A Status Update: The Common Weaknesses Enumeration

Robert A. Martin
MITRE Corporation
202 Burlington Road
Bedford, MA 01730
1-781-271-3001
ramartin@mitre.org

Sean Barnum
Cigital, Inc.
21351 Ridgetop Circle, Suite 400
Sterling, VA 20166
1-703-404-5762
sbarnum@cigital.com

ABSTRACT

This paper is a status update on the Common Weaknesses Enumeration (CWE) initiative, one of the efforts focused on improving the utility and effectiveness of code-based security assessment technology. It is hoped that the CWE initiative will help to dramatically accelerate the use of tool-based assurance arguments in reviewing software systems for security issues.

1. INTRODUCTION

More and more organizations want assurance that the software products they acquire and develop are free of known types of security weaknesses. High quality tools and services for finding security weaknesses in code are new. The question of which tool/service is appropriate/better for a particular job is hard to answer given the lack of structure and definition in the software product assessment industry.

There are several efforts currently ongoing to begin to resolve some of these shortcomings including the Department of Homeland Security (DHS) National Cyber Security Division (NCSD) sponsored Software Assurance Metrics and Tool Evaluation (SAMATE) project [1] being led by the National Institute of Standards and Technology (NIST), and the Object Management Group (OMG) Software Assurance (SwA) Special Interest Group (SIG) [2], among others. While these efforts are well placed, timely in their objectives and will surely yield high value in the end, they both require a common description of the underlying security weaknesses that can lead to exploitable vulnerabilities in software that they are targeted to resolve. Without such a common description, many of these efforts cannot move forward in a meaningful fashion or be aligned and integrated with each other to provide strategic value.

As part of their participation in the SAMATE project, MITRE has helped lead the creation of a community of partners from industry, academia, and government to develop, review, use, and support a common weaknesses dictionary/encyclopedia that can be used by those looking for weaknesses in code, design, or architecture as well as those teaching and training software developers about the code, design, or architecture weaknesses that they should avoid due to the security problems they can have on applications, systems, and networks.

2. FIRST STEPS

The initial steps of the CWE work entailed collecting and reviewing past efforts in organizing and itemizing security weaknesses and identifying those concepts, constructs and lessons that could be used to create the CWE dictionary. Lauren Davis, from the Johns Hopkins University Applied Physics Laboratory, facilitated this work. At the same time we started establishing the foundations of a web site design to hold the materials, ideas, and documents that would come out of the CWE initiative. An important element of the CWE initiative is to be transparent to all on what we are doing, how we are doing it, and what we used to develop the CWE List. We believe this transparency is important both during the initial creation of the CWE List so that all of the participants in the CWE Community will feel comfortable with the end result and won't be hesitant about incorporating CWE into what they do. However, the transparency must also include those that will come after the CWE creation activities are complete and should be provided the opportunity to review and learn about how the CWE List was created. To this end we will be making sure that copies of all of the source documents of publicly available information used in creating CWE List are available on the web site [3].

3. PRIMING THE PUMP

To start the creation of the CWE List we brought together as much public content as possible, using three primary sources:

- the Preliminary List of Vulnerability Examples for Researchers (PLOVER) collection [4] which identified over 300 weakness types created by determining the root issues behind 1,400 of the vulnerabilities in Common Vulnerabilities and Exposures (CVE) List [5];
- the Comprehensive, Lightweight Application Security Process (CLASP) from Secure Software. which yielded over 90 weakness concepts [6], and
- the issues contained in Fortify's Seven Pernicious Kingdoms papers, which contributed over 110 weakness concepts [7]

Working from these collections as well as those contained in the other thirteen information sources listed on the CWE web site "Sources" page we developed the current draft of the CWE List, which entails almost 500 separate weaknesses.

The CWE List content is provided in several formats and will have additional formats and views into its contents added as the CWE initiative proceeds. Currently one pane of the main CWE page contains an expanding/contracting hierarchical "taxonometric" view along with an alphabetic dictionary pane. The end items in the hierarchical view are hyper-linked to their respective dictionary entries in the second pane. Graphical depictions of CWE content, as well as the contributing sources, are also available on the site. Finally, the xml and xsd for the CWE List are provided for those who wish to do their own analysis/review with other tools. Dot notation representations of this material will be added in the future.

4. EXPANDING CWE

With the current draft of CWE List as a baseline/reference point, we are now gathering in the specific details and descriptions of 13 organizations that have agreed to contribute their intellectual property to the CWE initative. Under Non-Disclosure Agreements with MITRE, which allow the merged collection of their individual contributions to be publicly shared in the CWE List, Cenzec, Core Security, Coverity, Fortify, Interoperability Clearinghouse, Klocwork, Ounce Labs, Parasoft, proServices Corporation, Secure Software, SPI Dynamics, Veracode, and Watchfire are all contributing.

In addition to these sources, we will also leverage the work, ideas, and contributions of researchers at Carnegie Mellon's CERT/CC, IBM, KDM Analytics, Kestrel Technology, MIT Lincoln Labs, North Carolina State University, Oracle, the Open Web Application Security Project (OWASP), Security Institute, UNISYS, the Web Application Security Consortium (WASC), Whitehat Security, and any other interested parties that wish to contribute.

We expect the merging and combining of the contributed materials will take most of the summer and result in an updated CWE List that will be ready for community comments and refinement as we move forward. A major part of this will be refining and defining the required attributes of CWE elements into a more formal schema defining the metadata structure necessary to support the various uses of CWE List. This schema will also be driven by our need to align with and support the SAMATE and OMG SwA SIG efforts that are developing software metrics, software security tool metrics, the software security tool survey, the methodology for validating software security tool claims, and the reference datasets.

5. CURRENT THOUGHTS ON IMPACT AND TRANSITION OPPORTUNITIES

As stated in the concept paper that laid out the case for developing the CWE List [8], the completion of this effort will yield consequences of three types: direct impact and value, alignment with and support of other existing efforts, and enabling of new follow-on efforts to provide value that is not currently being pursued.

Following is a list of the direct impacts this effort will yield. Each impact could be the topic of much deeper and ongoing discussion.

1. Provide a common language of discourse for discussing, finding and dealing with the causes of software security vulnerabilities as they are manifested in code, design, or architecture.
2. Allow software security tool vendors and service providers to make clear and consistent claims of the security weaknesses that they cover to their potential user communities in terms of the CWEs that they look for in a particular code language. Additionally, a new "CWE Compatibility" will be developed to allow security tool and service providers to publicly declare their capability's coverage of CWEs.
3. Allow purchasers to compare, evaluate and select software security tools and services that are most appropriate to their needs including having some level of assurance of the level of CWEs that a given tool would find. Software purchasers would be able to compare coverage of tool and service offerings against the list of CWEs and the programming languages that are used in the software they are acquiring.
4. Enable the verification of coverage claims made by software security tool vendors and service providers (this is supported through CWE metadata and alignment with the SAMATE reference dataset).
5. Enable government and industry to leverage this standardization in the contractual terms and conditions.

Following is a list of alignment opportunities with existing efforts that are provided by the results of this effort. Again, each of these items could be the topic of much deeper ongoing discussion.

1. Mapping of CWEs to CVEs. This mapping will help bridge the gap between the potential sources of vulnerabilities and examples of their observed instances providing concrete information for better understanding the CWEs and providing some validation of the CWEs themselves.
2. Bidirectional alignment between the common weaknesses enumeration and the SAMATE metrics effort.
3. Any tool/service capability measurement framework that uses the tests provided by the SAMATE Reference Dataset would be able to leverage this common weakness dictionary as the core layer of the framework. This framework effort is not an explicitly called out item in the SAMATE charter but is implied as necessary to meet the project's other objectives.
4. The SAMATE software security tool and services survey effort would be able to leverage this common weaknesses dictionary as part of the capability framework to effectively and unambiguously describe various tools and services in a consistent apples-to-apples fashion.
5. There should be bidirectional alignment between this source of common weaknesses and the SAMATE reference dataset effort such that CWEs could reference supporting reference dataset entries as code examples of that particular CWE for explanatory purposes and reference dataset entries could reference the associated CWEs that they are intended to demonstrate for validation purposes. Further, by working with industry, an appropriate method could be developed for collecting, abstracting, and sharing code samples from the code of the products that the CVE names are assigned to with the goal of gathering these code samples from industry researchers and academia so that they could be shared as part of the reference dataset and aligned with the vulnerability taxonomy. These samples would then be available as tailoring and enhancement aides to the developers of software assessment security tools. We could actively engage closed source and open source development organizations that work with the CVE initiative to assign CVE names to vulnerabilities to identify an approach that would protect the source of the samples while still allowing us to share them with others. By using the CVE-based relationships with these organizations, we should be able to create a high-quality collection of samples while also improving the accuracy of the software product security assessment tools that are available to the software development groups to use in vetting their own product's code.
6. Any validation framework for tool/service vendor claims, whether used by the purchasers themselves or through a 3rd

party validation service, would rely heavily on this common weakness dictionary as its basis of analysis. To support this, we would work with researchers to define the mechanisms used to exploit the various CWEs for the purposes of helping to clarify the CWE groupings and as a possible verification method for validating the effectiveness of the tools that identify the presence of CWEs in code by exploring the use of several testing approaches on the executable version of the reviewed code. The effectiveness of these test approaches could be explored with the goal of identifying a method or methods that are effective and economical to apply to the validation process.

7. Bidirectional mapping between CWEs and Coding Rules, such as those deployed as part of the DHS NCSD "Build Security In" (BSI) website [9], used by tools and in manual code inspections to identify common weaknesses in software.

8. Leveraging of the OMG technologies to articulate formal, machine parsable definitions of the CWEs to support analysis of applications within the OMG standards-based tools and models.

Following is a list of new, unpursued follow-on opportunities for creating added value to the software security industry.

1. Expansion of the Coding Rules Catalog on the DHS BSI website to include full mapping against the CWEs for all relevant technical domains.

2. Identification and definition of specific domains (language, platform, functionality, etc.) and relevant protection profiles based on coverage of CWEs. These domains and profiles could provide a valuable tool to security testing strategy and planning efforts.

With this fairly quick research and refinement effort, this work should be able to help shape and mature this new code security assessment industry, and dramatically accelerate the use and utility of these capabilities for organizations and the software systems they acquire, develop, and use.

6. ACKNOWLEDGMENTS

The work contained in this paper was funded by DHS NCSD.

7. REFERENCES

[1] "The Software Assurance Metrics and Tool Evaluation (SAMATE) project," National Institute of Science and Technology (NIST), (http://samate.nist.gov).

[2] "The OMG Software Assurance (SwA) Special Interest Group," (http://swa.omg.org).

[3] "The Common Weaknesses Enumeration (CWE) Initiative," MITRE Corporation, (http://cve.mitre.org/cwe/).

[4] "The Preliminary List Of Vulnerability Examples for Researchers (PLOVER)," MITRE Corporation, (http://cve.mitre.org/docs/plover/).

[5] "The Common Vulnerabilities and Exposures (CVE) Initiative," MITRE Corporation, (http://cve.mitre.org).

[6] Viega, J., The CLASP Application Security Process, Secure Software, Inc., http://www.securesoftware.com, 2005.

[7] McGraw, G., Chess, B., Tsipenyuk, K., "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". "NIST Workshop on Software Security Assurance Tools, Techniques, and Metrics," November, 2005 Long Beach, CA.

[8] Martin, R. A., Christey, S., Jarzombek, J., "The Case for Common Flaw Enumeration". "NIST Workshop on Software Security Assurance Tools, Techniques, and Metrics," November, 2005 Long Beach, CA.

[9] Department of Homeland Security National Cyber Security Division's "Build Security In" (BSI) web site, (http://buildsecurityin.us-cert.gov).