Finding Software Vulnerabilities Before Hackers Do

Security breaches are on the rise. Confidential data is now stored and software applications are run on company computer systems, in the cloud and on mobile devices. With ever-increasing points of entry, it is not a surprise that breaches are on the rise. No organization, large or small, is immune to security risks. Therefore, it is critical that every business takes the necessary precautions to ensure their data and enterprise are protected from the numerous potential vulnerabilities that surround us every day. This includes protection using software applications.

Industries, such as healthcare, financial services and retail have been particularly vulnerable. Incidents at Home Depot, Sony, Target, and the federal government, to name a few, have made news headlines in recent years. These data breaches have impacted millions of consumers, cost organizations hundreds of billions of dollars, and elevated the concerns of information security professionals.

According to *Build Security In* (BSI), 90% of reported security problems result from weaknesses in applications, such as web and mobile apps.[1] Those organizations that believe that just having user names and passwords for authentication and encrypting data as it moves throughout cyber space are sufficient, may be putting their data at serious risk.

To defend from malicious attacks, large corporations often have bug bounties where they pay "white hat" hackers to find vulnerabilities in their applications before attackers do. Google is said to pay these hackers up to $20,000 per vulnerability, and Microsoft has been said to pay as much as $150,000. Even United Airlines has a bug bounty program, recently giving one individual 1,000,000 air miles for finding a vulnerability in their software. There is an entire community of people making a living on finding bugs to protect organizations from debilitating data breaches, and another large community of cyber criminals out to do harm.

What is Application Security Testing?

Most cyber security incidents can be traced back to a software vulnerability that was inadvertently put there when the code was developed.

Vulnerabilities in applications can present themselves during the design and development of the application, as well as during upgrades and maintenance. With so many opportunities for threats, organizations need to take the proper steps to test their applications for any security holes throughout the entire software development lifecycle (SDLC).

Despite the high risk of attacks, it is not uncommon for the software team to wait until the development process is complete before testing for weaknesses, which are defined as potential vulnerabilities that may or may not be exploitable. This goes against industry best practices, which have shown that it actually costs a lot less to "build security in" during the software development process than to fix the vulnerabilities later in the lifecycle.

Application Security Testing (AST) tools and methodologies are becoming more widely adopted by software developers and security analysts to identify holes in software applications; although it still needs to be moved higher on the list of security strategies for organizations to implement. And this does not just mean software development companies, it also means organizations developing their own in-house applications, or even those buying software solutions from third-party vendors. It is important, whether an organization builds or buys a software package, to ensure it does not contain any weaknesses that will make the data housed within the system vulnerable to exploits.

Quality and security are not separate worlds, but rather two sides of the same coin.

Quality vs. Security

AST is often confused with quality testing. Some think that if they have quality code then the application is also secure. That's not the case. Quality testing is focused on whether the application is doing what it is *supposed* to do. Quality issues can include: confusing code that is hard to follow; performance issues, such as the code working slower than it should; concurrency issues; memory leaks; null pointers; infinite loops; and redundant or dead code. On the other hand, security testing is focused only on ensuring the application is protected from intruders. Security issues could include SQL injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), buffer overflows, using hard-coded passwords, weak encryption, sensitive data, etc.

Quality and security issues are two separate but related concerns. One can easily make an argument that code that has quality concerns is more likely to have security concerns as well, and in fact a number of recent studies looking at vulnerability incidence in faulty code confirm that there is a strong correlation between the two. A recent study by the Software Engineering Institute, for instance, found that development groups with a strong focus on quality tended to have fewer vulnerabilities in their source code.[2] Therefore, it is important to keep in mind that quality and security are not separate worlds, but rather two sides of the same coin.

Types of AST Methodologies/Tools

There are a variety of different application security testing methods that should be considered by application developers and security professionals. These AST techniques include manual code review, Static Application Security Testing (SAST), and Dynamic Application Security Testing (DAST).

Manual Testing

Code reviews are a venerated tradition in software development. Typically, one or more reviewers will manually scan the source under review ensuring that the system requirements are being met, the design is consistent, and quality standards are maintained. Although code reviews have traditionally been used primarily with a quality focus, they are a key component of the security testing toolbox. Reviewers scan the source code for vulnerabilities that can be exploited, and bring them to the reviewee's attention. The review process is largely the same regardless of whether one is doing a quality or a security review. In fact, reviewing for both should be done as part of the same assessment. What changes is the mindset one uses to approach the review process, and the nature of the questions one asks during the assessment. Understanding the *threat model* for the code being reviewed, for instance, is an important part of the process, in order to be on the lookout for potential attack vectors that might be exploited.

For certain types of applications, such as the mission-critical software that powers a key infrastructure or the software used in a NASA space program, reviewing each line of code is not only a requirement but is also a more cost-effective approach than automated solutions. However, for most software, carefully reviewing every single source line is not viable. Therefore, understanding when and how to do so is an important skill. Code reviews typically yield the best bang for the buck when keeping an eagle eye on critical subsystems, and evaluating the security posture of new subsystems that are still evolving. Code reviews also help new engineers further develop a security mindset.

For additional reading on the topic, a variety of resources exist for understanding how code reviews fit into application security, including the well-documented *OWASP Code Review Project* from the Open Source Web Application Security Project (OWASP), an organization dedicated to fostering a global culture of secure application development.[3]

SAST

Also known as *white box testing*, SAST is a popular method that employs tools to automatically analyze the application's source code, byte code, or binary code line-by-line to expose weaknesses during the programming or testing phases of the SDLC, *before* the software is deployed.

SAST tools are used early in, and throughout, the SDLC to test the application from the inside out, and do not require a running system to perform those evaluations. By detecting flaws in the code early in the process, weaknesses can be fixed before hackers detect them, and before they become true vulnerabilities for an organization.

Essentially, with SAST tools the reviewer or tester inputs the source code and/or binary files that they want to analyze, and the tool scans those files and presents a list of potential vulnerabilities. The list produced can be extensive: potentially tens of thousands of weaknesses can be identified during a scan. Managing such a large number of findings can be overwhelming, so it is important to use the right tools and approaches to make the process more manageable. This includes:

- Distilling the list of reported weaknesses down to a manageable set. This is typically achieved by triaging the potential vulnerabilities to determine which ones are high impact issues that need to be dealt with immediately.
- Using existing top-weakness lists and industry standards to focus initial efforts on a directed subset of the issues. The *OWASP Top 10* [4] and the SANS Institute's *Top 25 Most Dangerous Software Errors*, for example, are good starting points to look at issues that are commonly seen in code. Alternatively, looking at the cross-section of weaknesses affecting *compliance*, such as with PCI or HIPAA, offers a focused approach to prioritizing the identified issues.
- Identifying false positives, and tagging the specific lines of code associated with the false positives so that they don't re-appear in future static code scans.
- Assigning weaknesses to developers for remediation.
- Collaborating among developers, and between developers and security analysts.

For all of the above, the use of a **software vulnerability management system** can significantly help alleviate the weakness processing and remediation challenge. The triage process for instance, although largely a manual one, can benefit significantly by using a vulnerability management system. It can help streamline the workflow, making the process quicker and more efficient. Similarly, mapping the results back to top-lists or compliance standards can be a largely automated process with the use of the right vulnerability management system.

A subset of SAST tools specializes in *dependency checking* to identify vulnerabilities in third-party libraries used by the system being scanned. A significant component of software projects developed today are third-party libraries, often open source, woven together to form the final application. Understanding the security posture of this significant portion of software projects is an important activity to perform during security assessments. Dependency checking tools will scan the software projects, identify those third-party libraries, and list the known vulnerabilities for those libraries.

DAST

DAST tools are considered *black box testing* tools, analyzing applications "dynamically," in real-time, while the application is running. DAST is also known as penetration testing, or fuzz testing.

The key difference between SAST and DAST tools is that DAST is done from the outside looking in, simulating attacks against the application and analyzing how the application behaves in response to those attacks, in order to identify vulnerabilities. Testers usually have no access to or knowledge of the inner workings of the application prior to testing, as they attempt to exploit any potential vulnerabilities, including those outside the code and in third-party interfaces.

The DAST process starts with setting up a staging environment since such testing, **at least initially**, should not be conducted in a production environment. The DAST tool is typically manually tuned to identify the *attack surface* (all potential vectors on which an attack can occur). The tools perform active probing for vulnerabilities – conducting one test case after another, logging issues as they are found. Like SAST tools, the vulnerabilities are identified and reported, and remediation can begin.

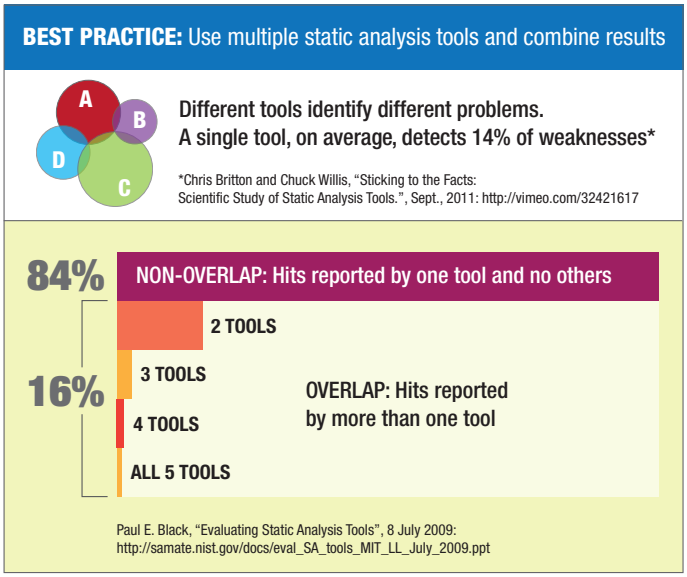
Source code, byte code, and binaries are not required with DAST, and it is easier to use and less expensive than SAST tools. Because the testing is conducted at runtime and vulnerabilities can be confirmed more easily, DAST typically produces more true positives than with SAST tools. On the other hand, DAST tools are usually unable to isolate the exact site of a weakness in the code, whereas SAST tools will often describe in detail the code paths leading to their identified weaknesses.

By providing the outside-in perspective, DAST tools can provide valuable insight and are ideal to be used closer to the end of the release cycle rather than at the start. In cases when source code is not available, DAST may even be the only viable security testing option.

Limited Code Coverage

As mentioned, AST tools can find thousands of weaknesses in an application. The testing results can leave a developer feeling devastated that so many weaknesses were found, or possibly gratified that the tool caught so many weaknesses before the application was released to the user community.

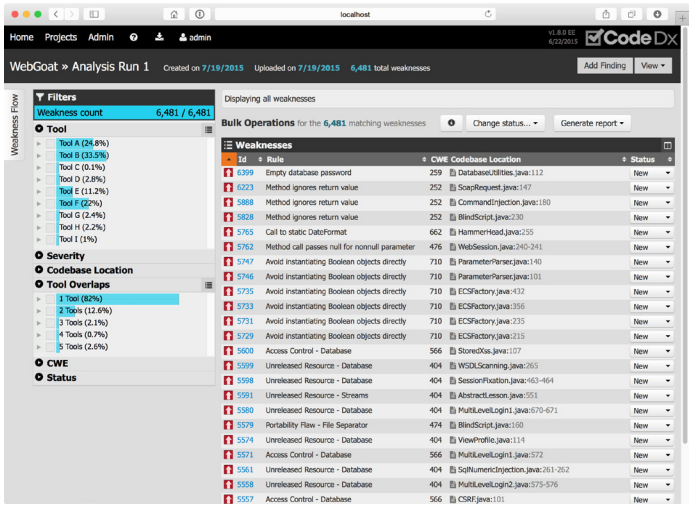
No matter how they feel about the results, software developers must understand that by running only one application security testing tool, even the best on the market, they are missing most of the weaknesses in their code. One tool only covers the tip of the iceberg. There could be thousands and thousands of flaws that the analysis tool is not seeing, some of which could result in serious weaknesses being missed that could leave the system vulnerable to exploits.



According to a study done by the National Security Agency's (NSA) Center for Assured Software (CAS), the average tool covers just eight of the 13 weakness classes (e.g. buffer handling, file handling, initialization and shutdown, and number handling), which is 61.5%. This study also found that the average tool covers only 22% of the flaws in each of the 13 weakness classes.[5] If the percentage of the flaws is multiplied by the percentage of weakness classes covered, the total coverage of the average tool is only 14%. This is eye opening for many software developers, who have assumed that their vulnerability scanners cover a much larger area. Missing more than 80% of the weaknesses in the application code should not be acceptable for any organization.

Managing Multiple Tools

In addition to discovering that each of the analysis tools failed to report a significant portion of the flaws studied, the NSA CAS found that the tools perform differently on different languages and on different weakness classes. Moreover,



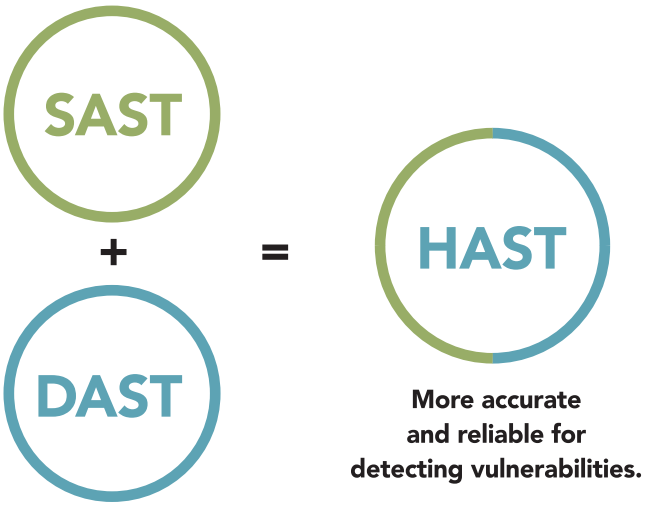
Output of Code Dx Software Vulnerability Management System that consolidates the results of nine source code analysis tools, showing minimal overlap in their 6,000+ findings.

as Paul E. Black of the National Institute of Standards and Technology (NIST) reports in a presentation on "Evaluating Static Analysis Tools," different static analyzers are used for different purposes. He provides examples such as checking for intellectual property violation, helping developers decide if anything needs to be fixed, and/or helping auditors or reviewers decide if it is good enough for use. However, both studies found that complementary tools could be combined to achieve better results.[6]

Using two or more tools will provide greater vulnerability coverage. The fact that each tool specializes in different weakness classes and different languages eliminates much of the overlap among the tools. In addition, when there *is* an overlap, the developer will be more confident that the identified flaws are not false positives, and can focus on ensuring that those weaknesses are fixed.

Leveraging multiple tools does have its challenges – namely in the additional time required to set up and run the tools and compare the results, as well as in the cost required to add more tools. Furthermore, comparing the results can be painstaking, as each tool produces a set of weaknesses with its own naming conventions and severity ratings.

This is where software vulnerability management systems come into play. These solutions show the overlap in the various AST tools. Whether commercial scanners, open source vulnerability tools, or a combination of both are being used, software vulnerability management systems will show the results of each and identify the vulnerabilities that were found by each tool. They correlate and normalize the results from commercial and open source tools to deliver a consolidated set of results that provides greater coverage of



potential vulnerabilities in the source code, and a better assessment of an organization's overall enterprise risk. In addition, software vulnerability management systems provide easy-to-use mechanisms for prioritizing weaknesses, tagging false positives, assigning them to developers, tracking the progress of remediation, and preparing reports of findings.

Software vulnerability management systems become increasingly important as users adopt the growing trend to combine the results of SAST and DAST using a technique called Hybrid Application Security Testing (HAST) to perform a behavioral assessment of the application. By leveraging both methodologies, HAST promises a more accurate and reliable approach for testing source code and detecting vulnerabilities.

Key Players in the AST Market

There are many AST tools on the market today that help organizations identify flaws in their software or applications that can be exploited in a number of different ways. The following is a list of some of the more well-known SAST and DAST tools on the market. They include both commercial products as well as open source solutions.

SAST Tools

Open Source/Free Tools:

- **Brakeman** is an open source vulnerability scanner specifically designed for Ruby on Rails applications. It statically analyzes Rails application code to find security issues at any stage of development.
- **Dependency Check** is an OWASP utility that identifies project dependencies and checks if there are any known, publicly disclosed vulnerabilities. Currently Java, .NET and Python dependencies are supported.
- **FindBugs** is a program that uses static analysis to look for bugs in Java code. It is free software, distributed under the terms of the Lesser GNU Public License, and has been downloaded more than a million times.
- **Retire.js** is an open source command line scanner that helps identify JavaScript libraries with known vulnerabilities in applications.
- **FxCop** is a Microsoft application that analyzes managed code assemblies (code that targets the .NET Framework common language runtime, or CLR) and reports information about the assemblies, such as possible design, localization, performance, and security improvements.

Commercial Products:

- **Checkmarx** is a Source Code Analysis (SCA) solution designed for identifying, tracking and fixing technical and logical security flaws from the root: the source code.
- **GrammaTech's CodeSonar** analyzes source code and binaries, identifying programming bugs that can result in system crashes, memory corruption, leaks, data races, and security vulnerabilities.
- **HP Fortify** offers comprehensive application security solutions that cover every aspect of application security testing, software security management, and application self-protection to help you secure the software that runs your business.
- **IBM AppScan Source** helps organizations lower costs and reduce risk exposure by identifying web-based and mobile application source code vulnerabilities early in the software development lifecycle, so they can be fixed before deployment.
- **Parasoft** enables development teams to build security into their application by facilitating code-hardening practices based on accepted industry standards, such as OWASP Top 10, CWE/SANS Top 25, and PCI DSS.
- **Veracode** is a patented binary SAST technology that analyzes all code – including third-party components and libraries – without requiring access to source code.
- **Sonatype Nexus Auditor** continuously monitors for security and legal risk in applications. It provides greater visibility into exactly which components are used, including dependencies, known security vulnerabilities, license obligations, and more.

DAST Tools

Open Source/Free Tools:

- **Arachni** is an open source, full-feature, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the security of web applications.
- **OWASP ZAP (Zed Attack Proxy)** is an easy-to-use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experience, and as such is ideal for developers and functional testers who are new to penetration testing, as well as being a useful addition to an experienced pen-tester's toolbox.
- **W3af** is a Web Application Attack and Audit Framework. The project's goal is to create a framework to help secure web applications by finding and exploiting all web application vulnerabilities. The framework is developed using Python to be easy to use and extend, and licensed under GPLv2.0.
- **Skipfish** is Google's active web application security reconnaissance tool. It prepares an interactive sitemap for the targeted site by carrying out a recursive crawl and dictionary-based probes. The resulting map is then annotated with the output from a number of active (but hopefully non-disruptive) security checks. The final report generated by the tool is meant to serve as a foundation for professional web application security assessments.

Commercial Products:

- **Acunetix Web Vulnerability Scanner (WVS)** is a tool that crawls a website, automatically analyzes the web applications, and finds perilous SQL injection, Cross-Site Scripting, and other vulnerabilities. Concise reports identify where web applications need to be fixed, thus enabling you to protect your business from impending hacker attacks.
- **Burp Suite** is an integrated platform for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface to finding and exploiting security vulnerabilities.
- **HP WebInspect** is an automated dynamic testing tool that mimics real-world hacking techniques and attacks, and provides comprehensive dynamic analysis of complex web applications and services.
- **IBM AppScan** enhances web application security and mobile application security, improves application security program management, and strengthens regulatory compliance. By scanning web and mobile applications prior to deployment, AppScan enables users to identify security vulnerabilities and generate reports and fix recommendations.
- **Netsparker** is a web application security scanner with support for both detection and exploitation of vulnerabilities. It aims to be free of false positives by only reporting confirmed vulnerabilities after successfully exploiting or otherwise testing them.
- **Veracode's** DAST technology identifies architectural weaknesses and vulnerabilities in running web applications.

Educating Stakeholders

The development team and security professionals must have buy-in from senior management to invest in AST tools up front. If they do not invest early, they run the risk of paying significantly later when a security problem is discovered after a product is released.

There are a number of resources available to help convince senior management that AST is essential for protecting valuable information assets from malicious attacks that can be detrimental to an organization's bottom line.

- **Glossary of AST Terminology** is an online resource of application security testing terminology developed by Code Dx, Inc. to help educate those just starting out in the AST market, or to refresh those who have been in the industry for years.[7]
- **OWASP** is a not-for-profit organization with more than 43,000 members worldwide, focused on improving the security of software. Its mission is to make software security visible, so that individuals and organizations worldwide can make informed decisions about true software security risks.

  OWASP is perhaps best known for its Top 10 list of the most common web vulnerabilities. Compiled by security experts from around the world and first published in 2004, the list is updated by the OWASP Foundation every three years. The 2013 list includes the following vulnerabilities, starting with the most prevalent: SQL Injection, Broken Authentication & Session Management, XSS, Insecure Direct Object references, Security Misconfiguration, Sensitive Data Exposure, Missing Data Exposure, CSRF, Using Components with Known Vulnerabilities, and Invalidated Redirects & Forwards.

- **ISACA** (Information Systems Audit and Control Association) is an independent, nonprofit, global association that engages in the development, adoption and use of globally accepted, industry-leading knowledge and practices for information systems. It provides practical guidance, benchmarks, and other effective tools for all enterprises that use information systems. Through its comprehensive guidance and services, ISACA defines the roles of information systems governance, security, audit, and assurance professionals worldwide. The COBIT framework and the CISA, CISM, CGEIT and CRISC certifications are ISACA brands, respected and used by these professionals for the benefit of their enterprises.[8]

- **BSI Software Assurance Initiative** is a project of the Strategic Initiatives Branch of the National Cyber Security Division (NCSD) of the Department of Homeland Security (DHS). *Build Security In* is a collaborative effort that provides practices, tools, guidelines, rules, principles, and other resources that software developers, architects and security practitioners can use to build security into software in every phase of its development.[9]

- **Software Assurance Marketplace**, also known as the SWAMP, was developed to make it much easier to regularly test the security of software applications and to provide an online laboratory for software assessment tool inventors to build stronger tools. It is a no-cost, high-performance, centralized cloud computing platform that includes an array of open-source and commercial software security testing tools, as well as a comprehensive results viewer to simplify vulnerability remediation. It was funded by DHS and is located at the Morgridge Institute for Research on the campus of the University of Wisconsin in Madison.

- **The SANS Institute** is a cooperative research and educational organization reaching more than 165,000 security professionals around the world. At the heart of SANS are the many security practitioners in varied global organizations, from corporations to universities, working together to help the entire information security community.

  SANS recently released a report entitled, *2015 State of Application Security: Closing the Gap,* which explores the current state of application security through the lens of both builders and defenders.[10]

Five Key Recommendations

With 90% of security incidents resulting from exploits against defects in software, according to DHS, it is not worth putting an enterprise at risk by not finding and fixing those defects. Application security testing needs to be a core part of every organization's information security strategy, whether it is developing its own software or purchasing applications from other sources.

Application security testing can seem like an overwhelming undertaking, but it is essential to keeping an organization's information assets secure. We conclude this white paper with five recommendations to help ease the AST process and minimize the potential for data to be exploited by malicious attackers.

1. Variety – adopt all three AST techniques (manual, SAST, DAST) in limited doses, initially, to determine how best to integrate them into your SDLC.

2. Manage – use a software vulnerability management system to manage the outputs of multiple testing tools to correlate and normalize results.

3. Focus – don't get overwhelmed. Focus on a subset of the initial findings, as the thousands of weaknesses that will be identified can't all be fixed.

4. Prioritize – focus on the most important weaknesses first. Starting with the OWASP Top 10 can help you focus on a well-known subset of critical vulnerabilities.

5. Integrate – make sure the testing process is part of the SDLC workflow. It is much less expensive to address issues early, rather than waiting until after a release is complete.

About Code Dx

Code Dx, Inc. provides easy and affordable software vulnerability management systems that enable software developers, testers, and security analysts to find and manage vulnerabilities in software. The Code Dx solutions integrate the results of multiple Application Security Testing (AST) tools and manual reviews into a consolidated set of results for easy triage, prioritization, and remediation. The core technology was partially funded by DHS Science & Technology to help secure the nation's software supply chain.

Footnotes:

1. Retrieved August 20, 2015 from https://buildsecurityin.us-cert.gov/software-assurance.
2. Carol Woody, Robert Ellison and William Nichols, "Predicting Software Assurance Using Quality and Reliability Measures," Software Engineering Institute, December 2014, page 11.
3. Retrieved August 20, 2015 from https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project.
4. Retrieved August 20, 2015 from https://www.owasp.org/index.php/Main_Page.
5. Retrieved August 20, 2015 from https://vimeo.com/32421617.
6. Retrieved August 20, 2015 from samate.nist.gov/docs/eval_SA_tools_MIT_LL_July_2009.ppt.
7. Retrieved August 25, 2015 from: http://codedx.com/ast-glossary/.
8. Retrieved August 20, 2015 from https://www.isaca.org/Pages/default.aspx.
9. Retrieved August 20, 2015 from https://buildsecurityin.us-cert.gov/.
10. Retrieved October 12, 2015 from https://www.sans.org/reading-room/whitepapers/analyst2015-state-application-security-closing-gap-35942