A Vision for the Software Assurance Marketplace

A Transformative Force in the Software Assurance Ecosystem

Version 2.0

Jim Basney, Irene Landrum, Miron Livny, Bart Miller, Von Welch
MORGRIDGE INSTITUTE for RESEARCH

330 N. Orchard St.
Madison, WI 53715
swamp@continuousassurance.org

SUPPORT
support@continuousassurance.org
1-608-316-4266

WEBSITE
continuousassurance.org
mir-swamp.org

SOCIAL
GitHub: github.com/mirswamp/deployment
Twitter: twitter.com/SWAMPTEAM
Facebook: facebook.com/SoftwareAssuranceMarketplace
LinkedIn: linkedin.com/groups/4715681
The SWAMP vision provides an ecosystem of Software Assurance capabilities guided by the fundamental principles of Continuous Assurance - an essential component of the software development lifecycle - provided by the SWAMP-in-a-Box (SiB) software distribution and anchored by the MIR-SWAMP facility. These ecosystem impacts include workforce development through education and training, bringing software assurance to high-security environments through on-premise software assurance, enhancing Cyber Supply Chain Risk Management programs by adding the ability to assess software, and enabling continuous assurance as an extension of continuous integration including integration into existing software lifecycles such as DevOps.
WELCOME TO THE SWAMP

WHO WE ARE

The Software Assurance Marketplace (SWAMP) delivers an integrated set of software assessment tools, with automation, web-based interfaces, and integration with IDEs, software repositories, and continuous integration frameworks to enable Continuous Assurance (CoA) during the software development lifecycle.

OUR REACH

SWAMP has influenced and increased the adoption of software assurance through collaborations or usage by Code Dx, ThreadFix, GrammaTech, Parasoft, Programming Research, Sonatype, Synopsys, UC-Irvine, Raytheon, Bowie State University, SANS, Indiana University, FAA, NASA, Open Science Grid, and CERN. SWAMP provides a software distribution, called SWAMP-in-a-Box (SiB), that is anchored by a free-to-use software assurance facility, called MIR-SWAMP.
WELCOME TO THE SWAMP

OUR COLLABORATION

To address the risk of inadvertent or malicious events in software, our team from four academic institutions, with funding from the Department of Homeland Security Science and Technology Directorate, built an open source software assurance (SwA) marketplace that is uniquely designed to vet software to report on software weaknesses and vulnerabilities. Led by the Morgridge Institute for Research in Madison, Wisconsin, the project provides a shared, secure hosting facility. SWAMP’s key collaborators (below) contribute their expertise in software assurance, security, identity management, and community building.
THE FUTURE OF CYBERSECURITY

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. The ubiquity of software can be seen in nearly every part of our lives including medical devices, self-driving cars, Internet of Things (IoT), banking, voting, etc.

In 2017, 111 billion new lines of code will need to be secured. Along with the benefits included with this level of connectivity, there comes a wide variety of security issues, ranging from the unintentional to the malicious exploitation of weaknesses leading to security breaches such as the multiple breaches at Yahoo or the Heartbleed vulnerability that went undetected for two years.

A new approach is necessary to mitigate the risk that untested software poses to businesses, organizations, and our national security.

:::> THE TIME IS NOW

“The Software Engineering Institute estimates that 90 percent of reported security incidents result from exploits against defects in the design or code of software. Ensuring software integrity is key to protecting the infrastructure from threats and vulnerabilities and reducing overall risk to cyber attacks.”

1. us-cert.gov/sites/default/files/publications/infosheet_softwareassurance.pdf
CONTINUOUS ASSURANCE (CoA)

SWAMP introduced the idea of continuous assurance, extending the concept of continuous integration into the software assurance space. The goal is to naturally integrate the security assessment of software into the software development workflow. CoA incorporates software assurance tools into the process of building and testing the software throughout its lifecycle. Incremental changes to software are immediately evaluated for their impact on software assurance, and fixes to these security problems are immediately applied.

This tight integration of software development and software assurance can dramatically reduce the number of security flaws that appear in deployed software and reduce the cost of fixing the security flaws that are found. It also allows for secure rapid development cycles, such as found in DevOps.\(^2\)

The SWAMP was established to support continuous assurance, helping to simplify and automate the process of running code analysis tools, especially static code analysis (SCA) tools. SWAMP can be integrated easily into the continuous assurance workflow, providing direct access from integrated development environments (IDEs) such as Eclipse, source code management systems such as git and Subversion, and continuous integration systems such as Jenkins.

::: CoA COVERAGE

- **NEW RELEASES**
- **NEW PROJECTS**
- **IN DEVELOPMENT**
- **LEGACY CODE**
- **THIRD PARTY**
- **OPEN SOURCE**

---

2. en.wikipedia.org/wiki/DevOps
CONTINUOUS ASSURANCE INTEGRATION MODELS

The goal is to organically integrate the security assessment of software into both the Waterfall and DevOps software development workflows.

CLASSIC

In the classic Waterfall Methodology, each stage is completed before proceeding to the next stage. The SWAMP supports the implementation, testing, and maintenance stages.

DEV-OPS

The continuous assurance model also fits into the DevOps model as part of the Verify phase. The addition of continuous assurance adds software assurance and security to the DevOps functional testing process.

OUR AUDIENCE

SWAMP has been designed to enable reliable, reproducible, and automated assessments of software packages by SwA tools.

As a national research facility, 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.
SOFTWARE IN THE SWAMP

The SWAMP offers 27 open source and 4 (as of 10/19/17) commercial code analysis tools that target a broad range of weakness classes. Through the automation services provided by the SWAMP, users are able to harness the power of a scalable, virtualized environment to apply these tools, or private tools they can bring with them to the SWAMP, to large numbers of software packages.

The SWAMP offers more than 500 such software packages, including NIST’s JULIET test suite, and the ability to view assessment results via viewing tools that support integrated reports.

All the software components produced by the SWAMP project are open source, so as to encourage the broadest adoption of these practices in both the commercial and research communities. This is an ongoing process, and we (and many other groups) continue to strive to develop new ways to streamline the inclusion of software assurance into the development process.

The SWAMP tools target a broad range of weaknesses and cover a diverse set of use cases. When used in combination, they 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 education and training tool for educators.

The SWAMP tools have been selected by the generation of a ranked list of common and critical weakness classes that are related to serious vulnerabilities. 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 class of 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.
CONTINUOUS ASSURANCE: SOFTWARE TOOLS

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 thirty-one tools (next page) are offered, including open source tools and commercial tools.

:::> SUBSET OF WEAKNESSES

THE SWAMP CAN DETECT

COMMAND INJECTION: Allow an attacker to control the text that the program passes to a system shell. This allows the attacker to execute arbitrary commands on the target system.

SQL INJECTIONS: Allow an attacker to control the text that the program passes to a SQL database. This allows the attacker to execute arbitrary SQL statements, or to change the results of SQL queries.

USE OF INHERENTLY DANGEROUS OS INTERFACES: An inherently dangerous OS interface generally results in code with security or quality weaknesses. Although some of these functions can be used safely with additional checks to validate the parameters and operational environment, these checks are complex and error-prone. Users should use alternative safer interfaces.

BUFFER OVERRUNS: Allow an attacker to write outside the bounds of a memory buffer. This affects the integrity or discloses the program’s runtime environment or program data, allowing an attacker to modify the state of the program in unexpected ways including executing arbitrary code.

RESOURCE LEAKS (ALLOCATED BUT NOT FREED RESOURCES): A weakness where unneeded resources are not reclaimed. An attacker can use this weakness in a denial of service attack.

POINTER USAGE ERRORS (E.G., NULL POINTER USAGE, DOUBLE FREEING, USE AFTER FREE): Pointer usage errors are an improper use of a memory pointer. This can result in a buffer overrun, a resource leak, or corruption of program state.

FORMAT STRING ATTACKS: Format string attacks are an injection of user controlled data into format strings used to control input or output. This allows an attacker to corrupt program or runtime state, or to disclose sensitive data on an output channel.

INTEGER OVERFLOW/TRUNCATION ERRORS: Integer overflows or truncation errors are the failure to correctly handle values that are outside the finite range of integers used in many programming languages. When a value (or result of an arithmetic operation) is outside the finite range of the variable, the value can be unexpectedly too small or large and can result in the corruption of program state or disclosure of sensitive data.

CROSS-SITE SCRIPTING/CROSS-SITE REQUEST FORGERY: Cross-site scripting (XSS) allows an attacker to inject text into a web page that is viewed by another user. This allows an attacker to run arbitrary scripts in the victim’s web browser, can trick the user into loading malicious content, or trick the user into clicking malicious links.

CROSS-SITE REQUEST FORGERY (CSRF): Allows an attacker to cause a victim’s web browser to visit an exploitable web page that performs an unwanted operation as the user. The exploitable web page normally validates that the user is authenticated but fails to validate that request came from a valid chain of pages starting with authentication. This is often used in conjunction with XSS and allows the attacker to perform malicious operations.

CROSS-SITE REDIRECTION (OPEN REDIRECT): A URL redirection allows an attacker to inject content into a redirection response from a web server, allowing the attacker to control the eventual web page that is loaded by the victim’s web browser. This can be used in phishing attacks and in general to mislead a victim as to the actual web page being loaded.
COMMERCIAL & OPEN SOURCE TOOLS

WHEN USED IN COMBINATION, THESE TOOLS SERVE AS A VALUABLE SWA INSTRUMENT OFFERING BOTH A RICH REFERENCE SET FOR TOOL DEVELOPERS AND RESEARCHERS, AS WELL AS EASY-TO-USE EDUCATION AND TRAINING TOOLS FOR EDUCATORS.

ANDROID

**ANDROID LINT**- Static code analysis tool that checks Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.

**REVEALDROID**- Analyzes Android APK files for known malware families using machine learning. The development of this tool was originally funded by DHS S&T.

C/C++

**CLANG AND THE CLANG STATIC ANALYZER**- Part of the Clang and LLVM compiler project. It analyzes C, C++, and Objective-C for correctness and security weaknesses.

**CPPCHECK**- Detects the types of bugs that compilers normally do not detect verses detecting syntax errors in the code. Cppcheck checks for common coding errors, problems using common class libraries, and catching use of dangerous functions.

**GCC: THE GNU COMPILER COLLECTION (GCC)**- Originally written as the compiler for the GNU operating system. GCC’s extensive diagnostic messages report on risky and error-prone code practices.

**GRAMMATECH’S CODESONAR**- Static analysis tool that identifies bugs and security vulnerabilities in C/C++ source code. The development of concolic testing in this tool was originally funded by DHS S&T through an SBIR award.

**PARASOFT’S JTEST**- Static code analysis for applications written in Java to improve software quality and to prevent defects that impact application security, reliability, and performance.

**PMD**- Static analysis tool for Java source code, finding common programming flaws like unused variables, empty catch blocks, and unnecessary object creation. It finds correctness and security weaknesses in the use of Java and libraries.

**PYTHON**

**BANDIT**- Analyzes Python source code to identify security weaknesses and correctness.

**FLAKE8**- A tool that wraps together pycodestyle, pyflakes, mccabe, and third-party plugins. It analyzes Python source code for conformance to the standard Python coding style PEP 8 and for common coding errors.

**PYLINT**- Analyzes Python source code for conformance to the standard Python coding style PEP 8 and for common coding errors.
COMMERCIAL & OPEN SOURCE TOOLS (CONT.)

RUBY

**BRAKEMAN**- An open source vulnerability scanner specifically designed for Ruby on Rails applications. It statically analyzes Rails application code to find security issues.

**DAWNSCANNER**- Analyzes Ruby source code. It also detects problems with the incorrect usage of the MVC web application frameworks Ruby on Rails, Sinatra, and Padrino.

**REEK**- Analyzes Ruby source code. It reports on code and style issues that are commonly misused or lead to maintenance issues.

**RUBOCOP**- Analyzes Ruby source code. It checks for failures to follow the guidelines outlined in the community Ruby Style Guide. The style guide is a combination of lexical conventions and idioms to improve the code's correctness, efficiency, and maintainability.

**RUBY-LINT**- Analyzes Ruby source code. It reports weaknesses related to commonly misused coding idioms.

WEB

**CSS LINT**- Analyzes CSS (cascading style sheet) files. It provides basic syntax checking and a set of rules to the code that look for problematic patterns or signs of inefficiency.

**ESLINT**- Analyzes JavaScript source files. It looks for common indicators of incorrect code and for non-conformance to best practices.

**FLOW**- Analyzes JavaScript source files for type correctness.

**HTML TIDY**- Analyzes HTML source files. It validates HTML documents for correctness and issues with legacy versions of HTML.

**JSHINT**- Analyzes JavaScript source files. It looks for common indicators of incorrect code and for non-conformance to best practices.

**PHPMD**- PHP Mess Detector analyzes PHP source files. It looks for common indicators of incorrect code and suboptimal code.

**PHP CODE SNIFFER**- A tool that scans PHP source code, JavaScript, and CSS files to detect and fix violations of a defined set of coding standards.

**XML LINT**- A tool that verifies that XML documents are well-formed, or syntactically correct. It is part of libxml2.

DEPENDENCY CHECKERS

**OWASP DEPENDENCY-CHECK**- An analysis tool that identifies project dependencies and checks if there are any known, publicly disclosed vulnerabilities for Java. The library versions and signatures of source files are looked up in a database to determine vulnerabilities.

**RETIRE.JS**- An analysis tool that identifies project dependencies and checks if there are any known, publicly disclosed vulnerabilities for JavaScript. The library versions and signatures of source files are looked up in a database to determine vulnerabilities.

DEPENDENCY CHECKER

**SONATYPE APPLICATION HEALTH CHECK**- An analysis tool that identifies project dependencies and checks if there are any known, publicly disclosed vulnerabilities for Java. The library versions and signatures of source files are looked up in a database to determine vulnerabilities.

C/C++

**PROGRAMMING RESEARCH STATIC ANALYZERS FOR C AND C++ (QA·C AND QA·C++)**- Tools that analyze C and C++ code, respectively, to identify weaknesses related to correctness and security.

UPCOMING CoA ADDITIONS (AS OF NOV 2017)

**DEPENCY CHECKER**

**SONATYPE APPLICATION HEALTH CHECK**- An analysis tool that identifies project dependencies and checks if there are any known, publicly disclosed vulnerabilities for Java. The library versions and signatures of source files are looked up in a database to determine vulnerabilities.

**C/C++**

**PROGRAMMING RESEARCH STATIC ANALYZERS FOR C AND C++ (QA·C AND QA·C++)**- Tools that analyze C and C++ code, respectively, to identify weaknesses related to correctness and security.
SOFTWARE PACKAGES

The availability of a rich set of more than 500 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), provides a broad testing environment for tool developers. A complete list of available packages can be found at https://www.mir-swamp.org/#packages/public.

RESULTS 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. Some 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.

The SWAMP provides a native result viewer. The output produced by tools is parsed to individual weaknesses with location (filename and line number). This viewer displays results of a single assessment, while multiple viewers can be opened to view the results from multiple tools. The native viewer presents a tabular presentation of the weaknesses discovered by the assessment showing the location in the source code and the description of the weakness. Additional information can be displayed based on the type of tool. Users can sort the table based on any of the columns.
In addition, two commercial viewers have been integrated into the SWAMP and are available for use:

**CODE DX™:**
An integrated results viewer, Code Dx is a software assurance analytics tool that consolidates, normalizes, prioritizes, and displays the weaknesses detected by disparate tools in a centralized viewer. The research and development of this viewer was originally funded by DHS S&T.

**THREADFIX:**
The Denim Group’s ThreadFix results viewer was integrated with the SWAMP in 2016. ThreadFix is a software vulnerability aggregation and management system providing a centralized view of software security defects. The ThreadFix Community Edition allows users to view assessment results from the Clang Static Analyzer tool run against C/C++ software packages in the SWAMP.
SWAMP COMPUTING CAPACITY

The SWAMP has the capacity needed to analyze over 275 million lines of code per day, sufficient to analyze hundreds of software packages using multiple tools across a variety of OS platforms. The SWAMP infrastructure is designed to scale as additional demand is generated through the addition of CPU cores and storage capacity.

Following Defense Advanced Research Projects Agency’s (DARPA’s) Cyber Grand Challenge in 2016, a server rack was transitioned to the SWAMP for use by the high-performance computing community. The DARPA hardware adds an additional 1,280 cores, 16 TB of memory, and 128 TB of storage. This will allow the SWAMP shared facility to scale when the need arises and to potentially host dedicated instances of the SWAMP (shared facility and SiB) for educators and other potential users, such as researchers who have limited computing resources.

SWAMP’S CORE SERVICES

MANAGE ACCOUNTS, PROJECTS, AND ACCESS CONTROL

Secure access to the SWAMP is accomplished through the use of user accounts and an associated credential. The SWAMP supports federated identity management and account creation with CILogon, GitHub, and Google credentials. SWAMP-in-a-Box supports integration with LDAP and Active Directory. In the SWAMP, a project is used to group users working on a set of software packages, tools, or results. Project can be created by users. The creator of the project 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 coarse-grained with access permitted to all members of the project or all users of the SWAMP.
MANAGE SOFTWARE PACKAGES & 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. By default, the software is private to the user’s SWAMP account, but users can easily share multiple versions of software packages with other SWAMP users via SWAMP Projects.

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 CoA). 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. 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 dashboard also facilitates periodic and large volume CoA runs.
OUR ON-PREMISE SOLUTION: SWAMP-IN-A-BOX (SiB)

SiB is a downloadable (portable) version of the SWAMP software that can be customized and deployed locally by a SWAMP user. SiB addresses three main needs: (1) it allows for integration with existing enterprise services, e.g., identity management; (2) it allows access to licensed software tools; (3) it allows assessments to be run locally, allowing a user to maintain localization and control of sensitive software and weakness information. SiB has already had positive reception at organizations like Raytheon, Open Science Grid, and CERN.

SiB maintains key features from the main SWAMP facility, including delivering a complete set of assessment tools, installed, configured, and ready to run in a single software bundle. In addition, SiB users benefit from the tool automation, web-based interfaces, and integration with IDEs, software repositories, and continuous integration frameworks.

::: SiB TESTIMONIAL

“Both our developers and I in my role as Head of Security, have found the service extremely useful! The most efficient [way] to improve security of software is indeed to identify and fix vulnerabilities already in the development phase. We think that is excellent that you already support so many programming languages, tools, and platforms and that it is possible to install the service on site too in case your code is confidential.”

- Urpo Kaila, Head of Security at CSC - IT Center of Science in Finland (csc.fi); use of SWAMP-in-a-Box
ADOPTION OF SWAMP

Making software assurance tools easier to use is key to increasing the number of organizations that use such tools. We have addressed this issue in the following areas.

TOOL INTEROPERABILITY STANDARDS

We developed a unified tool output reporting format, called the SWAMP Common Assessment Results Format (SCARF). This format makes it much easier for a tool results viewer to display the output from a given tool. As a result, we have fostered interoperability among commercial and open source tools, such as Parasoft, GrammaTech, and Code Dx. Our SCARF framework includes open source libraries in a variety of languages to produce SCARF and process SCARF. In addition, we have produced open source result parsers that translate the output of all the SWAMP-based tools to SCARF. We continue to work towards tool interoperability standards by joining the Static Analysis Results Interchange Format (SARIF) Technical Committee. As a participating member, we contribute to creating a standardized, open source static analysis tool format to be adopted by all static analysis tool developers.

INTEGRATION WITH THE SOFTWARE DEVELOPMENT WORKFLOW

The goal is to allow the software developer to have push-button access to software assurance tools from their development environment. To this end, we produced plug-ins for integrated development environments (IDEs) such as Eclipse, continuous integration frameworks such as Jenkins, and code repositories such as git and SVN, including direct access and authentication with GitHub. As an extension to continuous integration, the software assurance capability can be leveraged in DevOps testing.
TOOL AUTOMATION FRAMEWORK

Our goal is to ease the task of a programmer to properly and effectively run a tool on a software package. This automation capability was previously available only from the largest commercial tool providers; the SWAMP project has made it available as open source to all tool developers, including researchers and small companies. This automation capability also allows integration into DevOps testing phases, allowing software assurance checks to be added into DevOps.

FEDERATED IDENTITY MANAGEMENT

To ease the access to the SWAMP, we allow users with existing identities from GitHub, Google, and InCommon (higher education institutions) to use the SWAMP directly without needing a SWAMP-specific password.

TRAINING

Education is key to increase awareness and reach out to our various constituent communities. To that end (in cooperation with the National Science Foundation’s Cybersecurity Center of Excellence), we have produced extensive training materials on software assurance tools in general and use of SWAMP in particular.
Shows the workflow of tasks common to the typical implementation and testing steps of software development methodologies such as the Waterfall methodology, development tool technology commonly used by developers to perform the task, and current and future support to easily integrate this technology with the SWAMP to perform builds, run static assessment tools, and view results.
In a time when million-dollar security breaches of household name corporations regularly make headlines, computer science undergraduates at America’s universities remain surprisingly underexposed to basic cybersecurity tactics.

SWAMP has been working to address this skills gap through a unique partnership with Bowie State University in Maryland. The SWAMP offers a rich and accessible suite of software security tools that Bowie State has been integrating into undergraduate coding courses, giving students an efficient way to examine and rid their code of security weaknesses. The partnership offers a national model for integrating cybersecurity into the curriculum. Other educational usage has included a SANS course (SEC 504) and courses at Indiana University by Dr. Steven Myers.

“I have used SWAMP for two years in my introductory graduate security courses aimed at the secure implementation of protocols and systems. Within the course we spend approximately 2-3 weeks discussing methods for secure programming, and within that the need to use static analysis tools. In both years I have required students to analyze code bases to find a diverse group of security flaws, understand the flaws and determine how to fix the code. They must use SWAMP to both help find and understand the errors---as SWAMP provides links to other taxonomies such as Mitre’s CWE, to provide context to the errors. I feel that this is an invaluable exercise for students both in learning to use static analysis tools, and to broaden their exposures to real world flaws. Further, SWAMP’s inclusion of curated repositories makes it easy for students to find real-world flaws in projects that are related to their interests. From an instructional point of view, the ability to seamlessly make the tool available to my students, have them access it on the cloud from anywhere, and not have to worry about installing a number of packages, which creates significant lab overhead, is invaluable.”

- Dr. Steven Myers, Indiana University; use of SWAMP in the classroom

NATIONAL SCIENCE FOUNDATION

The National Science Foundation, in its “Vision and Strategy for Software for Science, Engineering, and Education,” recognizes that software “must be reliable, robust, and secure” to “produce trustable and reproducible scientific results.” SWAMP is becoming an important component in that vision. NSF’s solicitation for “Software Infrastructure for Sustained Innovation” strongly encourages “the appropriate use of analysis tools and capabilities such as those made available through the Software Assurance Marketplace.”

“Because SWAMP integrates the analyses of multiple static analysis tools, it provides information that, properly interpreted, reduces both false positives and false negatives. The SWAMP program provides technical support on installation and tool problem resolution. Additionally, the program will provide training on safe coding using SWAMP as a teaching tool for the cost of travel of the instructors.”

- George Gardner, Federal Aviation Administration (FAA); use of SWAMP-in-a-Box

“As a free SCA [static code analysis] platform to mitigate risk in SW [software]. It fills gaps for when SW [software] developers don’t have access to paid SCA [static code analysis] tools and it gives you multiple tool access…. [I am] Looking to fill gaps from a SW [software] Assurance perspective and make code more secure. [I want to] Widen the use of SCA [static code analysis] within NASA.”

- Brandon Bailey, NASA IV&V use of SWAMP-in-a-Box

BLACK HILLS INFORMATION SECURITY

Finally, we have usage by information security professionals, including Black Hills Information Security.

“BHIS has used SWAMP and found the interface very well thought out. I know of no other way to easily and quickly run a number of source code analysis tools as to use SWAMP. When I have a security source code review to perform, SWAMP will be my first choice.”

- Ethan Robish, Black Hills Information Security (BHIS); use of SWAMP

ADVOCATING SOFTWARE ASSURANCE

A key role for the leadership of the SWAMP has been to promote the overall topic of software assurance in academia, industry, and government. Some of the ways that the SWAMP team has accomplished this task are:

- Livny and Miller testified in front of the U.S. Senate Homeland Security and Governmental Affairs Committee, providing them with a summary of technical and legal issues associated with software assurance.

- Miller testified in front of the Wisconsin State Assembly Committee on Jobs and the Economy on software assurance and the role that education and training play in developing a workforce skilled in software security.

- Miller, along with his colleague Dr. Elisa Heymann, have advised Members of the European Parliament on the topic of software assurance for open source software.

- Miller, along with his colleague Dr. Elisa Heymann, are active participants in the Safe and Secure C Study Group of ISO WG14 (C Standards), helping develop standards for software assurance tool capabilities.

- Basney presented about SWAMP at the Fall 2017 meeting of the Coalition for Academic Scientific Computation (CASC).

- Welch presented about the SWAMP to the Regenstrief Developer University, the 2014 International Conference on Software Engineering Research and Practice (SERP), OWASP Indianapolis, OWASP Bloomington, and the Circle City Con.

- Livny presented the SWAMP as a guest speaker in the GDIT Software Tech Summit in Falls Church, VA.

- Livny presented about the SWAMP to the Wisconsin State Employee Trust Fund IT Director Council meeting in Madison, WI.
OUTREACH, TESTIMONIALS, & ADVOCATION

ESTABLISHING CONTINUOUS ASSURANCE COLLABORATIONS

Besides the impact of the SWAMP technology, the SWAMP project has grown and matured the software assurance community. Examples include:

- **Collaboration with major commercial software vendors** (Code Dx, Denim Group/ThreadFix, Parasoft, GrammaTech, Red Lizard Software, Programming Research/PRQA, Sonatype, Synopsys/Coverity) to include their software in the SWAMP software framework.

- **Working with the Open Web Application Security Project (OWASP)** on software assessments, deploying OWASP tools (OWASP Dependency Check), and presentations at the OWASP AppSec USA and EU conferences and regional OWASP chapter meetings.

- **OASIS SARIF (Static Analysis Results Interchange Format) Technical Committee**, voting and participating members.

- **Providing software assurance tutorials, presentations, and a booth presence at the O’Reilly Open Source Conference (OSCON) and Security Conference.**

- **Booth presence at the NSA Information Assurance Symposium.**

- **Representing the SWAMP community at the Software and Supply Chain Assurance Forum and Software Security Assurance Exploratory Group Meetings.** SWAMP personnel have regularly attended these meetings and presented topics relating to the SWAMP on several occasions.

- **Collaborating with the NSF Cybersecurity Center of Excellence** to provide software developers training in secure coding, software assurance, and usage of the SWAMP.
OUTREACH, TESTIMONIALS, & ADVOCATION

- Collaborations with educators, educational institutions, and research institutes: Bowie State University (Lethia Jackson), Indiana University (Steve Myers), Northcentral Tech (Calvin Thorne), West High School in Madison, Information Technology Academy (ITA), Science Saturday at MIR, University of Texas Dallas, Rochester Institute of Technology (RIT), MIT, CERN, WISE Workshop – Wise Information Security for collaborating E-infrastructures, University of Colorado – Boulder (Dan Massey)

- Collaborations with tool creators: Kestrel (CodeHawk tool), HRL (TIF tool), NYU/NJIT (in-toto tool), Roberto Bagnara (ÉCLAIR tool), UC-Irvine (RevealDroid tool), Carnegie Mellon University-Software Engineering Institute (CMU SEI) (DrSpace tool), RAM Laboratories (RASAR tool), Mehdi Mirakhorli - Archie (private tool), BugCrowd, Checkmarx, NowSecure, Veracode

- Software integrations: HTCondor, CILogon (Google, Universities), GitHub, Git, Jenkins, Eclipse, Subversion/SVN

- Collaborations with conference attendees while exhibiting or presenting at the following trade shows: InfoSec World, MISTI, InfoSec Insider blog; ApacheCon; Circle City Con; LASCON; RSA Conference; STAR EAST Conference; DevOpsDays Madison, International ImageJ User and Developer Conference

• Collaborations with media outlets: Dark Reading, FedScoop/Cybersecurity Insights & Perspectives podcast, Silver Bullet podcast, Wisconsin State Journal, Viviry Tech INFOSEC Encyclopedia (Andy Freeborn blog/article series)


• FAA

• IEEE, Software Technology conference, SecDev 2017, “From Continuous Integration to Continuous Assurance” white paper by Bart Miller and Jim Kupsch was accepted by IEEE

• NIST Workshop on Software Measures and Metrics to Reduce Security Vulnerabilities

• CTSC Webinar Series

• Digital Infrastructures for Research meeting

• First International Conference on the Internet, Cyber-Security, and Info Systems
THE SwA ECOSYSTEM & THE SWAMP'S ROLE

The SWAMP is part of a larger government effort to increase the use of SwA tools and make these tools more effective.

STATIC TOOL ANALYSIS MODERNIZATION PROJECT (STAMP)

STAMP is “closing the gaps in two key areas: research and development, and implementation, of new techniques for static software analysis; and applying new and improved testing and evaluation activities capabilities. The goal of STAMP is to modernize a list of candidate software analysis tools to improve tool performance and coverage, to seamlessly integrate and support continuous integration and DevOps operational environments, and provide stronger analysis of results by reducing false-positives, and provide more visibility into false-negatives that often leave residual risks. STAMP should be designed to create new techniques that advance the state-of-the-art capabilities found in software analysis tools.”7

The goal of integration with the SWAMP is overall awareness and adoption of improved static analysis tools. Part of the modernization of these static analysis tools means the creation of a test case generator to measure the current gaps in static analysis tools. This test case generator will include examples of production software (“real programs”). As part of STAMP, newly modernized static analysis tools will be integrated into the SWAMP for use by the software development community. SWAMP will be one of the primary means for distribution and access to these newly improved static analysis tools, thereby allowing SWAMP to continue to reduce the barrier to adoption of continuous assurance practices. STAMP is targeted to deliver the newly modernized static analysis tools to the SWAMP in 2017 and 2019. Full integration of all deliverables from STAMP to the SWAMP is targeted in 2020.

THE FUTURE OF SWAMP SOFTWARE

The overall goal is for MIR-SWAMP to be augmented by the SWAMP-in-a-Box (SiB) distribution to create an ecosystem of customizable SWAMP instances to meet specific (local) demands. MIR-SWAMP will serve to support educational, training, research, trial usage, supply chain, and usage by individuals with limited infrastructure capabilities with individual SiB deployments supporting custom needs for higher assurance, privacy, or integration with local infrastructure and capabilities.

INNOVATIVE OPPORTUNITIES

ALLOW SHARING (EXCHANGE) BETWEEN SiBS AND THE MIR-SWAMP FACILITY

This sharing ability will allow users inside the same organization to run their own SWAMP instances and share their projects and assessment data at will and with complete control.
PROVIDE AN ACCELERATED PATH FOR SWAMP INTEGRATION WITH COMMERCIAL ASSESSMENT TOOLS

A few commercial tool vendors have been willing to deploy their tools in the open SWAMP facility on a limited basis, only accessible to open source projects and instructional (academic classroom) users. However, they are looking for a more attractive value proposition for such inclusion. SiB offers a way to deliver their tools to any user, controlling tool access with a bring-your-own-license (BYOL) model. We have already had several major tool vendors step forward to participate in this way.

SIMPLIFY THE SWAMP SECURITY STRUCTURE

With a locally installed SiB, many of the threats that originate from an open-access service are eliminated or reduced. As a result, the structure of SiB can be simplified, with a corresponding reduction in execution overhead, software complexity, and resource consumption.

We will leverage the developments and technical accomplishments of the past few years, with SWAMP-in-a-Box anchored by MIR-SWAMP as our vehicles going forward. In this section, we present a discussion on the role of the SWAMP service in the years going forward, specifically:

- The characteristics of SWAMP-in-a-Box (SiB) and MIR-SWAMP that make them ideal platforms in different contexts for delivering agile, innovative, and widely-accessible SwA resources, supporting tool developers, software projects, education, and software assurance research.
- A description of how SWAMP can effectively enable SwA education and training.
- A path for using SWAMP as the foundation for software certification resources and services.
MIR-SWAMP DEPLOYMENT

The SiB distribution will allow for on-premise, high-privacy/security and customized SWAMP instances but represents a non-trivial deployment effort. To allow for easier use and evaluation of SWAMP, we will evolve the existing MIR-SWAMP environment, as previously described, to support custom SWAMP instances for educational and training usage (e.g., Bowie State, Indiana University, SANS), for trial usage by potential users of SiB, and as a computation intensive resource for projects and groups with large assessed needs.

THE FUTURE OF SWAMP

EDUCATION

To date, the two biggest successes for the SWAMP service have been in the classroom and in independent exploration of software assurance and continuous assurance concepts.

While the feedback from both the instructors and students has been positive, the SWAMP needs new features for the instructor and student to streamline their tasks and better match the educational workflow, including:
• DEVELOPING HOMEWORK ASSIGNMENTS: A SWAMP instance can be preconfigured with all the necessary components to complete the assignment. The SWAMP can be provisioned with only the tools necessary for the assignment. It can also contain sample application package code or pre-configured directories ready for the students to develop their own code.

• DISTRIBUTING ASSIGNMENTS TO THE CLASS: Making the assignment available to the class is as simple as having it available for web download.

• TURNING IN ASSIGNMENTS: The SWAMP instance can be pre-configured with “turn in” scripts and web interfaces so that proper materials are made available to the instructors. These materials might be the application package they developed, the results from one or more assessment runs, or both.

• GRADING SUPPORT: We will provide scripts and tools for easily extracting assessment results from a SWAMP instance and collecting and summarizing them.

Exploratory use of the SWAMP needs to allow the user to access the SWAMP features with a minimum number of steps. As such, we will provide predefined SWAMP instances, pre-configured with simple application packages in a variety of programming languages, ready to run assessments. These exploratory users can run assessments on these packages or use them as prototypes to get their own packages running quickly.

“...the students gain an understanding of what is secure coding, but most importantly their confidence is what they really gain. They feel more confident in programming, period.”

- Dr. Lethia Jackson, Bowie State University; use of SWAMP in the classroom

ENABLING SwA TRAINING

The SWAMP has already been actively involved in supporting SwA training and produced training materials targeted at understanding and use of SwA tools and use of the SWAMP. The materials have been presented at multiple professional and academic venues as part of the activities of Miller and his colleague Dr. Elisa Heymann.
As a next step in this training program, they are developing hands-on exercises in the use of SwA tools and the SWAMP. As for the educational use described above, SWAMP offers the ideal delivery vehicle for the material associated with these exercises. For each subject area, a pre-configured SWAMP instance can be prepared with a complete programming environment, SwA tools, and example programs, allowing the students to minimize the time needed to install and configure their computers.

For training programs such as the one we are helping to establish for the FAA, we can customize the tool use, tool settings, and example programs to match their curriculum. We can help produce pre-configured virtual machines for other organizations that are establishing similar programs.

**SWAMP AS A CERTIFICATION FRAMEWORK**

There is growing interest in providing certification of software security properties. The idea of a software certification or standard provides a tangible motive and reward to a company for adopting a software assurance effort, something that is missing today.

Recent efforts to provide such certification include the emerging Underwriters Lab UL-2900 standard, Peiter Zatko’s (aka, Mudge) Cyber Independent Testing Labs (CITL) scoring system, and software standards from the I Am the Cavalry group for the automotive industry and connected medical devices. In addition, there are increased discussions in legislative bodies and industry trade groups about developing software assurance standards.

A standard such as UL-2900 may be out of reach for all but the most high-end software applications. As a result, they are less likely to receive wide adoption. UL-2900 includes comprehensive design and documentation standards, with rather specific analysis of specific risks.

This is certainly a challenge for any software vendor to produce all this material. Software also needs to be fuzz tested and scanned for known vulnerabilities (with a tool such as OWASP Dependency Check or Sonotype Application Health Check) and viruses.
The last section of UL-2900 requires that static analysis tools be used to look for problems such as the SANS Institute top 25 weaknesses. This part of the standard is the one most easily adopted by software projects and one that could be supported directly by the SWAMP.

The SWAMP can deliver a pre-configured collection of assessment tools and (to be developed) software to process the tool output. The software developer would load their application package into this certification SWAMP instance and run the required collection of tools.

There are many issues to address on the way to providing this facility, including:

- With which organizations we should partner to establish such a facility. As an important starting point, we are beginning discussions with Underwriters Laboratories (UL), with respect to the proposed UL-2900 rating.
- Based on our discussions with certification partners, we will develop a plan that describes:
  - The tools for each language that will be included in the testing.
  - The checks that should be enabled for each tool.
  - Criteria for translating tool output to ratings criteria.

**SWAMP AS INTEGRAL TO THE SECURE SOFTWARE SUPPLY CHAIN**

Software integrators and users are often faced with insecure software supply chains. By analyzing software, risks resulting from software from unclear or insecure supply chains can be managed as part of a Cyber Supply Chain Risk Management (C-SCRM) process. Augmenting provence information, or replacing it in the case of unclear provenance, the SWAMP offers the ability for a software user or integrator to evaluate software before use or deployment to judge its risk to their organization. Use risk evaluation can be either built on the aforementioned certification efforts described in the previous section or be customized by the evaluator to their own risk tolerances or environmental concerns.
The SWAMP vision provides an ecosystem of Software Assurance tools guided by the fundamental principles of Continuous Assurance—an essential component of the software development lifecycle—provided by the SiB software distribution and anchored by the MIR-SWAMP facility.

Through this document, we have discussed the SWAMP solution, continuous assurance software, and our impact through outreach and testimonials. We have also discussed the future vision of the SWAMP and its guiding principles through collaboration with various organizations, educational institutes, and expansion of SWAMP-in-a-Box. The SWAMP software will continue to enable continuous assurance as a natural extension of continuous integration, including the DevOps model.
MORGRIDGE INSTITUTE for RESEARCH

330 N. Orchard St.
Madison, WI 53715
swamp@continuousassurance.org

SUPPORT
support@continuousassurance.org
1-608-316-4266

WEBSITE
continuousassurance.org
mir-swamp.org

SOCIAL
GitHub: github.com/mirswamp/deployment
Twitter: twitter.com/SWAMPTEAM
Facebook: facebook.com/SoftwareAssuranceMarketplace
LinkedIn: linkedin.com/groups/4715681