Vulnerability and Remediation for a High-assurance Web-based Enterprise

William R. Simpson and Coimbatore Chandersekaran
Institute for Defense Analyses, 4850 Mark Center Dr., Alexandria, Virginia 22311, U.S.A.

Keywords: Threat Mitigation, Vulnerability, Penetration Testing, Flaw Remediation.

Abstract: A process for fielding vulnerability free software in the enterprise is discussed. This process involves testing for known vulnerabilities, generic penetration testing and threat specific testing coupled with a strong flaw remediation process. The testing may be done by the software developer or certified testing laboratories. The goal is to mitigate all known vulnerabilities and exploits, and to be responsive in mitigating new vulnerabilities and/or exploits as they are discovered. The analyses are reviewed when new or additional threats are reviewed and prioritized with mitigation through the flaw remediation process, changes to the operational environment or the addition of additional controls or products). This process is derived from The Common Criteria for Information Technology Security Evaluation, Common Evaluation Methodology which covers both discovery and remediation. The process has been modified for the USAF enterprise.

1 INTRODUCTION

The sheer volume of regulations and analyses to reduce cyberspace threats is astounding US Department of Defense, 2012a,b, NIST, 2009, Common Criteria 2009, Wassermann , 2007, Livshits , 2008, Kiezun, , 2009, Jovanovic,2006, Huang, 2004, Kals, 2006, Mcallister, 2008, Maggi, 2009, are just a few. Threat mitigation is undertaken to reduce the attack space and to minimize losses due to cyber activities either malicious or accidental. From Hacks to Nation-States, these threats continue to attempt to penetrate networks every day. These threats are growing, evolving, and sophisticated. Loss of information capability and information integrity as well as the loss of intellectual property is a significant security risk.

The goal of this paper is to describe a process to mitigate many known vulnerabilities and to be responsive in mitigating new or newly discovered vulnerabilities. Vulnerability identification is a continuous process whereby threats are reviewed and prioritized with mitigation through either the flaw remediation process, modifications to the operating environment, or the reliance of multiple products in mutual support of one another. The analyses are reviewed either periodically or on demand as new threats evolve or are identified.

2 VULNERABILITY CAUSES

Causes of vulnerabilities are numerous. The Common Weakness Enumeration (CWE), Mitre 2013b, was developed to track causes of vulnerabilities as a guide to their elimination. Vulnerabilities are of concern because they may be turned into exploits that in turn can be used to disrupt IT systems or ex-filtrate their assets. Figure 1 shows some of the more common causes amongst non-website software as derived from Mitre 2013b.

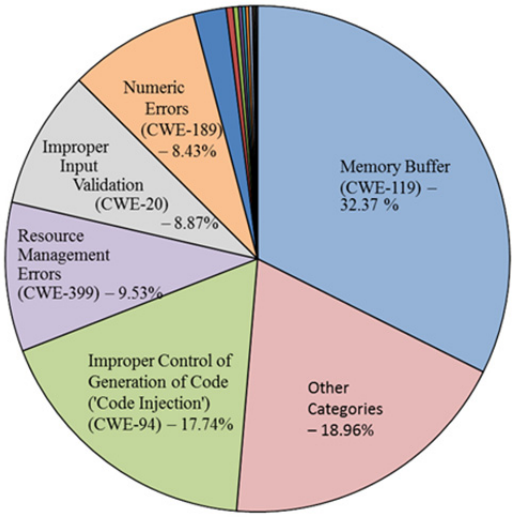

Figure 1: Vulnerabilities in Non-Website Software.

R. Simpson W. and Chandersekaran C..
Vulnerability and Remediation for a High-assurance Web-based Enterprise.
DOI: 10.5220/0004760501190128
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 119-128
ISBN: 978-989-758-028-4

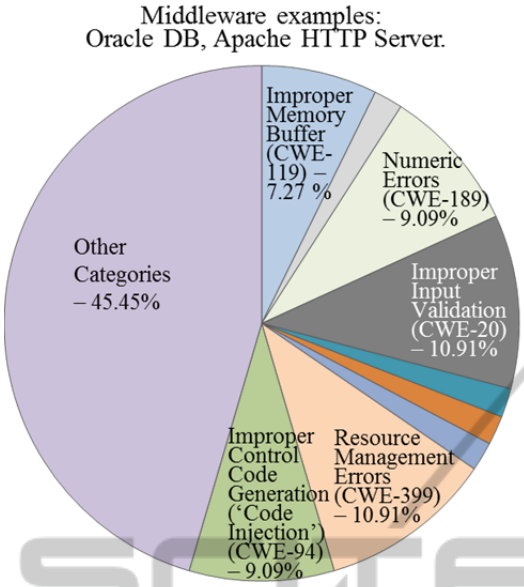

Figure 2: Vulnerabilities in Middleware.

Figure 2 shows the more common causes of vulnerability among middleware software elements as derived from Mitre 2013b.

The identification of vulnerabilities as they relate to software allows software development practices and tools to look for structures and processes in software that may be exploited. Many of these tools will check for susceptibility to specific exploits also. Proper software development practices will eliminate many of these vulnerabilities. Formal design practices offer additional capabilities to avoid buffer overflows, improper data acceptance and other software flaw based vulnerabilities, Jones 2010. Nonetheless tools must be used to verify that they have been properly done, or remediation must be applied. These tools are not perfect, and the analysis may miss some vulnerabilities. There use should be followed by penetration testing. Even then, other security measures are needed within the enterprise. Those additional measures will not be discussed in this paper. These analyses, however, will reduce the attack space and eliminate some common attack methodologies for the enterprise.

3 RELATED WORK

Vulnerability analyses are usually coupled with software quality assurance and fall into three main categories:

1. Static code analysis
2. Dynamic analysis or execution tracing
3. Penetration testing

3.1 Static Code Analysis

Static program analysis is the analysis of computer software that is performed without actually executing programs. In most cases the analysis is performed on some version of the source code and in the other cases some form of the object code. The term is usually applied to the analysis performed by an automated tool, with human analysis being called program understanding, program comprehension or code review. The sophistication of the analysis performed by tools is rule-driven which provides the intellectual property associated with the tool set. A growing commercial use of static analysis is in the verification of properties of software used in safety-critical computer systems and locating potentially vulnerable code.

These types of analyses are discussed in (Jones, 2010), (Livshits, 2006), and (Wichmann, 1995).

3.2 Dynamic Code Analysis

Dynamic program analysis is the analysis of computer software that is performed by executing programs on a real or virtual processor. For dynamic program analysis to be effective, the target program must be executed with sufficient test inputs to produce interesting behavior. Care must be taken to minimize the effect that instrumentation has on the execution (including temporal properties) of the target program. The sophistication of the analysis performed by tools is both rule-driven and instrumentation (debug monitors) which provides the intellectual property associated with the tool set. Some dynamic code checker tools are given below:

- HP Security Suite is a suite of Tools at various stages of development. QAInspect and WebInspect are generally considered Dynamic Analysis Tools, while DevInspect is considered a static code analysis tool, (HP 2013).
- IBM Rational AppScan is a suite of application security solutions targeted for different stages of the development lifecycle. The suite includes two main dynamic analysis products – (IBM 2013).
- Intel Thread Checker is a runtime threading error analysis tool which can detect potential data races and deadlocks in multithreaded Windows or Linux applications, (Intel 2013).

3.3 Penetration Testing

A penetration test is a method of evaluating computer and network security by simulating an attack on a computer system or network from

external and internal threats. The process involves an active analysis of the system for any potential vulnerabilities that could result from poor or improper system configuration. The analyses include both known and unknown hardware or software flaws, or operational weaknesses in process or technical countermeasures. This analysis is carried out from the position of a potential attacker using a threat model, and can involve active exploitation of security vulnerabilities.

(Mosaic, 2013) has captured a number of tools (including The Penetrator, SAINTexploit™, Metasploit Pro, Core WebVerify™, CORE INSIGHT™ Enterprise, CORE IMPACT® Pro, Core CloudInspect and others) that will perform such exploitation with pros and cons.

In the analysis of requirements the penetration testing and both static and dynamic analyses to setup the penetration testing are required. Further, the work is only begun with these analyses as the discovery of vulnerabilities and exploits once the software is fielded must also be submitted to the flaw remediation system and resolved as quickly as possible. To get ahead of this cycle many software vendors now offer rewards for vulnerability and exploit discovery (Finifter, 2013).

4 VULNERABILITY ANALYSIS

The vulnerability analysis requirement is levied on all software procured for the high assurance environments. This is true for Commercial Off-The-Shelf Software (COTS) or Government Off-The-Shelf Software (GOTS), as well as legacy systems to be ported to the high assurance environment, and assumes that the developer (either COTS or the enterprise representative through a custom GOTS developed contract) will perform these analyses and provide the flaw remediation (an adaptation of Common Criteria, 2009).

While formal methods in the development process and the obtaining of software development credentilas such as Capability Maturity Model (CMM), (CMMI, 2013), are valuable, they cannot be used to replace the vulnerability analyses. In an adversarial procurement environment, where the developer either cannot or will not perform these analyses, the developer must agree to allow the procuring agency to perform these analyses or hire a competent laboratory to do the analyses. These costs, if not included in a competitive bid will be added to the price for the evaluated product when evaluating alternative products for procurement.

The level of effort will depend on the product complexity. Execution of the vulnerability analyses does not require the execution of a full Common Criteria evaluation, although any such Common Criteria evaluation should include these analyses. The vulnerability analysis described here is based upon the product in its operational environment as opposed to the Target of Evaluation described in (Common Criteria, 2009). The flaw remediation process cannot be performed by anyone other than the software developer. It is assumed that license or purchase of a product will include a support contract that includes the flaw remediation provisions if these are not provided with the product.

Within the US, a number of competent laboratories exist to do these analyses. These laboratories have tools and testing techniques for penetration testing, and will provide a level of assurance on the threat mitigation that will lead to informed decisions. Many are able to conduct these analyses at higher security classification levels and have the capability to sign non-disclosure, non-complete arrangements with the software developers. Contracting for these analyses would include a competitive procurement where the developer is unable or unwilling to perform such tasks. In addition, the purchase of such equipment will be dependent upon successful completion of the vulnerability analysis. Prior vulnerability analysis reports may be used to the extent that the operational environments are similar or modifications to the prior analysis are completed and executed.

In the US there are seven laboratories that are licensed under the National Voluntary Laboratory Accreditation Program (NVLAP) Program, (NIST 2006). NVLAP provides third-party accreditation to testing and calibration laboratories. NVLAP's accreditation programs are established in response to Congressional mandates, administrative actions by the Federal Government, and requests from private-sector organizations and government agencies.
NVLAP operates an accreditation system that is compliant with ISO/IEC 17011, Conformity assessment — General requirements for accreditation bodies accrediting conformity assessment bodies, which requires that the competence of applicant laboratories be assessed by the accreditation body against all of the requirements of ISO/IEC 17025, General requirements for the competence of testing and calibration laboratories.

Figure 3 shows the basic vulnerability analysis flow.

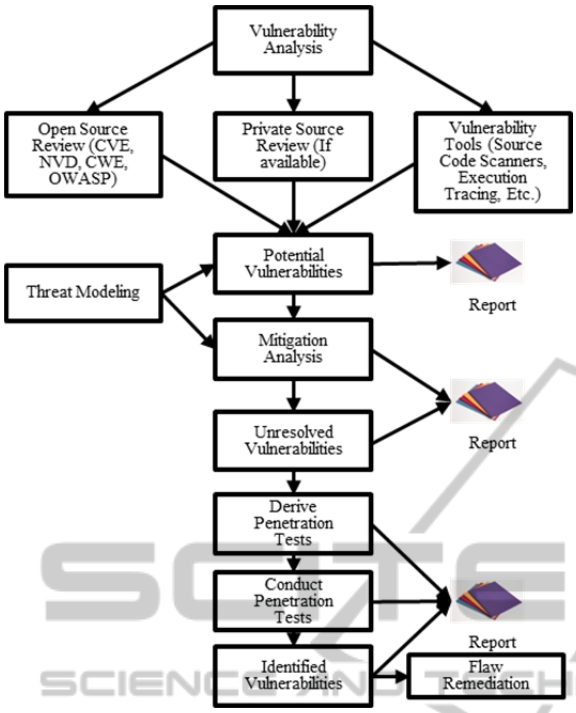

Figure 3: Vulnerability Analysis Process.

4.1 Vulnerability Analysis Objective

The objectives of a vulnerability analysis are:

1. To determine whether the product(s), in its operational environment, has easily identifiable exploitable vulnerabilities.
2. Identify those vulnerabilities.
3. Begin a remediation process that will close those vulnerabilities.
4. Excessive vulnerabilities may disqualify the product for enterprise use.

4.2 Vulnerability Analysis Required Information

The required information for vulnerability analyses is:

- The Product(s);
- The guidance documentation
- Identification of all interfaces.
- The Product(s) suitable for testing;
- Harnesses and software instrumentation necessary for testing the product.
- Information publicly available to support the identification of potential vulnerabilities. Information that may be used in these analyses are listed below:
- Common Vulnerabilities and Exposures (CVE®) (Mitre, 2013a) is a dictionary of common names (i.e., CVE Identifiers) for publicly known information security vulnerabilities. CVE's common identifiers make it easier to share data across separate network security databases and tools, and provide a baseline for evaluating the coverage of an organization's security tools. If a report from one of the security tools incorporates CVE Identifiers, the tool may quickly and accurately access fix information in one or more separate CVE-compatible databases to remediate the problem.
- National Vulnerability Database (NVD) (NIST, 2013) is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics.
- Common Weakness Enumeration (CWE™) (Mitre, 2013b) International in scope and free for public use, CWE provides a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems as well as better understanding and management of software weaknesses related to architecture and design.
- Others including the Open Web Application Security Project (OWASP, 2013).

The product developer or evaluator performs additional tests as a result of potential vulnerabilities encountered during the conduct of other parts of the evaluation. The use of the term guidance in this process refers to the operational guidance and the preparative guidance. Potential vulnerabilities may be in information that is publicly available, or not, and may require skill to exploit, or not. These two aspects are related, but are distinct. It should not be assumed that, simply because a potential vulnerability is identifiable from information that is publicly available, it can be easily exploited.

4.3 Obtaining Vulnerabilities

The product developer or evaluator examines sources of information publicly available to identify potential vulnerabilities in the product. There are

many sources of publicly available information, which should be considered, but a minimum set is listed above as input to the analysis. The product developer or evaluator should not constrain his consideration of publicly available information to the above, but should consider any other relevant information available.

The product developer or evaluator records in an evaluation report the identified potential vulnerabilities that are applicable to the product in its operational environment. The product developer or evaluator may use manual methods or source code scanning and execution tracing tools, or all in this analysis, but minimal coverage of the above listed sources of vulnerabilities and weaknesses are 100%. Each of the vulnerabilities identified as appropriate to the product in its operational environment, is assigned a category and rationale as follows:

a. That no further consideration of the potential vulnerability is required if for example the product developer or evaluator identifies mitigations in the operational environment, either IT or non-IT that prevent exploitation of the potential vulnerability in that operational environment. This may include mitigations within the product itself.

b. That for any reasons the potential vulnerabilities may be excluded from further consideration if the potential vulnerability is not applicable in the operational environment.

c. Otherwise the evaluator records the potential vulnerability for further consideration. This list of potential vulnerabilities applicable to the product in its operational environment, which can be used as an input into penetration testing activities, is reported in the evaluation report.

4.4 Deriving Penetration Tests

The product developer or evaluator derives penetration tests that are based on the search above for potential vulnerabilities, threat modelling activities, and other analysis methods. The product developer or evaluator prepares for penetration testing as necessary to determine the susceptibility of the product, in its operational environment, to the potential vulnerabilities identified during the search of the sources of information publicly available.

The product developer or evaluator produces penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation includes:

a) Identification of the potential vulnerability the product is being tested for;

b) Instructions to connect and setup all required test equipment as required for conducting the penetration test;

c) Instructions to establish all penetration test prerequisite initial conditions;

d) Instructions to stimulate the product;

e) Instructions for observing the behavior of the product;

f) Descriptions of all expected results and the necessary analysis to be performed on the observed behavior for comparison against expected results;

g) Instructions to conclude the test and establish the necessary post-test state for the product.

The product developer or evaluator conducts penetration testing based on the list of potential vulnerabilities identified above, and prepares an evaluation report on these tests. Where, as a result of evaluation, the product developer or evaluator discovers a potential vulnerability, this is reported in the evaluation report as a residual vulnerability. The vulnerability is reported as a flaw in the flaw remediation system with a priority commensurate with its potential for exploit and the consequences of a successful exploit.

4.5 Continuous Updating

The product developer or evaluator re-examines sources of information publicly available to identify potential vulnerabilities in the product either periodically or on demand. The analysis may be on demand when critical vulnerabilities or damaging exploits in similar products have been identified, changes in the operational environment are made, or other changes requiring further analysis.

4.6 Review and Approve

The enterprise representative examines the results of the evaluation report of actions above and of all penetration testing to determine that the product, in its operational environment, is resistant to an attacker appropriate to the high assurance environment. If the results reveal that the product, in its operational environment, has vulnerabilities exploitable by an attacker appropriate to the high assurance environment, then remedial action must be taken by the product developer. The enterprise representative ensures that periodic re-evaluation as provided is above is undertaken.

For critical and trusted software such as the Secure Token Server (STS) or services within the Enterprise Attribute Store (EAS), the enterprise representative may, at his option, conduct or procure

independent penetration testing as described in section II D above, including repeating some of the testing described and independently derived tests. The results of these tests may lead to the identification of vulnerabilities which are subject to the flaw remediation processes described below.

5 FLAW REMEDIATION

The flaw remediation process is the responsibility of the product developer (an adaptation of Common Criteria, 2009, Evaluation Methodology).

5.1 Flaw Remediation Objectives

The objective of this flaw remediation is to demonstrate that the product developer has established procedures that describe the tracking of security flaws, the identification of corrective actions, and the distribution of corrective action information to product users. Additionally, this process demonstrates that the product developer's procedures provide for the corrections of security flaws, for the receipt of flaw reports from product users, for assurance that the corrections introduce no new security flaws, for the establishment of a point of contact for each product user, and for the timely issue of corrective actions to product users. The administrative tracking and reporting of flaws may be outsourced, but the developer must be involved in the corrective actions and must be included in the developer's quality control system.

In order for the developer to be able to act appropriately upon security flaw reports from product users, product users need to understand how to submit security flaw reports to the product developer, and product developers need to know how to receive these reports. Flaw remediation guidance addressed to the product user ensures that product users are aware of how to communicate with the developer; flaw remediation procedures describe the product developer's role is such communication. The Flaw Remediation process as applied to security issues is shown in Figure 4.

5.2 Flaw Remediation Required Information

The required information for evaluation of flaw remediation processes is:

a) The flaw remediation procedures documentation;

b) Flaw remediation guidance documentation.

The product developer may use automated flaw remediation tracking systems such as trouble ticketing software or may manually track such flaws. However, it is expected that any flaw not remedied with 10 working days are referred to the quality assurance tracking system for formalized assignment of actions and priorities, and timely reviews to assure progress in resolution of open items. Security flaws normally have the highest assigned priority for remediation in high assurance systems. Remediation may be code changes and patch update, configuration changes and recommended changes to STIGs, or other remedies outside of the product in the operational environment. The enterprise representative may reject the last approach depending on its impact.

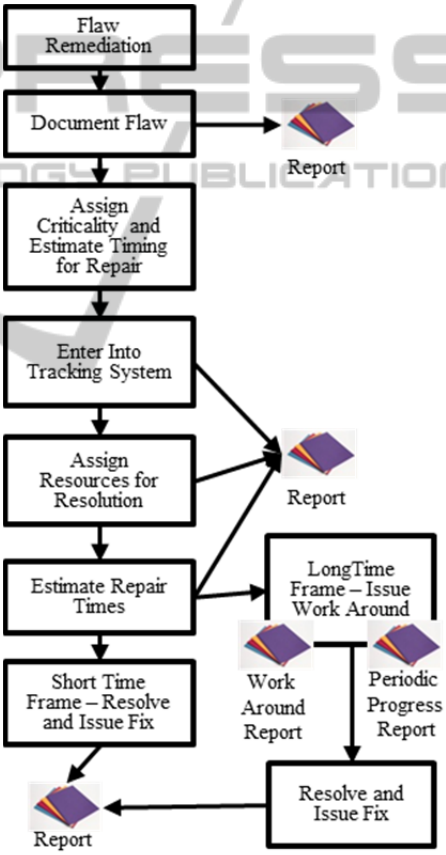

Figure 4: Flaw Remediation Process.

5.3 A Flaw Remediation Process

The product developer provides a flaw remediation process:

a. That describes the procedures used to track all reported security flaws in each release of the product. The procedures describe the actions that are taken by the product developer from the time each

suspected security flaw is reported to the time that it is resolved. This includes the flaw's entire time frame, from initial detection through ascertaining that the flaw is a security flaw, to resolution of the security flaw. If a flaw is discovered not to be security-relevant, there is no need (for the purposes of the Flaw remediation requirements) for the flaw remediation procedures to track it further; only that there be an explanation of why the flaw is not security-relevant.

b. That describe of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw. The procedures identify the actions that are taken by the product developer to describe the nature and effects of each security flaw in sufficient detail to be able to reproduce it. The description of the nature of a security flaw addresses whether it is an error in the documentation, a flaw in the design of the product, a flaw in the implementation of the product, etc. The description of the security flaw's effects identifies the portions of the product that are affected and how those portions are affected.

c. That when applied these procedures would identify the status of finding a correction to each security flaw. The flaw remediation procedures identify the different stages of security flaws. This differentiation includes at least: suspected security flaws that have been reported, suspected security flaws that have been confirmed to be security flaws, and security flaws whose solutions have been implemented. It is permissible that additional stages (e.g. flaws that have been reported but not yet investigated, flaws that are under investigation, security flaws for which a solution has been found but not yet implemented) be included.

d. That requires corrective actions be identified for each of the security flaws. Corrective action may consist of a repair to the hardware, firmware, or software portions of the product, a modification of product guidance, or both. Corrective action that constitutes modifications to product guidance (e.g. details of procedural measures to be taken to obviate the security flaw) includes both those measures serving as only an interim solution (until the repair is issued) as well as those serving as a permanent solution (where it is determined that the procedural measure is the best solution). If the source of the security flaw is a documentation error, the corrective action consists of an update of the affected product guidance. If the corrective action is a procedural measure, this measure includes an update made to the affected product guidance to reflect these corrective procedures.

e. That describes the methods used to provide flaw information, corrections and guidance on corrective actions to product users. The necessary information about each security flaw consists of its description, the prescribed corrective action, and any associated guidance on implementing the correction. Product users may be provided with such information, correction, and documentation updates in any of several ways, such as their posting to a website, their being sent to product users, or arrangements made for the product developer to install the correction. In cases where the means of providing this information requires action to be initiated by the product user, product guidance must be adequate to ensure that it contains instructions for retrieving the information. The only metric for assessing the adequacy of the method used for providing the information, corrections and guidance is that there is a reasonable expectation that product users can obtain or receive it. For product users who register with the, the passive availability of this information is not sufficient. Product developers must actively send the information (or a notification of its availability) to registered product users.

f. That describes a means by which the product developer receives from product user's reports and enquiries of suspected security flaws in the product. The procedures ensure that product users have a means by which they can communicate with the product developer. By having a means of contact with the developer, the user can report security flaws, enquire about the status of security flaws, or request corrections to flaws. This means of contact may be part of a more general contact facility for reporting non-security related problems.

g. That includes a procedure requiring timely response and the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw. The issue of timeliness applies to the issuance of both security flaw reports and the associated corrections. However, these need not be issued at the same time. It is recognized that flaw reports should be generated and issued as soon as an interim solution is found, even if that solution is as drastic as turn off the product. Likewise, when a more permanent (and less drastic) solution is found, it should be issued without undue delay. It is unnecessary to restrict the recipients of the reports and associated corrections to only those product users who might be affected by the security flaw; it is permissible that all product users be given such reports and corrections for all security flaws,

provided such is done in a timely manner.

h. That results in automatic distribution of the reports and associated corrections to the registered product users who might be affected. Automatic distribution does not mean that human interaction with the distribution method is not permitted. In fact, the distribution method could consist entirely of manual procedures, perhaps through a closely monitored procedure with prescribed escalation upon the lack of issue of reports or corrections. It is unnecessary to restrict the recipients of the reports and associated corrections to only those product users who might be affected by the security flaw; it is permissible that all product users be given such reports and corrections for all security flaws, provided such is done automatically.

5.4 Flaw Remediation Quality System

The product developer provides reporting processes:

a. That ensures reported security flaws are remediated and the remediation procedures issued to product users. The flaw remediation procedures cover not only those security flaws discovered and reported by developer personnel, but also those reported by product users. The procedures are sufficiently detailed so that they describe how it is ensured that each reported security flaw is remediated. The procedures contain reasonable steps that show progress leading to the eventual, inevitable resolution. The procedures describe the process that is taken from the point at which the suspected security flaw is determined to be a security flaw to the point at which it is resolved.

b. That ensures that the product users are issued remediation procedures for each security flaw. The procedures describe the process that is taken from the point at which a security flaw is resolved to the point at which the remediation procedures are provided. The procedures for delivering remediation procedures should be consistent with the security objectives.

c. That provides safeguards that any corrections to these security flaws do not introduce any new flaws. Through analysis, testing, or a combination of the two, the developer may reduce the likelihood that adverse effects are introduced when a security flaw is corrected.

5.5 Flaw Remediation Reporting

The product developer provides remediation guidance:

a. That describes a means by which product users report to the developer any suspected security flaws in the product. The guidance ensures that product users have a means by which they can communicate with the product developer. By having a means of contact with the developer, the user can report security flaws, enquire about the status of security flaws, or request corrections to flaws.

b. That describes a means by which product users may register with the developer, to be eligible to receive security flaw reports and corrections. Enabling the product users to register with the developer simply means having a way for each product user to provide the developer with a point of contact; this point of contact is to be used to provide the product user with information related to security flaws that might affect that product user, along with any corrections to the security flaw. Registering the product user may be accomplished as part of the standard procedures that product users undergo to identify themselves to the developer, for the purposes of registering a software license, or for obtaining update and other useful information.

There need not be one registered product user per installation of the product; it would be sufficient if there were one registered product user for an organization. It should be noted that product users need not register; they must only be provided with a means of doing so. However, users who choose to register must be directly sent the information (or a notification of its availability).

5.6 Review and Approve

The enterprise representative examines the results of the above actions to determine that the product, in its operational environment, has sufficient flaw remediation. If the results reveal that the product, in its operational environment, has insufficient flaw remediation, then remedial action must be taken by the product developer or the product may be replaced.

6 SUMMARY

This paper has provided a set of processes by which the enterprise can field relatively vulnerability-free software, at least to the extent of known vulnerabilities and exploits. New exploits are subject to temporary solution and become part of the flaw remediation system where they are reported and monitored until a satisfactory solution is achieved. It is not anticipated that the enterprise will be 100% exploit free from this process alone. Additional

measures, including firewalls, ports and protocols restrictions, and some communication scanning may be required. This process should, however, reduce the attack space significantly. The implementation (scheduled for summer 2014) will have to be monitored and evaluated as it proceeds. This includes the tracking of exploits, the response of the software remediation system, and the degree to which the vulnerability was knowable before the exploit as well as newly discovered vulnerabilities. It is expected that these analyses listed in this paper will need refinement based upon that feedback. Portions of this architecture are described in (Simpson, 2011, 2012a, b).

REFERENCES

Common Criteria for Information Technology Security Evaluation, 2009 (all version 3.1, revision 3):
a) Part 1: Introduction and general model.
b) Part 2: Functional security components.
c) Part 3: Assurance security components.
d) Common Methodology for Information Technology Security Evaluation.

CMMI Institute, 2013, Standard CMMI Appraisal Method for Process Improvement (SCAMPI) Version 1.3a: Method Definition Document for SCAMPI A, B, and C, http://cmmiinstitute.com/resource/standard-cmmi-appraisal-method-process-improvement-scampi-b-c-version-1-3a-method-definition-document/

Department of Defense, 2012a, *Committee on National Security Systems Instruction (CNSSI*) No. 1253, "Security Categorization and Control Selection for National Security Systems' categories for Moderate or High Risk Impact as delineated in NIST 800-53.

Department of Defense, 2012b, DoD Directive (DoDD) O-8530.1, Computer Network Defense (CND).

Finifter, Matthew, et. al., "An Empirical Study of Vulnerability Rewards Programs", USENIX Security 2013, August 15, 2013.

HP Security Tools, 2013, http://h20331.www2.hp.com/hpsub/cache/281822-0-0-225-121.html?jumpid=ex_2845_vanitysecur/productssecurity/ka011106

Huang, Y.-W., et. al., 2004, "Securing web application code by static analysis and runtime protection," in WWW '04: Proceedings of the 13th international conference on World Wide Web. New York, NY, USA: ACM, , pp. 40–52.

IBM Rational, 2013, http://www.03.ibm.com /software/products /us/en/appscan

Intel Compilers, 2013, http://software.intel.com/en-us/intel-compilers/

Kiezun, A., et. al., 2009, "Automatic creation of SQL injection and cross-site scripting attacks," in ICSE'09, Proceedings of the *30th International Conference on Software Engineering,* Vancouver, BC, Canada, May 20–22.

Jones, Paul, 2010, "A Formal Methods-based verification approach to medical device software analysis". Embedded Systems Design., http://www.embedded.com/design/prototyping-and-development/4008888/A-Formal-Methods-based-verification-appro ach-to-medical-device-software-analysis

Jovanovic, N., et. al., 2006, "Pixy: A static analysis tool for detecting web application vulnerabilities (short paper)," in *2006 IEEE Symposium on Security and Privacy*, pp. 258–263, [Online]. Available: http://www.iseclab.org/papers/pixy.pdf

Kals, S., et. al., 2006, "Secubat: a web vulnerability scanner," in WWW '06: Proc. 15th Int'l Conf. World Wide Web, pp. 247–256.

Livshits, Benjamin, 2006, Improving Software Security with Precise Static and Runtime Analysis, , section 7.3 "*Static Techniques for Security," Stanford doc. thesis.*

Livshits B., et. al., 2008, "Securing web applications with static and dynamic information flow tracking," in *PEPM '08: Proceedings of the 2008 ACM SIGPLAN* symposium on Partial evaluation and semantics based program manipulation. New York, NY, USA: ACM, pp. 3–12.

Maggi, F., 2009, "Protecting a moving target: Addressing web application concept drift," in RAID, pp. 21–40.

Mitre, 2013a, Common Vulnerability and Exposures, http://cve.mitre.org/

Mitre, 2013b, Common Weakness Enumeration, http://cwe.mitre.org/

Mcallister, S., et. al., 2008, "Leveraging user interactions for in-depth testing of web applications," in RAID '08: Proc. 11th Int'l Symp. Recent Advances in Intrusion Detection, pp. 191–210.

Mosaic, 2013, http://mosaicsecurity.com/categories/27-network-penetration-testing

NIST, 2006, National Voluntary Laboratory Accreditation Program, http://www.nist.gov/nvlap/upload/nist-handbook-150.pdf

NIST, 2009, National Institute of Standards, Gaithersburg, Md: FIPS PUB 800-53, Recommended Security Controls for Federal Information Systems and Organizations, Revision 3, August 2009.

NIST, 2013, National Vulnerability Database, http://nvd.nist.gov/

Simpson, William R, et.al.., 2011, Lecture Notes in Engineering and Computer Science, Proceedings World Congress on Engineering and Computer Science, Volume I, "High Assurance Challenges for Cloud Computing", pp. 61-66, Berkeley, CA.

Simpson, William R, and Chandersekaran, C.., 2012a, *Lecture Notes in Engineering and Computer Science, Proceedings World Congress on Engineering*, The 2012 International Conference of Information Security and Internet Engineering, Volume I, "Claims-Based Enterprise-Wide Access Control", pp. 524-529, London,.

Simpson, William R, and Chandersekaran, C.., 2012b, *International Journal of Scientific Computing,* Vol. 6, No. 2, "A Uniform Claims-Based Access Control for the Enterprise", ISSN: 0973-578X, pp. 1-23.

The Open Web Application Security Project (OWASP), 2013, https://www.owasp.org/index.php/Main_Page

Wassermann G. and Z. Su, 2007, “Sound and precise analysis of Web Applications for Injection Vulnerabilities,” *SIGPLAN Not., vol.42, no.6,* pp.32–41.

Wichmann, B. A., et. al. 1995, Industrial Perspective on Static Analysis. *Software Engineering Journal,* 69-75.