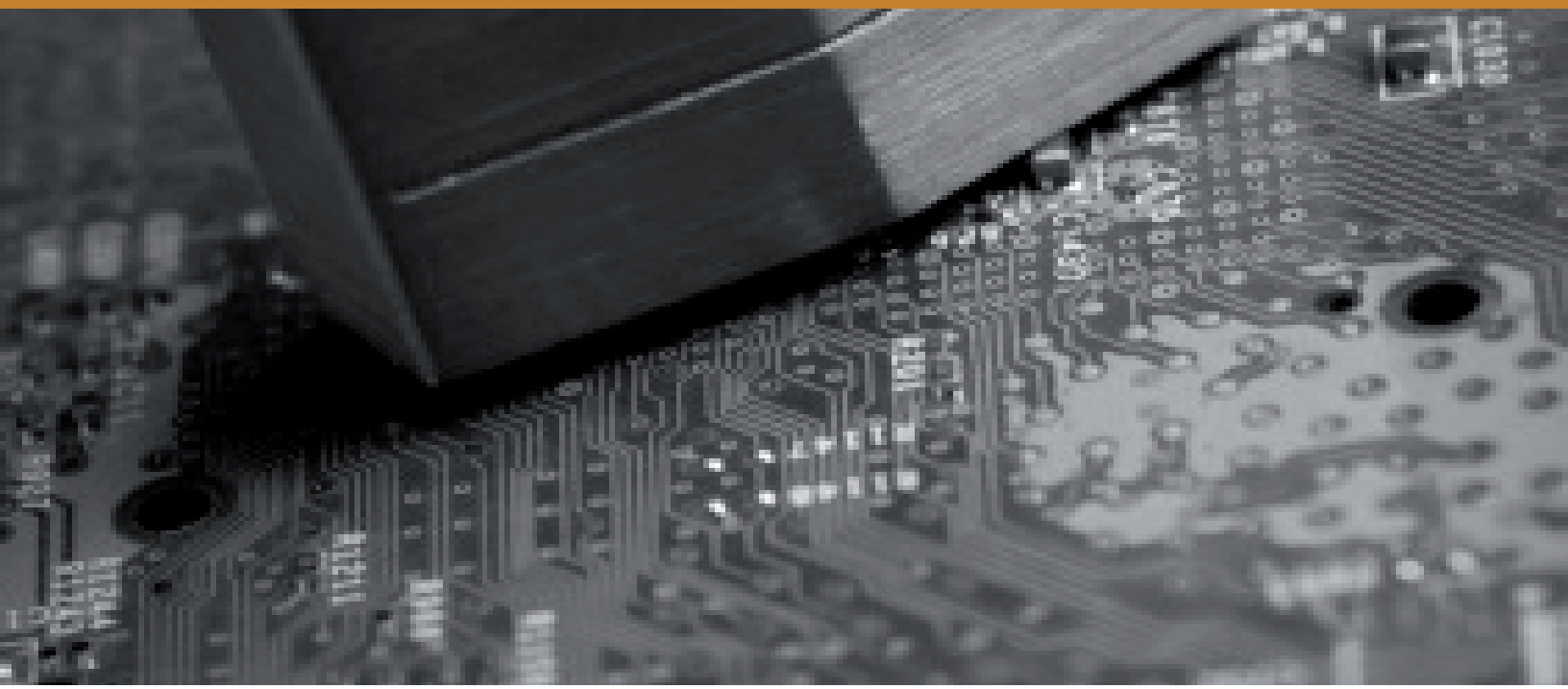
THE VISION

SOFTWARE ASSURANCE MARKETPLACE

A Transformative Force in the Software Ecosystem



SWAMP

SOFTWARE ASSURANCE MARKETPLACE

www.ContinuousAssurance.org

SWAMP
SOFTWARE ASSURANCE MARKETPLACE

Providing continuous assurance services to the marketplace

THE VISION

The SWAMP will become a national resource for software assurance (SwA) technologies, used across research institutions, non-governmental organizations and civilian agencies and their communities as both a research platform and a core component in the software development life-cycle.

It is our vision that widespread adoption of the SWAMP services will lead to a more secure and safer software ecosystem. □

THE SWAMP CHANNELS THE DHS MISSION

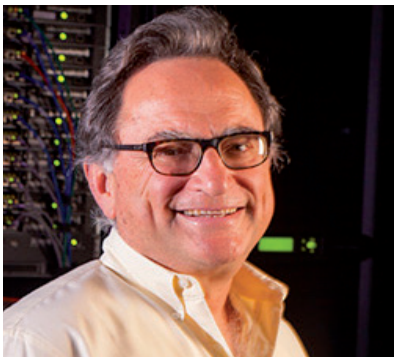
The Software Assurance Marketplace (SWAMP) channels the mission of the Department of Homeland Security Science and Technology Directorate's (DHS S&T) Cyber Security Division by:

- Developing and leveraging technologies, tools and techniques to defend and secure current systems to better protect critical infrastructures against attacks from our adversaries,
- Facilitating technology transition through a marketplace approach where a collection of innovative technologies can be harnessed by the community, and
- Providing a collaborative research environment by which DHS customers, agencies of the U.S. government, academia, private industry and international partners can exchange technical and research ideas to help advance software security and quality improvements.

Software has become such an essential component of our nation's critical infrastructure. It has grown in size, capability and complexity at a rate that exceeds our ability to keep pace with assuring its quality. The SWAMP provides opportunities through continuous assurance services to help narrow the gap that exists in the way software is tested and evaluated for security weaknesses and vulnerabilities. We expect that the marketplace will enable advancements and breakthroughs that will form new paradigms for software development activities. Closing the existing gap not only requires innovative technologies, but also more research in software analysis techniques and methods, as well as better awareness and education about security in the software development process.

"The Software Assurance Marketplace has been carefully constructed, developed and implemented with community feedback. It is with this approach we expect the SWAMP to be a revolutionizing force in the software assurance community for years to come. A software assurance marketplace is a great place for the community to meet for research collaboration and technical exchange. The concept of the marketplace has influenced and shaped the vision outlined in this document – ideally the vision is to provide a unique set of services and capabilities that can be leveraged by the community, creating a collaborative marketplace for continuous assurance." **Kevin E. Greene**, DHS S&T --Software Assurance Program Manager

BACKGROUND



The SWAMP research infrastructure currently under construction in Madison, Wisc. introduces a transformative element into the expanding fabric of national and international efforts to advance the state of the art and to increase the impact of Software Assurance technologies. The concept of this novel infrastructure was first introduced in a Broad Agency Announcement (BAA) published by the DHS S&T Directorate in January 2011 (02-11). The desired research infrastructure was presented in the BAA as *"… a software assurance facility and the associated services that will be made available to both software analysis researchers and software developers, both open source and proprietary. Software analysis researchers will have access to services allowing them to test new algorithms for static, dynamic, and binary analysis against a variety of software in a multi-platform environment. Software developers will gain maximum value through access and use of many software analysis tools, including those funded by DHS S&T, open source analysis tools, and potentially commercial tools, without having to acquire licenses or learn how to use each one individually. The SWAMP will become a national level resource in software assurance for open security technologies, used across civilian agencies and their communities as both a research platform and core component supporting US Government supported software development activities."*

Within the defined 5-year workplan, the SWAMP will achieve Initial Operating Capability (IOC) in January 2014. It is designed, implemented and operated by a team (page 5) led by the Morgridge Institute for Research (MIR) and includes the Indiana University, the University of Illinois at Urbana-Champaign and the University of Wisconsin-Madison. In a proposal submitted in July 2011, the team put forth the vision that *"… By offering SwA tool researchers, tool users and the educators who train our workforce, a rich suite of secure and dependable services for continuous assurance, the proposed SwA marketplace (SWAMP) will reduce the number of vulnerabilities in deployed software. Researchers who develop new SwA tools and methodologies will use the repositories and cyberinfrastructure offered by the SWAMP to improve their technologies and tools, while software developers and adopters will use the same services to hunt for vulnerabilities in their software. Educators will use these services to offer hands-on experience in SwA techniques to their students. By facilitating the sharing of tools, techniques, information, experiences and resources, the SWAMP will: 1) help advance the quality and adoption rate of SwA tools, 2) lower the threshold for using them, and 3) make it easier to interpret and use their output."*

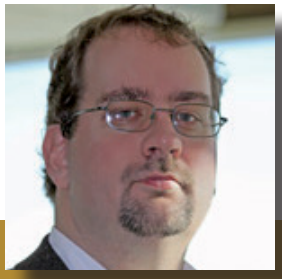
MORGRIDGE INSTITUTE FOR RESEARCH

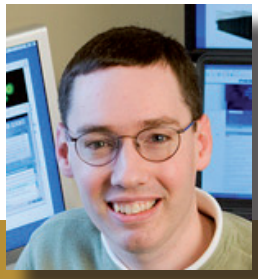
INDIANA UNIVERSITY
Center for Applied Cyber Security

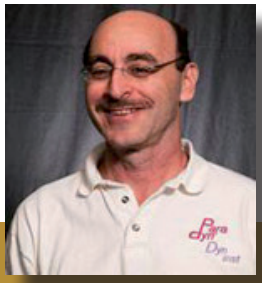
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE UNIVERSITY of WISCONSIN
MADISON

NCSA







From left: Von Welch, Security Officer; Jim Basney, Identity Management Lead; Barton Miller, Chief Scientist

MEET THE TEAM

The SWAMP staff and trusted advisors are thought leaders in their respective fields and bring resources from major academic institutions.

SWAMP is housed in and led by the Morgridge Insitute for Research and includes Indiana University, the University of Illinois-Champaign/Urbana and the University of Wisconsin-Madison. MIR will lead software development, infrastructure management, and facility operations. The University of Indiana focuses on security and 24x7 monitoring and user support. The University of Illinois heads identity management and Wisconsin handles SwA tool and software package integration. □

THE PROBLEM

Risks abound in our connected world

The rapid pace of software development has meant that in the last 20 years, nearly every facet of our lives is organized and controlled by billions of lines of code. No industry remains untouched by the pace of this evolution. Banking, healthcare and government entities all use software (both proprietary and open source) to create efficiencies and actionable data. The world has a network of connected ideas, services and products.

Along with the benefits included with this level of connectivity, there come a wide variety of hidden risks, ranging from the unintentional to the malicious exploitation of weaknesses. A new approach is necessary to mitigate the risk that untested software poses to businesses, organizations and our national security.

The Software Assurance Marketplace will address this problem by building a physical and virtual facility to help improve the quality of software that is being designed, both proprietary and open-source. This platform democratizes the SwA environment by providing a free, virtual market where the user community can come to share tools, techniques, resources and experiences with the goal of improving software assurance (reducing software vulnerabilities). The SWAMP will attract SwA tool developers, software developers, SwA researchers, educators and students and infrastructure providers.

The SWAMP will address these problems by: 1) helping advance the quality and adoption rate of SwA tools, 2) lowering the bar for using them for all communities, and 3) making it easier to interpret and apply their output.

"There's no perfect solution right now. The point and click security tools don't enable customization. And the tools that can be customized are either expensive or often, out of date."

Peyton Engel, JD, Security Assessment Team Leader for CDW, Inc.

Department of Homeland Security states....

"The nation's critical infrastructure (energy, transportation, telecommunications, banking and finance, and more), businesses and services are extensively and increasingly controlled and enabled by software. However, vulnerabilities in that software put those resources at risk. This risk is compounded by software size and complexity, the ways in which software is developed and maintained, the use of software produced by non-vetted suppliers and the interdependence of software systems. The President's "National Strategy to Secure Cyberspace (2003)" clearly states the need to "reduce and remediate software vulnerabilities. " http://www.dhs.gov/topic/software-quality-assurance

THE SOLUTION

Open source SwA platform advances tools, access

To address the risk of inadvertant or malicious events in software, our team from four academic institutions, with funding from the Department of Homeland Security Science and Technology Directorate, is building an open source SwA marketplace that is uniquely designed to vet and report on software weaknesses and vulnerabilities.

This five year project is led by the Morgridge Institute for Research in Madison, Wisc., which also provides a state-of-the-art, secure hosting facility. SWAMP's key collaborators from the University of Illinois-Champaign/Urbana, the University of Indiana and the University of Wisconsin-Madison contribute their expertise in software assurance, security, identity management, and community building. This unique collaboration will launch Initial Operating Capacity (IOC) in January 2014, followed by the rolling deployment of additional software, tools and capabilities scheduled through June 2017.

MORGRIDGE INSTITUTE FOR RESEARCH

INDIANA UNIVERSITY Center for Applied Cyber Security

ILLINOIS UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE UNIVERSITY of WISCONSIN MADISON

The design and implementation of the SWAMP will facilitate the exchange of knowledge within five target user groups: SwA tool developers, SwA researchers, software developers, educators and facility operators. Using the services provided by the SwA marketplace, these users will have access to a no cost, secure research facility in which they can test their methodologies and software, improve their technologies and analyze the results of assessment runs.

SWAMP's essential services, predicated upon the concepts of universal access, sharing and collaboration, will be a central element of the SwA and landscape.

SWAMP is located at MIR in Madison, Wisc.

continuousassurance.org/about-us/

COA

Continuous Assurance (CoA)

Driven by this vision, the SWAMP has been designed to enable reliable, reproducible, and automated assessments of software packages by SwA tools. As a national research facility, the SWAMP aims to maximize the adoption of the services it provides to the following five groups of users: software assurance tool developers, software developers, educators and students, infrastructure operators and software assurance researchers (figure below).

Continuous Assurance (CoA) takes the software engineering practice of Continuous Integration to a new level. CoA incorporates SwA tools into the frequent process of building and testing the software throughout its life cycle. CoA also applies the principal of frequent quality control to the development of SwA tools. CoA facilitates evaluation of the impact of incremental changes in a SwA tool on its ability to identify software defects.

This two-prong vision and implementation of the CoA practice address the key two challenges of the SwA community: improve the quality of SwA tools and expand the adoption of SwA tools.

Initial Operating Capabilties

At the time of IOC, SWAMP will provide SwA tools, software packages and computing platforms to support software static analysis. To stay up to date on our latest selected packages and tools, visit www.continuousassurance.org/technical-resources. For hardware, our facilities currently host 700 cores, 5 TB of RAM, 104 TB of HDD and 12 teraFLOPS which processes 12 trillion floating-point operations per second.

Source Code Analysis Tools

To help guide the selection of the five initial static SwA tools, a ranked list of common and critical weakness classes that are related to serious vulnerabilities was composed. This list was synthesized from published studies, such as the "CWE/SANS Top 25 Most Dangerous Software Errors," personal communications with practitioners in the field and the experiences of the team in performing in-depth, analyst driven software vulnerability assessments of a broad range of software packages.

The classes of weaknesses on which the SWAMP will initially concentrate are injections, buffer handling, information leaks, number handling and web deceptions. These classes were selected to try to satisfy several goals. First, they rate prominently on lists of most frequently occurring weaknesses. Second, they are ones that within the capabilities of the current best open source SwA tools. Third, they will provide a strong demonstration of the SWAMP's capabilities at IOC. As examples, specific types of weaknesses in these classes are:

Injections

- Command injections
- SQL injections
- Use of inherently dangerous operating system interfaces
- Buffer handling

Buffer overruns

- Resource leaks (allocated but not freed resources)
- Pointer usage errors (e.g., NULL pointer usage, double freeing, use after free)
- Format string attacks
- Integer overflow/truncation errors
- Cross-site scripting/Cross-site request forgery
- URL redirection (Open Redirect)

These weaknesses were selected to represent a few basic sources of common vulnerabilities: interfaces to systems external to the program, such as the operating system or a database system; basic coding errors; and web programming.

Guided by this list of weakness classes and by publicly available information about the capabilities and adoption of the various open source tools, the following five open source tools will initially be offered:

- **FindBugs** is the most widely used and accepted tool for Java source code analysis. In addition to its standalone use, many commercial tools incorporate FindBugs to enable Java code analysis. FindBugs addresses Java coding errors, and from the above list, injection and web attacks.

- **PMD** is a static analysis tool for Java, JavaScript, XML and JSP. It is a community wide tool under active development, hosted on sourceforge.net. The primary functionality of PMD is targeted towards web programming errors. While focusing on correctness and quality issues, PMD has an increasing number of security checks added to its functionality. PMD's main focus is Java, with limited support for JavaScript, XML and JSP.

- **Cppcheck** is a community-wide open source tool (hosted at sourceforce.net) that works for C and C++. It checks for common coding errors, problems using common class libraries and catching use of dangerous functions. Cppcheck should address weaknesses from the above list.

- **Clang and the Clang Static Analyzer** are part of the LLVM project and receives contributions from a wide community, including active participation from Apple and Google. Clang and the Clang Static Analyzer work on C and C++ (also Objective-C, used in Mac OS X and Apple iOS development). Clang and Clang Static Analyzer are targeted at basic coding errors.

- **Oink** is a static analysis tools for C and C++ that uses CQual++ to do its basic analysis. It is hosted on appspots.com. Oink is reputed to be quite solid and has strong whole program analysis abilities. The checks are based on what was done in its predecessor tool, CQual. It is designed to detect the potential use of dangerous user inputs and type errors. Oink should be able to find common programmer errors, especially those related to use of OS functions and some memory errors.

Together, these five SwA tools target a broad range of weaknesses and cover a diverse set of use cases. When used as group, they have the potential to serve as a valuable SwA instrument. They also offer a rich reference set for tool developers and researchers as well as an easy to use training tool for educators.

Software Packages

The availability of a rich set of software packages, including test suites, such as the National Institute of Standards and Technology's (NIST) Software Assurance Metrics and Tool Evaluation (SAMATE) Reference Dataset (SRD) or packages used in a NIST Static Assessment Tool Exposition (SATE), will provide a broad testing environment for tool developers. The developers of these packages are expected to become active users of the of SWAMP's services. The IOC target is to host at least 100 packages.

The current package selection includes Java, C and C++ packages. Currently, 25 Java Packages will be

assessed with two tools - FindBugs and PMD. Next, we will add C and C++ capabilities.
Package selection through the five-phase process can be tracked at - https://continuousassurance.org/technicalresources

Result Viewers

While most commercial analysis tools come with their own interface to view the results of the weakness analyses in a GUI or web interface, open source tools typically output their results in text form. The open source tool results, through plug-ins, can be viewed with an integrated development environment (IDE) to display the results alongside the source code. FindBugs, PMD, Cppcheck and the Clang Static Analyzer, can integrate with the Eclipse open source IDE, and the Clang Static Analyzer can integrate with Apple's Xcode IDE.

In addition, there are a couple of new tools for viewing integrated reports of weaknesses from multiple analysis tools. These viewing tools include CodeDX from Secure Decisions, TOIF (tool output integrated framework) from KDM Analytics and Data Access Technologies funded through DHS S&T Small Business Innovation Research (SBIR) program.

The SWAMP is designed to support a suite of such viewers to meet the needs and preferences of the different groups of users. In addition to viewers developed by the SwA community, the SWAMP will provide a simple result viewer. The output produced by the five tools will be parsed to individual weaknesses with location (filename and line number). A single software package can be assessed multiple times, using different tools and OS platforms. The multiple results can be merged, filtered and sorted in a common result viewer interface. The result viewer will support a tabular view or viewing the weaknesses alongside the source code.

Capacity

At IOC the SWAMP will have the capacity needed to analyze over 275 million lines of code per day, sufficient to analyze 100 software packages with over 5 tools on 8 different OS platforms. The SWAMP infrastructure is designed to scale this capacity as additional demand is generated through the addition of CPU cores and storage capacity.

Core Services

The core services needed to harness the capabilities of the SWAMP are:

Manage Accounts, Projects and Access Control. Secure access to the SWAMP is accomplished through the use of user accounts and an associated credential. In the SWAMP a project is used to group users working on a set of software packages or tools. An application to establish a project is subject to approval by the SWAMP. Once approved, the requestor becomes the project administrator and approves additional users to join the project.

Permission to perform operations is restricted by access controls. Initial access controls are course-

grained with access permitted to all members of the project, or all users of the SWAMP.

Manage Software Packages and SwA Tools. A core function is to bring software, including SwA tools, into the SWAMP and build the software. The user describes how to acquire the software (uploaded by the user, or fetched by the SWAMP from a remote server such as a web server or a version control system), how to build the software, and the OS platforms to perform the build(s). The SWAMP manages the acquisition and building of software on the selected OS platforms. The SWAMP also supports importing of assessment results that were generated on external resources.

In the case of a SwA tool, the user must also specify how to create an installer that is usable by the SWAMP to place the tool on the assessment OS platform to perform the assessment. After a SwA tool successfully builds and an installer is created, the tool developer can make the tool available to other members of the SWAMP.

If the build fails, the software developer is able to debug the failure using the output files or through interactive access to the virtual host environment where the build failed.

Assess a Software Package with a SwA Tool. To establish a new automated assessment, a user selects a combination of software packages, tools and OS platforms to use, and the SWAMP manages the execution of the requested assessment(s).

The user is able to start the assessment(s) immediately, or to schedule assessments as a one time or recurring task (supporting continuous assurance). The SWAMP manages placing the software package and installing the SwA tool on the OS platform and running the assessment. After the assessment completes, the results are stored in the SWAMP's results repository for review. When an assessment completes, a notification can be sent.

View Assessment Results and the Dashboard. After an assessment completes, the assessment is added to a list of assessments available for viewing. For each assessment the user can download or view the output produced during the run; data about the assessment including software package, tool, OS platform, time, duration, resource usage and summary results such as the number of weaknesses; and the weaknesses identified.

The user can select multiple assessments and view the merged results with viewers like CodeDX. The user can filter and sort the results to facilitate analyzing the results. The user can also select a weakness and view the source code at the location where the weakness occurred. The source code viewer will also mark other weaknesses found in the same file.

The users can view the status of their upcoming, ongoing and completed assessment runs along with summary results of successfully completed assessments through a dashboard display. The

IOC

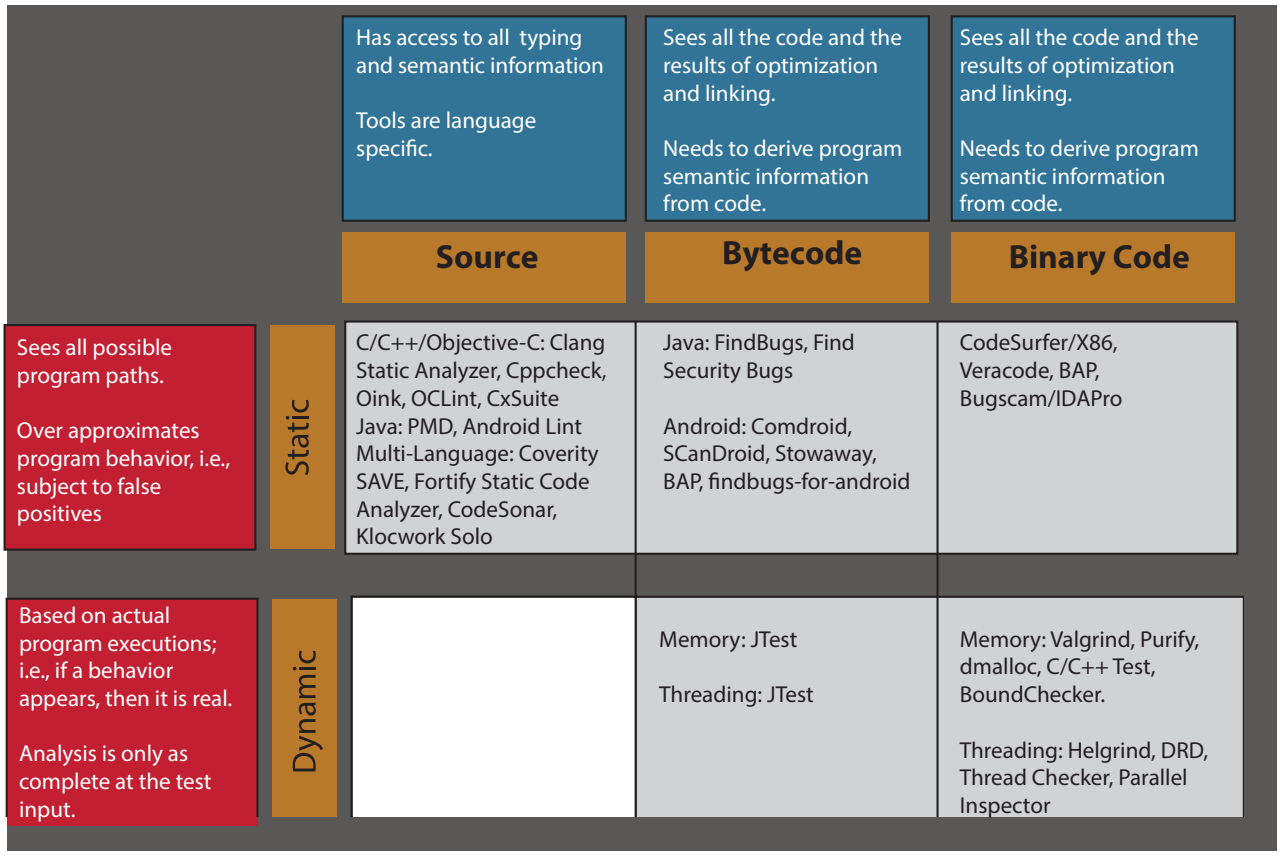


Figure 1: The SwA Tools Landscape

A representation of the types of tools used in software assurance and examples of such tools.

dashboard also facilitates periodic and large volume continuous assurance runs.

Future Capabilities

Following IOC, new capabilities and services will be added in a multistage process to the SWAMP. These new capabilities will leverage advances in SwA technologies and will address gaps in the needs and expectations of the five target users groups. These new capabilities and future services will offer a richer marketplace experience where ideas and results can be exchanged to improve software tools and software development activities. This will be a gradual process that will be guided by experience with the existing capabilities, advances in SwA technologies and community input. New tools and software packages will be added to the repositories, services and platforms will be expanded and the computing power will be increased. The portability of the software stack that powers the SWAMP will also be

addressed so that organizations that are interested in operating "private" instances of the SWAMP and the different repositories can easily do so. All the software needed to build and manage such an instance will be available under an open source license.

Expanding the Tools Repository

The power and diversity of the SwA tools in the SWAMP repository are key to the success of the SWAMP. As shown in Figure 1, SwA tools can be classified into five categories determined by two dimensions: using just the artifacts of the software (static) or observing the software's execution (dynamic), and the type of artifact used (source code, Java bytecode binaries, or native binaries). As previously described, the initial capabilities of the SWAMP will offer tools for the static analysis of source code and (in the case of Java tools) bytecode. This capability is the first step on a progression of tool technologies that will be added to the SWAMP following IOC. The road map for this progression is presented in Figure 2 that captures the introduction time and sustained enhancement of the different technologies. A detailed description of these technologies and currently identified gaps are presented on pages 19-22.

We will expand the space of tools in the SWAMP in several ways. First, we will increase the number of source code analysis tools to support new languages; second, we will add new platforms such as mobile; third, we will add dynamic analysis tools that operate by monitoring program behavior; and fourth, we will add static analysis tools that operate on binary programs. These new capabilities will be added during the first and second years of SWAMP operation.

Mobile Platforms

Mobile computing devices now outnumber the conventional computers (desktop and laptop). The most important of these mobile platforms are Google's Android, Apple's iOS and Microsoft's Windows Mobile. Recent numbers from Kantar Worldpanel Comtech report that in the U.S. market for phones, Android has 52% of the market to iOS's 43%; for tablets, Android has 54% of the market to iOS's 43%. Worldwide, the numbers are even more skewed in favor of Android.

Applications for Android are written in Java, using standard class libraries augmented by a substantial collection of Android-specific libraries for mobile use. The Java code is compiled conventionally into Java bytecode and then translated to an Android-specific bytecode called DEX that runs on the Dalvik virtual machine. Existing SwA tools provided by the SWAMP that support analysis of Java software can be used immediately on Android applications. However, most Java tools operate on bytecode, so require building the applications. As a result, the SWAMP will need to be able to support the Android build environment to run the majority of Java tools. Some tools like FindBugs, PMD, Klocwork, Lint, Parasoft and Veracode have specific modules or versions for Android. In addition, there are tools developed specifically for Android applications such as SCan-Droid, Stowaway, ComDroid, Androguard and Dexter. Use of these tools in the SWAMP will require the full Android application build environment.

Applications for iOS are written in Objective-C, an object oriented programming language that takes many of its features from the Smalltalk programming language. This language is also used heavily

within Apple for system development. Unlike Android, iOS applications are compiled directly to binary code, usually for an ARM processor. Similar to Android, the SWAMP would need to support the iOS development environment. There is the additional complexity that this environment runs only on Apple computers. Although few, there are source code analysis tools that assess Objective-C such as Clang Static Analyzer, CheckMarx and OCLint. While several projects are promising future support for analyzing ARM binaries, there are few current tools available. The current tools include BAP Binary Code Analysis, and zynamics' BinNavi.

Dynamic Analysis

Tools that monitor a program's execution (dynamic analysis tools) are an important complement to tools that statically analyze a program. Dynamic analysis tools have the strength that their results are based on the actual behavior of a program, not potential or speculated behavior. However, they only see behaviors of the program that are exercised by the input data. While static analysis tools naturally over approximate program semantics (so are subject to false positives), dynamic analysis tools can under approximate program semantics (so are subject to false negatives). The SWAMP is strengthened by the presence of both types of tools.

Dynamic analysis tools require the additional ability to have a controlled and monitored execution of the dynamic analysis tool and the application package. Given a successful build of the package, it must be launched in the correct execution environment with the necessary options and input data. The results of the dynamic analysis tool are collected, processed, and stored in the results database. The team's experience with the University of Wisconsin's Build and Test Lab, that provides infrastructure for supporting dynamic analysis tools, will be valuable as we add these capabilities to the SWAMP.

Some of the dynamic analysis tools considered include: Borland DevPartner (for detecting memory errors), Dmalloc (for detecting heap memory misuse), Helgrind (for detecting races and thread errors), Intel's Thread Checker (for detecting races and thread errors), JTest (a tools for detecting memory, threading, exception, resource and security vulnerabilities in Java programs), Parallel Inspector (for detecting thread errors), Parasoft Purify (for detecting memory errors) and Valgrind's Memcheck (for detecting memory errors).

Binary Code Analysis

While source code analysis tools have the advantage of access to the full semantics of the program expressed in a high-level programming language, there are many transformations done to a program while compiling, optimizing and linking. As such, the binary code may have subtle (and sometimes not-so-subtle) semantic differences from the source code. The binary code is the final product of the build process, so all ambiguity about compiling and linkage options have been resolved at this point. Tools that operate on binary code have a more difficult job in analyzing the semantics of the program but produce a more complete and authoritative result. In addition, binary code analysis tools have the advantage that they usually are independent of the source code programming language (though, of course, they are dependent on the processor architecture of the binary code). Another advantage they

have is the ability to analyze software where the source code is not available.

Supporting binary code analysis requires that the user provides an executable or that the software package be buildable, i.e., compiled and linked to produce an executable file. The requirement to build a package is a small additional requirement over source code analysis tools. As part of our infrastructure to support packages with complex build structures, we will design and implement the BuildTrack tool that monitors the key compiling steps during the building of a package and automates the application of tools to that package. The build process must be tracked to detect executable files produced. If a user provides or the build produces executables, the binary tool assesses the executables. The results of such a tool will be tracked in the same way as results from a source code analysis tool.

Some of the prospective binary analysis tools that will be considered include: BAP ("Next Generation Binary Analysis Platform", successor to Vine), Black Duck (a commercial tool that analyzes binary or source code for the use of third party intellectual property), BugScam (scripts for the IDA Pro binary code analysis and viewing tool), GrammaTech's CodeSurfer/X86 (for analyzing and rewriting binaries), protecode (a commercial tool that analyzes binary or source code for the use of third party intellectual property and security issues), Veracode (bi-nary code as a service) and Vine (BitBlaze's static analysis component).

Commercial analysis tools

The team will continue to explore ways to engage commercial tool providers in including their tools in the SWAMP. Our architecture and services were designed and have been implemented to address the special needs of such tools.

Result Viewing

Each weakness detected by a tool will be normalized to a common weakness type enabling comparisons between tools. Support for triage information will be added as well as enhancements to the source code viewer to support source code navigation functions to assist users in verifying weakness results.

Build the SWAMP Ecosystem

Software assurance is a very broad and active area. The long-term success of the SWAMP research facility will depend on our ability to build a broad eco-system that leverages the services of other projects and organizations. The architecture of the SWAMP follows the principals of distributed computing and will therefore easily interface to SwA capabilities and computing capacity provided as a service using cloud interfaces and technologies. Elements of such an ecosystem may include among others:

Repository of Certified Open Source Software

The Repository of Certified Open Source Software offers open source software packages that "passed" the SwA analysis tools of the SWAMP. This repository will operate as an independent project/service that will use the SWAMP to certify the packages. An example for such a service is the repository of

components operated by Sonatype.

Continuous Compliance Validation

The SWAMP will provide continuous assurance services that will help meet application security compliance guidance such as (but not limited to): NIST 800-53 controls, Defense Information Systems Agency (DISA) Application Security and Development Security Technical Implementation Guides, Payment Card Industry (PCI) Payment Application Data Security Standard, as well adherence to industry regarded best practices like SANS Top 25 Most Dangerous Software Errors and Open Web Application Security Project (OWASP) Top 10. The SWAMP will play a key role in ensuring that all systems and applications that support government and critical infrastructure are not only secure, but meet applicable compliance guidance to ensure proper coding and secure development practices are being followed.

Consumer Report for SwA Tools

The SWAMP's structure of automating the running of a collection of analysis tools against a collection of software packages can provide the basis for a facility evaluating the effectiveness of the analysis tools. The basic mechanism for supporting this evaluation is the incorporation of calibration reference analysis data (the ground truth) into the analysis results for a given software package. Supporting this calibration reference data requires three major features: (1) sources of authoritative reports of weaknesses or (better) vulnerabilities in a standard form, (2) a pseudo-analysis tool that incorporates these reports into the analysis of a given package and (3) the addition of calibration factor in the range of [0, 1] for each weakness report.

Sources of Authoritative Weakness and Vulnerability Reports

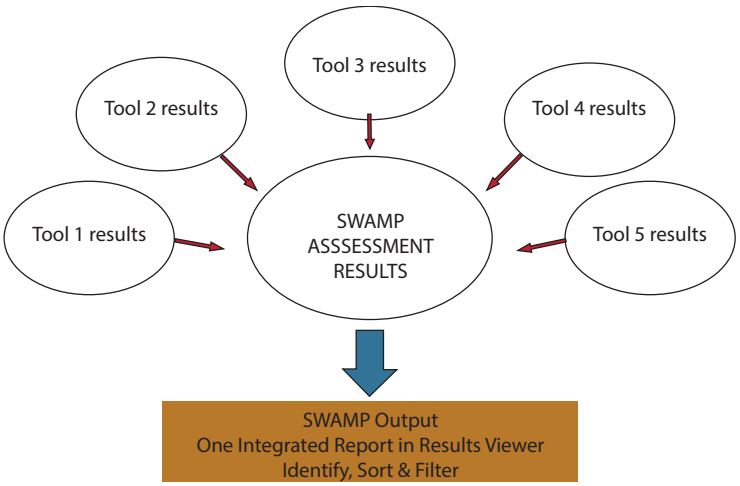
Authoritative weakness or vulnerability datasets consist of two parts: the packages to which the reports apply and the reports themselves. Several potential sources of authoritative weakness or vulnerability data have been identified. The first source is packages used in a NIST SATE, which includes a wide variety of different types of test packages (from very small and simple to larger and more realistic). The weakness reports associated with these packages are documented in NIST's SAMATE format. The second source is the UW Middleware Security and Testing (MIST) project's verified vulnerability dataset. This dataset applies to full packages that have been subjected to indepth analyst driven assessment and have verified exploits for each identified vulnerability. These vulnerabilities also have the interesting characteristic that they are beyond the ability of the best current tools to detect defects. The MIST project is producing a "redacted" version of each vulnerability, i.e., a simplified version of each vulnerability isolated from the package from which they were derived. The third potential source will come from tools that have verified behaviors for the results that they generate. The fourth source is the NIST SRD, which consists of a collection of mainly synthetic and small programs with known weaknesses or known to be weakness free.

Note that the calibration data does not have to be complete to be effective in the SWAMP. One of the strengths of the SWAMP is that it can incorporate data from a variety of sources, and each new

addition. Meaning, we can incorporate data whether it 1) comes from a source that can report only a limited set of vulnerabilities or 2) from a source that can report vulnerabilities for a limited number of packages or 3) just limited parts of a package. Each additional result strengthens the overall result.

Incorporating Reference Results in the SWAMP

Incorporating reference analysis results into the SWAMP will be accomplished by creating a new "oracle" tool that presents the reference results in a way similar to a normal tool. It will take reports in the SAMATE format and present them as results from the oracle tool.



A Calibration Factor for Analysis Tool Results

Introducing a calibration factor for tool results provides a way to label the strength of results. Initially, it can be used to label reference results with a calibration factor of 1. In the longer term, as the reference data to evaluate analysis tools is obtained, it will be used to label the quality of the results of those tools. A cumulative view of the calibration factors for a given analysis tool can form the basis evaluation of that tool.

To make this calibration activity effective, novel techniques will be required to compare the calibration results to those of actual tools. We will need to summarize them in a way that gives an accurate view of the capabilities of an analysis tool, and visualize the results in a way that is intuitive and easy to understand for large, complex code bases. Developing such techniques and deploying the associated tools will require leveraging of results from other organizations.

IOC AND BEYOND

The following subsections present the staged process of introducing new functionalities to the SWAMP. Following the IOC, which is scheduled for January 2014, each following stage will target one key functionality and a number of major enhancements to existing capabilities. Two of the stages target new capabilities in addition to the key functionality. For each functionality and enhancement, candidate technologies are listed and technology gaps identified.

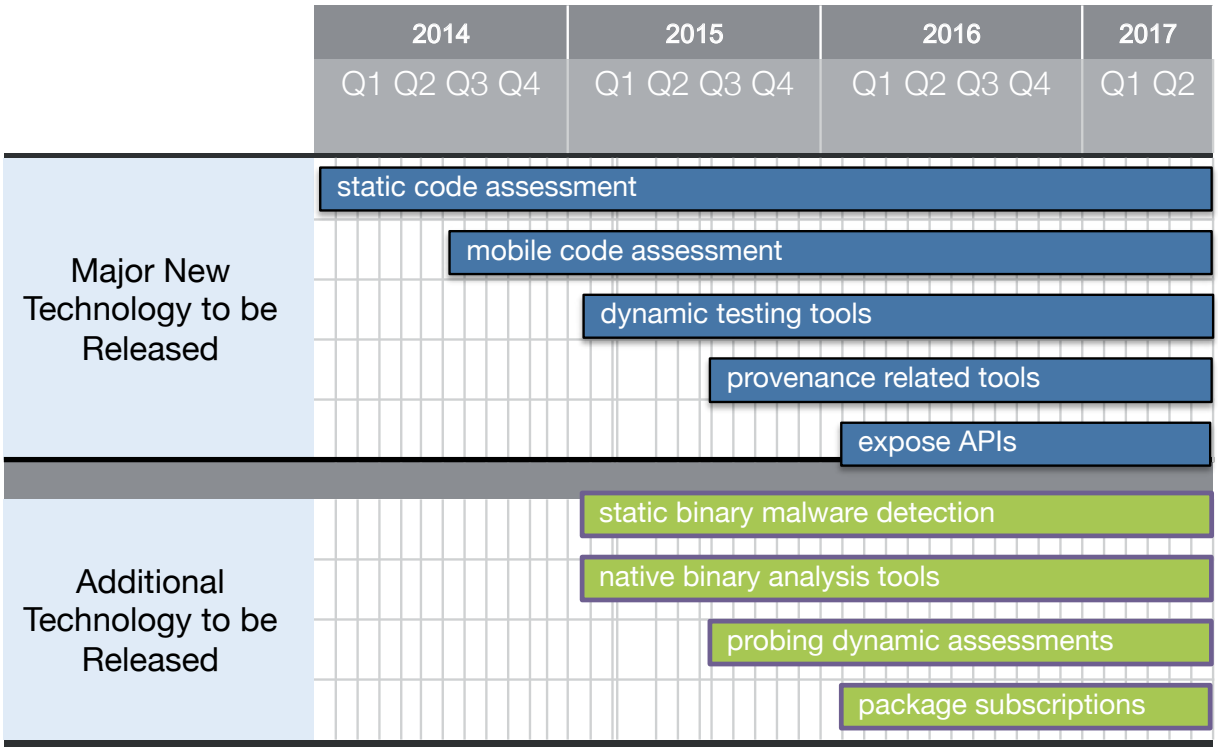


Figure 2: The SWAMP Roadmap post IOC

IOC + 0

Major New Functionality:

- Support static code assessment (SCA) tools.

  Technology gap – There is a considerable gap between the commercial tools and the open source tools. In general open source tools do not perform whole program analysises that impede their ability to accurately detect many types of weaknesses. Even the commercial tools report a significant number of false positives and fail to report many weaknesses.

IOC + 6 months

Major New Functionality:

- Support mobile code assessment. Support the assessment of Android and iOS mobile applications.

  Candidate technologies – Android: Android Lint, Comdroid, SCanDroid, Stowaway, BAP, findbugs-for-android; iOS: OCLint, Clang Static Analyzer, CxSuite*.

  Technology gap – Similar to SCA tool above, but tools are less mature.

Enhancements:

- Normalized SwA tool results--allows uniform tool results.

  Candidate technologies---in house development.

- Triage support--supports recording and filtering of human assessment of results.
  *Candidate technologies --i*n house development.
- Finer-grained access controls.
  Candidate technologies --in house development.

IOC + 12 months

Major New Functionality:

- Support dynamic testing tools--enables additional testing techniques based on executing the code, such as QA testing, resource and memory error detection, test coverage, and performance measurement.
  Candidate technologies – Memory: Valgrind, dmalloc, Purify*, C/C++ Test*, JTest*, BoundsChecker*; Threads: Helgrind, DRD, ThreadChecker*, Parallel Inspector*; Performance: Cachegrind, Shark*, Memcheck, Quantify, gprof, gperftools, oprof, Intel VTune*, Parasoft Insure++*, JTest*; Coverage: gcov, CoverStory*, CodeCover, Cobertun, jcoverage/gpl, Jester, Quilt, Hansel, JVMDI Code Coverage Analyzer, GroboCodeCoverage, InsECT.

New Functionality:

- Support static binary malware detection--detect malware by inspecting the applications produced by the software package.
  Candidate technologies – Clam AntiVirus, Pyew, Avast!*, AVG*, Avira*, BitDefender*, Comodo, F-Prot*.
  Technology gap –although there are many methods and tools to aid in malware detection, there is a lack of tools to detect malware embedded in source code. Another useful capability would be the ability to detect insider malicious modification to source code, such as insertion of malware like functionality such as back doors and time bombs.
- Support native binary analysis tools--enable assessments or software in binary form with source code.
  Candidate technologies – BAP, Bugscam/IDAPro*, CodeSurfer/X86*.
  Technology gap – there are few static binary analysis tools that operate in an automated fashion.

Enhancements:

- Interversion triage support--automatic realignment of triage location as source code changes.
  Candidate technologies – In house development.
- Source code navigation features--assist human analysis of assessment results.
  Candidate technologies – in house development.
- Enhanced dashboard support--add trendline and summary information.
  Candidate technologies – in house development.

IOC + 18 months

Major New Functionality:

- Provenance related SwA tools. Use of secure hashes to detect copied code and alert users when software components they are using have known defects.
  Candidate technologies – In house development.
  Technology gap – There are few tools to assess source code and binary code to determine provenance.

New Functionality:

- Enable probing dynamic assessments--external assessment tools that operate by connecting to a system such as penetration and fuzz testing tools.
  Candidate technologies – OpenVAS, nmap, w3af, sqlmap, Netsparker, Metasploit, DirBuster, monkey, Skipfish, WFuzz, Wapati, Wikto, Websecurify, Grendel-Scan, Nikto AppScan*, Acunetix WVS*, HP WebInspect*.
  Technology gap – Fuzzing and penetration tools benefit from human direction. There could be work done to automate the usage of these tools perhaps by using the source code as a guide or by building more intelligence into the tool.

Enhancements:

- Enhance support for educators--simplify roster management, class setup and scheduling tasks.
  Candidate technologies – In house development.

IOC + 24 months

Major New Functionality:

- Expose APIs to interact with the SWAMP--the APIs will include support to import and export data, notifications to external systems when events occur in the SWAMP, notifications from external systems to trigger actions in the SWAMP, and connections to external software development system such as version control system, issue tracking systems, and IDEs.
  Candidate technologies – In house development.

New Functionality:

- Support subscriptions for software packages--automatically assess packages when new versions of the software or SwA tools are available.
  Candidate technologies – In house development.

Enhancements:

- Enhance summary result reporting--give non-software developers simplified metrics about the quality and security of the software.
  Candidate technologies – In house development.
  Technology gap – There are no tools that can take SwA results and reduce them to quantifiable results that non-technical users can use to determine the quality and security of a software package.

- Enhance access control to support 3rd party assessments
  Candidate technologies – In-house development

Post-IOC + 24 months
Technology Gaps in SwA Tools for Future SWAMP Capabilities:

- Tools to determine the structure of a system--dynamic analysis, possible in conjunction with static analysis, can be used to determine the structure of a software system including the architecture, resources, privileges, and trust boundaries. Security analysts could use this as an initial step in performing a manual assessment or for anyone wishing to understand how a software system is structured.
- Hybrid static/dynamic analysis tools--develop models of combining information from static and dynamic analyses and construct tools based on those models.
- Develop databases of exploits and vulnerabilities to focus new tool development--exploits and vulnerabilities need to be gathered from real-world assessments and potentially from mining bug repositories. These test cases present an immediate focus for new and evolving analysis tools.
- Analysis of installation and configuration--analysis of the installation and configuration of a software system.
- Whole system static and dynamic analysis--analyzing dataflow across process and system boundaries should improve the results of SwA tools

Providing continuous assurance services to the marketplace

Morgridge Institute for Research
330 North Orchard Street
Madison, WI 53715
Phone: 608.316.4300

Primary Program Contact

Patrick Beyer, PhD, PMP
pbeyer@continuousassurance.org
608.509.5203

ContinuousAssurance.org

Join our Software Assurance Marketplace LinkedIN Group

Follow SWAMP on Twitter @swampteam