CWE Mapping Analysis

Introduction

As CWE grows, it is important to understand how it is used, and if the usage can be affected by the amount and quality of coverage. To attempt to measure coverage, the mapping of individual CWE nodes to issues reported by two commercial analysis tools as well as by an online secure programming reference were analyzed. This analysis of CWE's content structure will help us determine whether the changes incorporated into Version 1.0 of CWE can be leveraged to accurately and cleanly map the content of external repositories of information about security weaknesses. A total sample of 573 items were examined from 3 external collections of information for a cross-check of CWE 1.0's coverage structure. Our goals were to improve the overall quality of CWE, as well as provide a point of reference for individuals and organizations wishing to create content that maps to CWE. This point of reference will hopefully provide some level of guidance in the mapping process, and allow those wishing to give detailed feedback a sense of the vocabulary and issues we have dealt with during the CWE creation and mapping process.

The following questions were asked during the examination of each of the individual mappings:

- Did the choice of CWE node mapping make sense? Was it accurate?
- If no CWE node was mapped to the issue, can an accurate mapping be made?
- Were there any nodes missing from CWE or that need to be corrected?
- If the CWE mapped to an issue was deemed incorrect, what can we do to correct it? Was the CWE node too vague, too general, or just not quite a good fit?

If CWE nodes are too vague or general – or missing entirely – this would suggest areas for improvement. If areas of concern begin to trend, as in the same overall issues or nodes tend to correlate, this could suggest either a need for further clarity and depth in CWE, or even too much depth, in the CWE nodes.

ABSTRACTION	85	14.83%
BAD ENTRY	6	1.05%
CONFUSING	6	1.05%
EXACT	262	45.72%
IMPRECISE	40	6.98%
INCLUSION	57	9.95%
MISSING	58	10.12%
NEEDS DETAILS	25	4.36%
OTHER	7	1.22%
PERSPECTIVE	27	4.71%
Total	573	

In the table, the 573 individual items were categorized into 10 different categories. The names of the categories are for the most part self-explanatory, but they will be covered individually.

It should be noted that while the three resources and their mapping examples were very different, they all were able to be categorized into these 10 categories quite easily. And while the resources varied in scope as far as reporting of individual issues, the depth of each was deep enough to allow for thorough categorization.

Analysis

While the individual categories are listed in the table in alphabetical order, they will be covered in an order of simplest to most complex.

EXACT – The majority of issues covered had an exact match with a CWE node. There were some cases where a vendor tool had no CWE node mapping or even the wrong CWE node mapping (possibly due to referring to an older version or CWE, typo, overlooking an available node, or selecting the wrong node when there was an exact node that covered the issue), but since the issue being described and there was a CWE entry that was conceptually the same, the categorization would be listed as exact.

BAD ENTRY – This is in reference to what we would call a bad entry in CWE, not in the resource mapping to it. During the mapping process, the CWE node seemed to fit, however a generality or vagueness about the CWE node could provide confusion or lead to incorrect mappings in the future. Nodes like this were flagged for editing and revision. A good example of this is CWE-391, which states “Ignoring exceptions and other error conditions may allow an attacker to induce unexpected behavior unnoticed.” In other cases, a vendor mapping was made to two nodes in an attempt to cover what was perceived as one issue. This would indicate the need for a more thorough node that covered the issue, as opposed to having to use multiple nodes.

MISSING: Some of the reported issues were indeed issues CWE should have a node for, but there was either no existing node or the node was “immature” and required expansion to fully cover the issue. A good example of this are issues pertaining to logging. There are multiple issues related to logging that need further expansion inside of CWE to turn them into mature nodes.

NEEDS DETAILS: In the reported issues, there were a few that were either extremely complex in nature or contained vague references to important aspects. To fully understand the issues would require further investigation with the resource to determine the original intent. The resources may or may not have had CWE entries, but it was indeterminate if the mappings would be accurate, or even if new mapping might be needed.

CONFUSING: There were a small group of reported issues that had either no CWE reference or a seemingly odd choice of CWE entry or entries. Similar to the NEEDS DETAILS category, these will require resource input to resolve as to whether the mappings are correct, and if new nodes are needed or existing nodes with suffice.

OTHER: This was a “catch all” category for those issues that could not be categorized, at least into one category. There were two groups within the OTHER category. The first one involved issues that had multiple categorization characteristics, which will ultimately require further research as to how to approach them. The second group was rather specific in that it involved issues regarding the use of LDAP access control via anonymous binds. One could argue that the issue could be handled programmatically by eliminating the anonymous bind in the source code and require credentials; however it is also possible to mitigate the issue by configuring the LDAP server to disallow anonymous binds.

IMPRECISE: The resource reports on an issue and it is clearly defined, but the CWE node mapping choice is not an exact match. The resource may have made the mapping choice or it could have been unmapped; however during the analysis of what the CWE node was, it was determined not to be a precise fit. The issue may cover multiple distinct concepts or the issue maps to one element in a chain.

For example, a resource reports an issue with usage of Trace output in ASP.NET applications, by setting the Trace attribute of the `<page>` directive to true. While perfectly acceptable in a development or test environment, in a production environment this could be leveraged by an attacker to gain additional insight as to how the application works and interacts with other components. Currently there is no exact match in CWE, although CWE-215 Information Leak Through Debug Information (“The application contains debugging code that can leak sensitive information to untrusted parties.”) and CWE-11 ASP.NET Misconfiguration: Creating Debug Binary (“Debugging messages help attackers learn about the system and plan a form of attack.”) are close. However CWE-215 is more commonly referring to the use of extra code that provides data output useful during development and testing (e.g. a section of C source code enclosed with `#ifdef DEBUG` and `#endif`), and CWE-11 is referring to ASP.NET-built debug binaries. Neither is an exact fit, both are close, but the mapping does not match up.

In a different resource example, there is a stated issue involving an attempt to override a common Java or dotnet method name, however the attempt spelled incorrectly or the argument list causes it to not override the intended method. The chosen mapping was CWE-628 Function Call with Incorrectly Specified Arguments (“The product calls a function, procedure, or routine with arguments that are not correctly specified, leading to always-incorrect behavior and resultant weaknesses.”) which is close, but not an exact match. In the details in CWE-628, it states there are five ways to introduce this weakness, including the wrong type of variable or weakness as well as the wrong value. Dealing with the argument list is covered, but spelling errors are not. A misspelled method name could be considered a “wrong value”, but again it is not precise. When examining the 5 children of CWE-628 (CWE-683, CWE-685, CWE-686, CWE-687, and CWE-688) which correspond to CWE-628’s five ways to introduce this weakness, there is no exact match on the spelling part of the issue, and multiple matches on the argument list part of the issue.

Yet another example of an imprecise mapping occurs when mapping an issue related to signed comparison. The issue involves using a signed comparison to check a value that is later treated as unsigned, causing the program to read data from outside the bounds of allocated memory. This

issue was mapped to CWE-126 Buffer Overread ("The software reads data past the end of the intended buffer.") is close, but the issue specifically involves the signed comparison.

INCLUSION: Inclusion issues are ones that cover items that probably should not be in CWE. These are related to issues with non-security impacts (code quality, environment and build issues, configuration issues) and ones with security impacts where the issue referenced is more of a result than a root cause.

Many of the vendor tools will report back issues related to the tool itself (e.g. unable to open the software.c file) so there is no reason to consider those for CWE mapping. As the mapping effort involved all reported output from the resource, these were noted as items with an inclusion issue.

A code quality issue includes things such as redundant or useless assignment of a variable. Assignment of a value to a variable and then never using the variable might be one example. Another example might be the initialization of a variable (`int var = 0;`) followed by an explicit assignment using the same value (`var = 0;`). There is no actual security impact, however many resources – particularly software analysis tool vendors – will cover general code quality issues in addition to security-related issues.

None of the reported issues by these software vendors which were deemed to be inclusion issues actually had a CWE ID mapped to those issues. However as most vendors in the space are looking for security-related impacts, it does open up the idea to debate. In essence, all of the security-related issues are code quality issues at their core, so the discovery of code quality issues in general could be considered indicative of potential issues with security implications that could exist elsewhere in the code.

Another inclusion issue involves the possibility of a missing node. If the issue is covered, but it is determined that it could be possible to create a new node to cover it in greater detail, should the new node be created? A good example of this is found in the examination of one of the ideas behind mitigating CWE-415 Double Free and CWE-416 Use After Free. The idea is to assign NULL to a pointer after it has been freed, eliminating references to that memory location. The inclusion issue becomes apparent when asking if CWE should include the lack of this mitigation technique as a separate node. This would be a node that states No NULL Assignment of a Pointer After Free. In some cases, resources that reference CWE-415 and CWE-416 are checking to ensure the mitigation technique is used as opposed to something that clearly falls under these two nodes.

PERSPECTIVE: Perspective issues are exactly that – they depend on what perspective one has as to which CWE node mapping makes the most sense to them. The resources typically had a CWE node mapping, however alternate choices of node mappings could have been made, and still seem correct.

Some issues fall into seemingly different sections of CWE. For example, a Double Unlock on a software component could fall under CWE-404 Improper Resource Shutdown or Release ("The program fails to release – or incorrectly releases – a system resource before it is made available for re-use."), CWE-413 Insufficient Resource Locking ("A product does not sufficiently lock

resources, in a way that either (1) allows an attacker to simultaneously access those resources, or (2) causes other errors that lead to a resultant weakness."), or CWE-675 Duplicate Operations on Resource ("The product performs the same operation on a resource two or more times, when the operation should only be applied once."). One could choose CWE-411 Resource Locking Problems, which is in fact a category node, and is a parent node to CWE-413. Which mapping is correct? They all are in one way or another – all of them fit, depending on the perspective of the individual reviewing the information at hand.

Another issue is Missing Return Statement. Three possible nodes for mapping are CWE-573 Failure to Follow Specification ("The software fails to follow the specifications for the implementation language, environment, framework, protocol, or platform."), CWE-398 Indicator of Poor Code Quality ("The code has features that do not directly introduce a weakness or vulnerability, but indicate that the product has not been carefully developed or maintained."), and CWE-691 Insufficient Control Flow Management ("The code does not sufficiently manage its control flow during execution, creating conditions in which the control flow can be modified in unexpected ways."). All are valid, all seem to be a good individual match, yet a mapping to one does not seem correct in the face of the existence of the others.

A final example would be a case where a resource's mapping is to one element of a chain when another element of the same chain could be appropriate. For example take the issue of a Missing Check Against NULL. One choice could be CWE-252 Unchecked Return Value ("Ignoring a method's return value can cause the program to overlook unexpected states and conditions.") when CWE-690 Unchecked Return Value to NULL Pointer Dereference ("The product does not check for an error after calling a function that can return with a NULL pointer if the function fails, which leads to a NULL pointer dereference.") might be a good mapping as well.

With perspective issues the act of mapping individual nodes to issues becomes rather complex. This area generated more discussion and changing (and rechanging) of choices for a node mapping than any other group.

ABSTRACTION: An abstraction issue is one where there is a logical choice of node based upon a process of elimination; however the issue itself takes the concept further than the original node reference.

One resource had a check that looks for a floating point format string issue. Mapping to CWE-134 Uncontrolled Format String ("The software uses externally-controlled format strings in printf-style functions, which can lead to buffer overflows or data representation problems.") is clearly correct, however the node entry's overall flavor centers around the usage of %h and %n, whereas the %f and %F issues are overlooked.

Another common abstraction problem is a resource-specific variant. In handling three SSL-specific exceptions in Java, a mapping to CWE-391 Unchecked Error Condition ("Ignoring exceptions and other error conditions may allow an attacker to induce unexpected behavior unnoticed.") makes sense, however the vendor is checking for a specific variant related to specific errors implemented under Java.

Conclusion

When mapping issues with CWE nodes, things are not always as clear cut as they might appear. Extra analysis into the mappings by content creators can lead to improved and more accurate node mappings, in addition to the editing and creation of new nodes. It could even lead to improved content, as details which help clarify the issue being mapped could be added. The model we have outlined for this mapping analysis was originally designed as a quality control exercise for CWE; however through questions asked (from the introduction), categorization of the node mappings, and analysis of the categorization, we feel this can benefit others as well.

The main concern is resolution – how do we reconcile issues which are not an exact match? Resolution will come in the form of CWE node alteration, content alteration, or a combination of the two. Other alternatives might include multiple node mappings for a single issue, although this is not always recommended. But this analysis does provide a common ground for discussion of potential resolution points, and should help all parties involved in understanding the complexity of mapping issues.