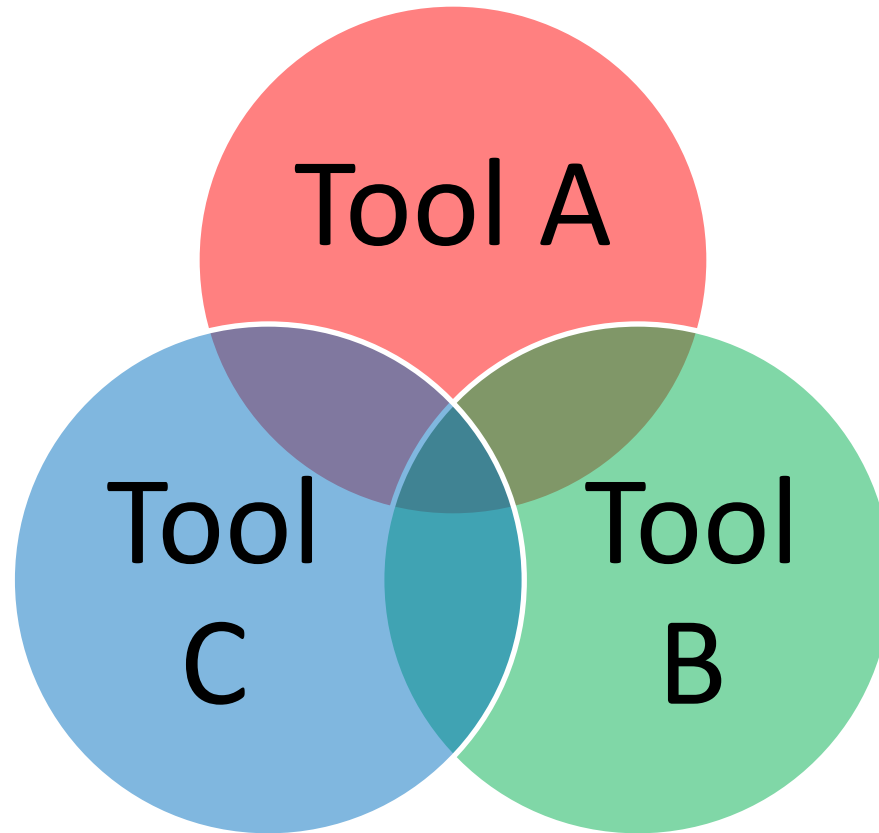
to represent CWE coverage *claims*



some reasons...

why do we need a standard representation?





to make it easy to compute coverage



CWE - CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injectio' - Windows Internet Explorer

http://cwe.mitre.org/data/definitions/89.html

File Edit View Favorites Tools Help Convert Select

McAfee

CWE - CWE-89: Improper Neutralization of Special Ele...

CWE and SANS Institute
TOP 25 MOST DANGEROUS SOFTWARE ERRORS

Home > CWE List > CWE- Individual Dictionary Definition (1.11) Search by ID:

CWE List

- Full Dictionary View
- Development View
- Research View
- Reports

About

- Sources
- Process
- Documents

Community

- Related Activities
- Discussion List
- Research
- CWE/SANS Top 25
- CWSS

News

- Calendar
- Free Newsletter

Compatibility

- Program
- Requirements
- Declarations
- Make a Declaration

Contact Us

- Search the Site

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Weakness ID: 89 (*Weakness Base*) **Status:** Draft

Description

Description Summary

The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

Extended Description

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected and a web site with even a minimal user base is likely to be subject to an attempted attack of this kind. This distinction between the control and data planes.

Time of Introduction

- Architecture and Design
- Implementation
- Operation

Applicable Platforms

Languages

All

Tools Claiming Coverage:

- Zap!
- Code-Nitpicker
- Super-Duper Analyzer

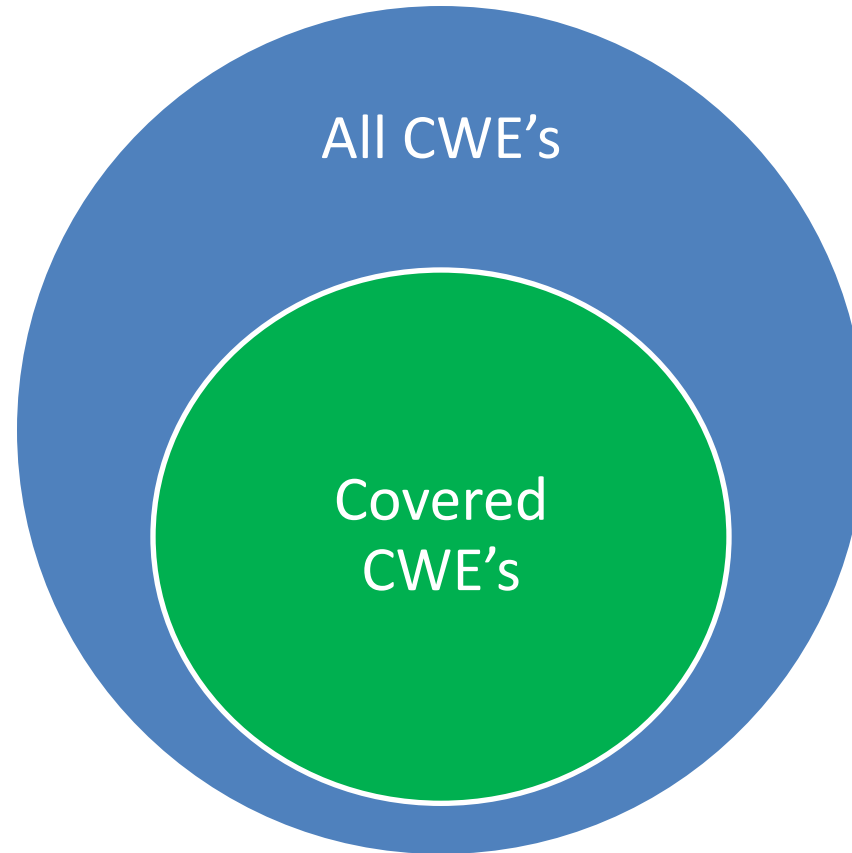
start

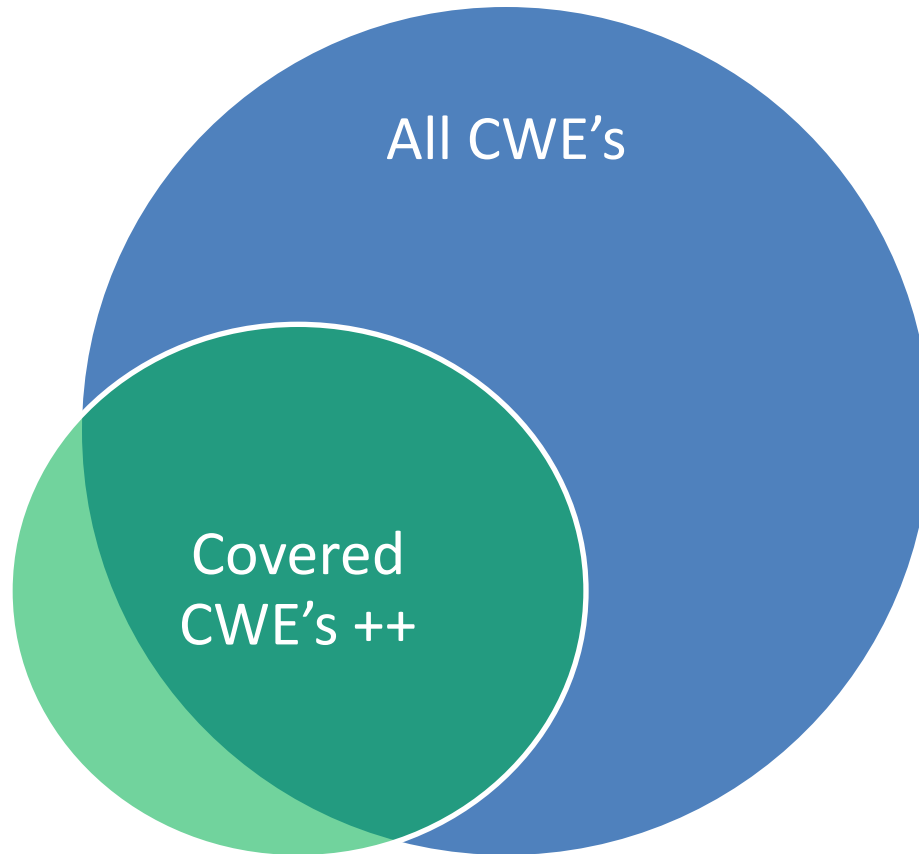
Secure Access SSL VP... Google - Windows Int... CWE - CWE-89: Impr... Inbox - Microsoft Out... Untitled - Message (H... 10:27 AM

help CWE users



to see where R&D might be needed

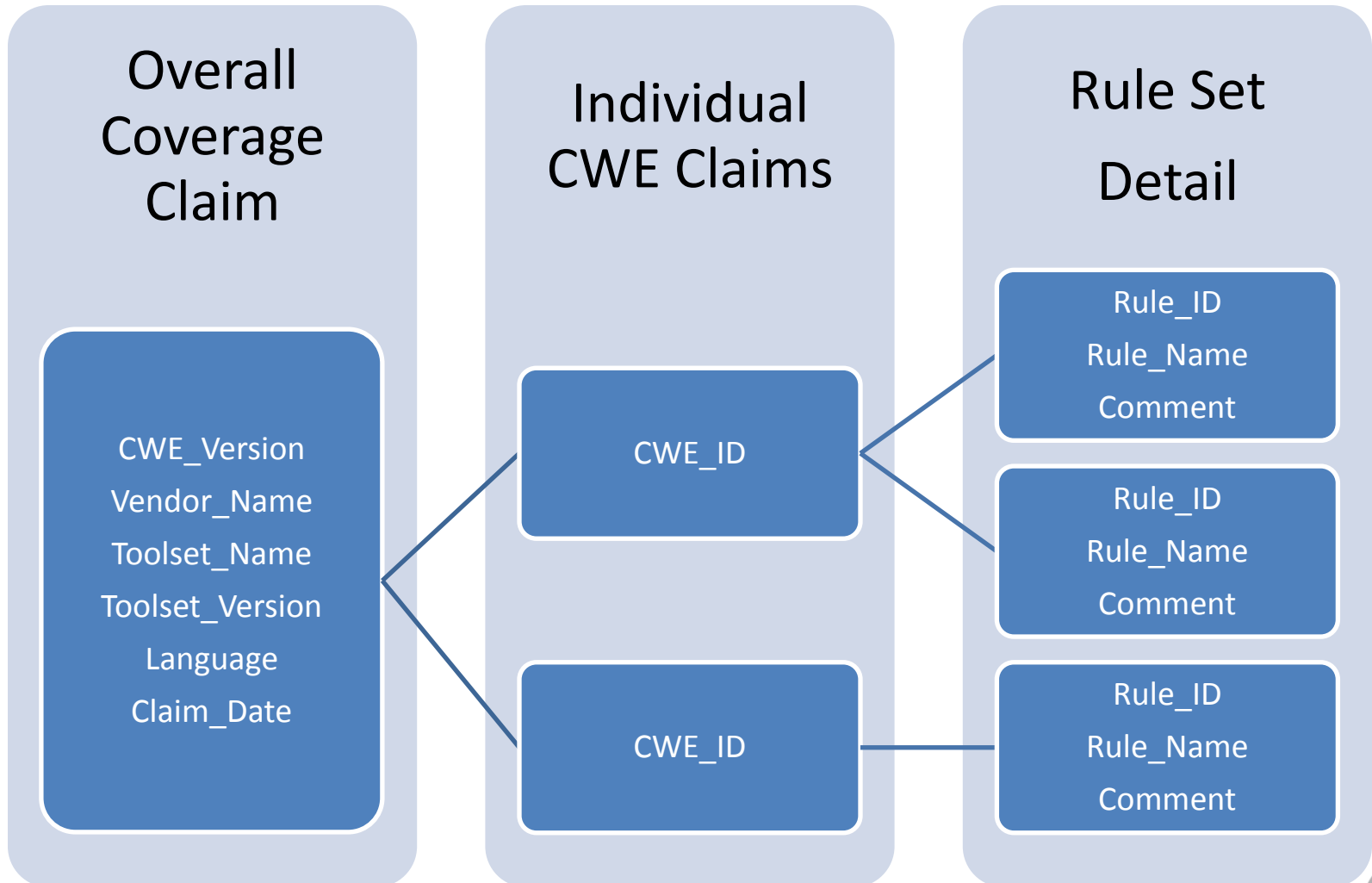


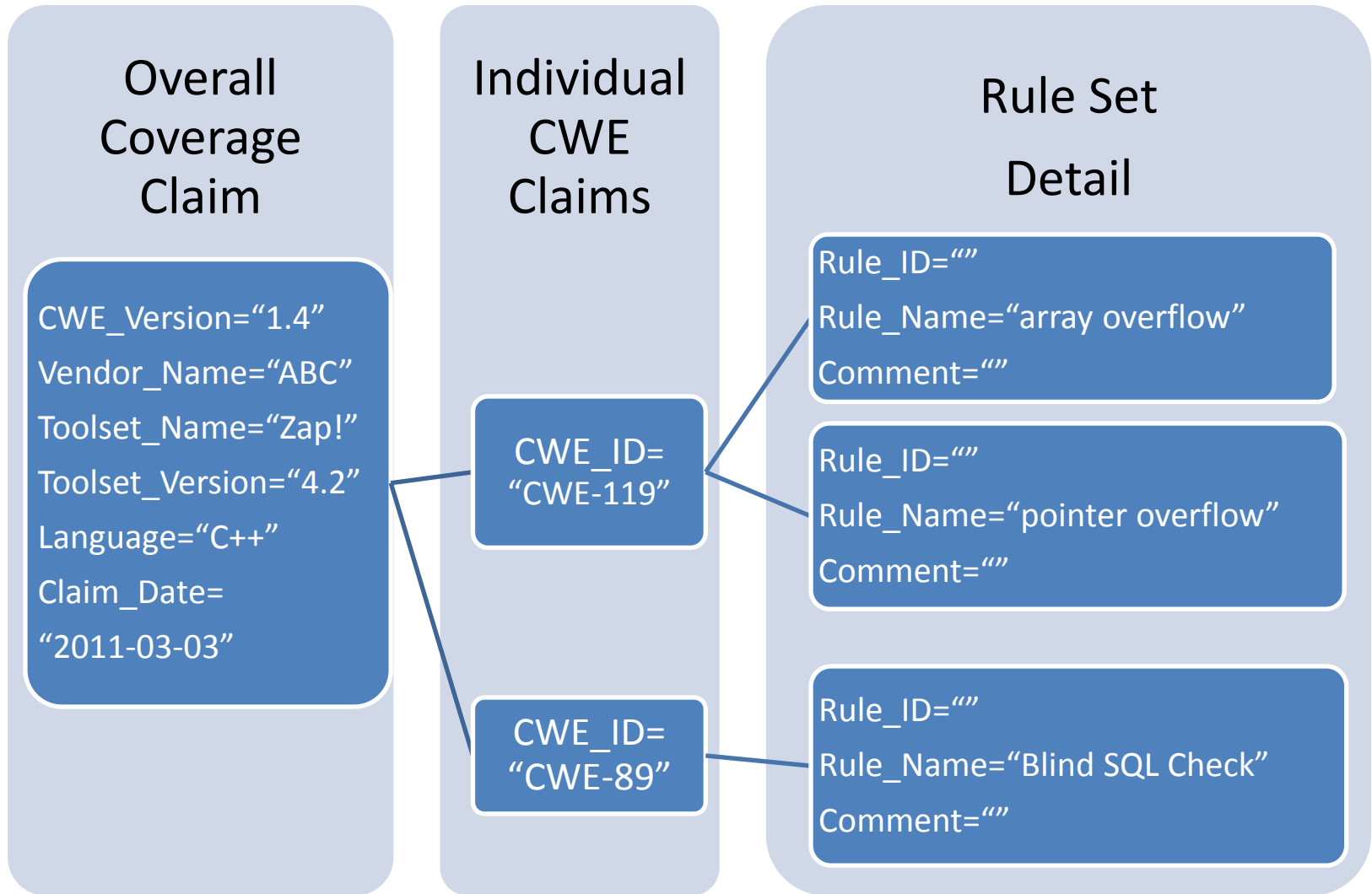


to see where CWE may need to grow



the general idea





something more concrete



services vs. tools

there are many open issues

specificity of claims

CWE compatibility program

disclaimers

dynamic vs. static analysis



we need input from the community

today: starting point for discussion – CCR v0.3

the **action** part



input from users

goals

input from vendors



Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Sample XML file generated by XMLSpy v2011 http://www.altova.com)
-->
<!-- NOTE: this data was created by MITRE, using information published
on the Internet by certain vendors. It is being used to demonstrate
CCR and does not represent any official position by those vendors. -->
<CWE_Coverage_Claims
  xsi:noNamespaceSchemaLocation="CWE_Coverage_Claims_Schema_v
0.2.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <CWE_Coverage_Claim CWE_Version="???" Vendor_Name="Klocwork"
  Toolset_Name="?" Toolset_Version="?" Language_Type="Source Code"
  Language="??" Date_of_Claim="2011-04-01">
```

Example (cont)

<Claims>

<Claim CWE_ID="79" CWE_Name="XSS" Match_Accuracy="Exact">

<CWE_Claim_Comments />

<Rule_Set>

<Rule Rule_ID="SV.XSS.DB" Rule_Name="">

<Rule_Comments />

</Rule>

<Rule Rule_ID="SV.DATA.DB" Rule_Name="">

<Rule_Comments />

</Rule>

<Rule Rule_ID="SV.XSS.REF" Rule_Name="">

<Rule_Comments />

</Rule>

</Rule_Set>

</Claim>

Example (cont)

```
<Claim CWE_ID="352" CWE_Name="CSRF"  
  Match_Accuracy="Not-Covered">
```

```
<CWE_Claim_Comments>It is very difficult for  
static analysis to identify any CSRF issues,  
because each application has its own implicit  
security policy that dictates which requests  
can be influenced by an  
outsider.</CWE_Claim_Comments>
```

```
</Claim>
```

Example (cont)

```
<Claim CWE_ID="738" CWE_Name="Insecure Permissions"  
  Match_Accuracy="CWE-more-abstract">
```

```
  <CWE_Claim_Comments>
```

Checkers such as SV.FIU.PERMISSIONS do provide some coverage, but typically, loose permissions for operations and custom permission models produce too many warnings from static analysis tools.

```
  </CWE_Claim_Comments>
```

```
  <Rule_Set>
```

```
    <Rule Rule_ID="SV.FIU.PERMISSIONS" Rule_Name="">
```

```
      <Rule_Comments />
```

```
    </Rule>
```

```
  </Rule_Set>
```

```
</Claim>
```

Match_Accuracy Element

- **Exact** - The CWE entry exactly covers the same weakness/weaknesses as the given rule set.
- **CWE-more-abstract** - The CWE entry covers more concepts than the given rule set, but there are not any more precise matches available. For example, a rule set might detect resource consumption for a resource that is not specifically covered by CWE.
- **CWE-more-specific** - The CWE entry is more specific than the weakness reported by the given rule set, but the entry's parent(s) are not appropriate matches. This might indicate a difference in perspective between CWE and the capability providing the coverage mapping. It could also include a single rule that covers multiple CWE entries (which might imply that there would be multiple claims for a single rule/rule set).
- **CWE-partial** - The CWE entry is only a partial match with the weakness reported by the given rule set, but the entry is the closest available match.
- **Not-covered** - The CWE entry is not covered by any rule set. The provider is not required to include information about uncovered CWEs. The intention of this assertion is to provide a means for tool vendors to explain why their tool does not claim to discover a certain CWE-defined weakness, if they so choose.
- **No-CWE-available** - There is no CWE entry available that closely matches the weakness reported by the given rule set, but the provider believes that a CWE entry should exist for the reported weakness. The associated CWE_ID should be 0.
- **Not-CWE-applicable** - The rule/rule set is not applicable to CWE, i.e., it is not necessarily about a weakness. This could include rule sets related to coding style conformance, informational messages about the scan, etc. The associated CWE_ID should be -1. The provider is not required to include information about non-applicable rules.
- **Unknown** - The match accuracy is unknown. Typically this would be used by a third party who is creating a coverage claim and does not have insight into the technology.
- **No-claim** – The creator of the CCR document is asserting no claim with respect to this CWE.

Richard.Struse@dhs.gov

thank you.

