
AFRL-RI-RS-TR-2020-214

# CONTINUOUS SOFTWARE ASSURANCE THROUGH A NATIONAL MARKETPLACE

MORGRIDGE INSTITUTE FOR RESEARCH

*NOVEMBER 2020*

FINAL TECHNICAL REPORT

***APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED***

STINFO COPY

**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

# NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2020-214 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

**/ S /**
STEVEN DRAGER
Work Unit Manager

**/ S /**
GREGORY HADYNSKI
Assistant Tech Advisor, Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

# REPORT DOCUMENTATION PAGE

*Form Approved*
**OMB No. 0704-0188**



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| NOVEMBER 2020 | FINAL TECHNICAL REPORT | SEP 2012 – MAR 2020 |

**4. TITLE AND SUBTITLE**

CONTINUOUS SOFTWARE ASSURANCE THROUGH A NATIONAL MARKETPLACE

**5a. CONTRACT NUMBER**
FA8750-12-2-0289

**5b. GRANT NUMBER**
N/A

**5c. PROGRAM ELEMENT NUMBER**
N/A

**6. AUTHOR(S)**

Livny, Miron; Miller, Bart; Basney, Jim; Welch, Von; Landrum, Irene; Kupsch, James A.; Burger, Josef T.; Peterson, Jeffery; Megahed, Abe

**5d. PROJECT NUMBER**
DHS2

**5e. TASK NUMBER**
SW

**5f. WORK UNIT NUMBER**
MP

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Morgridge Institute for Research
330 North Orchard Street
Madison WI 53715

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Air Force Research Laboratory/RITA
525 Brooks Road
Rome NY 13441-4505

DHS / CSD
1100 Vermont Ave
Washington DC 20005

**10. SPONSOR/MONITOR'S ACRONYM(S)**
AFRL/RI

**11. SPONSOR/MONITOR'S REPORT NUMBER**
AFRL-RI-RS-TR-2020-214

**12. DISTRIBUTION AVAILABILITY STATEMENT**
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

The SWAMP project was built and executed on the foundation of a commitment to the goal of promoting effectiveness and adoption of software assurance. The project pioneered the concept of "continuous software assurance" and followed a multipronged approach to create an open source, portable continuous assurance platform that addressed the needs of an evolving ecosystem of software assurance practices. The SWAMP strategy targeted software developers, tool developers, educators and researchers. With these users in mind, the project created an open platform that demonstrated the power of continuous software assurance. The SWAMP public facility and SWAMP-in-the-Box software provided a working blueprint for the architecture and functionality of a continuous assurance capability with the ability to be fully integrated into the software development life cycle. In addition, the project identified gaps in available technology that guided and advanced future R&D in software assurance. By operating a public marketplace, the SWAMP project brought the power of hands-on, continuous software assurance to individual developers, small development groups, class rooms and training sessions that would not have otherwise been able to access such resources without being in large organizations with a well establish software assurance program. The SWAMP project lowered the threshold for organizations to continuously harness software assurance tools.

**15. SUBJECT TERMS**
Continuous assurance; continuous software assurance; software assurance; static analysis; standardized static analysis results

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | **STEVEN DRAGER** |
| U | U | U | UU | 60 | 19b. TELEPHONE NUMBER (*Include area code*) **N/A** |

## TABLE OF CONTENTS

## LIST OF FIGURES

## 1.0 SUMMARY

The Software Assurance Marketplace (SWAMP) project has been committed, since its inception over seven years ago, to promoting software assurance by pioneering the concept of "*continuous* software assurance" and by developing an open source, portable continuous assurance platform. Continuous software assurance is the practice of integrating security assessments into the entire software development lifecycle (SDLC), a natural extension of continuous integration (CI). The practices of continuous assurance are in many cases not followed due to time-to-market constraints and the usability of software assurance tools. **The SWAMP project lowered the threshold for continuously harnessing software assurance tools.**

In order to support continuous assurance throughout the SDLC, the SWAMP platform integrated a set of software assurance tools with automation, web and integrated development environment (IDE) based interfaces, parsing of results in a standard format (a precursor to the Static Analysis Results Interchange Format (SARIF)), cloud integration (Amazon Web Services (AWS)), and federated identity management and software repositories. The platform was offered in two forms: the SWAMP-in-a-Box (SiB) downloadable software distribution, for local deployment and customization; and through the free-to-use open SWAMP (mir-swamp.org) facility. SiB (1) allows for integration with private enterprise services (e.g., identity management, software repositories, processing capacity, cloud) and (2) it enables organizations to maintain control of sensitive software and weakness information.

The open SWAMP facility offered an anchor for the SiB deployment. Hosted in a state-of-the-art research data center and sharing a code base with the SiB, the open facility met the security, privacy, and confidentiality needs of the software assurance (SWA) community. As the interest of the community in the SiB option grew, the open facility continued to serve as a means to bring DHS-funded assurance tools, such as RevealDroid, the Code Dx results viewer, and 11,574 Bug-Injector test packages, to the wider community.

SiB and the open SWAMP facility constituted an open ecosystem of continuous assurance capabilities that was customizable to meet the evolving needs of a range of users, from education and training to on-premise integration in high-security, software development environments.

## 2.0 INTRODUCTION

### 2.1 The Problem

Every year companies lose billions of dollars due to security vulnerabilities in software built within their walls or from commercial-off-the-shelf (COTS) software purchased for use within their company. Most software built within companies and COTS utilize open source software components. Open source software projects tend to have limited staff and money. There is no guarantee the developers have analyzed their code for weaknesses, and, in fact, often those developers do not have the means to analyze their code due to significant barriers. These barriers to static code analysis (SCA), or continuous software assurance include: time-to-market constraints, lack of knowledge or training on static analysis, results interpretation, funding constraints, physical resource constraints, and the quality of the currently available static analysis tools. The purpose of the SWAMP project was to address as many of these barriers as possible in order to simplify the process of adopting continuous software assurance practices into the software development lifecycle. In order to lower the threshold for software assurance, a novel platform dedicated to continuous software assurance was needed for easy adoption and integration into existing workflows.

In 2012, the SWAMP project pioneered the concept of *Continuous Assurance*, which extends continuous integration into the software assurance space. Our vision was to develop an open platform that naturally integrates the security assessment of software into the software development workflow. Continuous assurance incorporates and automates software assurance tools into the process of building and testing software throughout its lifecycle. Software developers would have an automated platform that could easily integrate into their software development workflow. Static analysis tool developers would have a series of test suites to develop and test their static analysis tools against, thereby improving the quality of static analysis tools. Infrastructure managers would have an automated platform to analyze locally created and third party software before implementing. Educators and students would have a resource for teaching and learning about static analysis with the hope that students would apply continuous assurance practices to work in future jobs.

Another goal of the SWAMP project was to improve software assurance outcomes by combining the strengths of multiple analysis tools. The resources needed to integrate one tool, let alone multiple, into the SDLC and successfully interpret disparate analysis results was seen as a barrier to adoption of Continuous Assurance practices. The SWAMP project set out to provide an easy means to continuously assess software using multiple analysis tools while providing results from those tools in a common format.

Finding skilled developers who can install, configure, run, and interpret results from one static analysis tool is hard to do. The other issue with finding a skilled developer to run one static analysis tool is that one tool does not provide good code coverage; not all of the source code will be scanned by one tool. Using multiple quality tools to adequately analyze source code is a requirement. Finding a developer who can install, configure,

run, and interpret results from multiple tools is extremely rare. Most developers are capable of running a static analysis tool, but do not have the knowledge or training to interpret results and each static analysis tool provides a unique output. Another goal of the SWAMP project was to address the disparate analysis results, reducing the threshold for interpreting results from different tool outputs.

As a national effort to secure the Nation's critical infrastructure and provide a resource to secure open source software, the SWAMP project aimed to maximize the adoption of the platform's capabilities to meet the evolving needs and technologies users face in today's software ecosystem.

# 3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

## 3.1 Cybersecurity

The Cybersecurity team was responsible for developing and executing the security program for SWAMP (i) to maintain confidentiality of user data in the SWAMP and (ii) to protect the SWAMP software itself, from the external entities. The security program included ongoing, repeated, and situational activities such as security leadership, risk assessment, control design and implementation, auditing, monitoring, incident coordination, communication, and training and awareness.

The SWAMP Cybersecurity team followed a risk-based approach to information security, based in part on readily available best practices and standards such as those found in the NIST Special Publication series [1]. The top priorities of the Cybersecurity team were (i) maintaining confidentiality of nonpublic user data (for example, weaknesses, software, and tools) in the SWAMP and hence maintaining the confidence of the user community and (ii) preventing misuse and protecting the SWAMP software itself, particularly from the external entities. The cybersecurity program included ongoing, repeated, and situational activities listed by sections below.

The Cybersecurity team carried out specific activities (defined in section topics) during and throughout the life of the SWAMP project and provided the SWA tool researchers and developers an assurance of privacy and security of both their tools and results, which may be sensitive due to commercial or other competitive considerations and/or sensitive from a cybersecurity point of view.

### 3.1.1 Security Leadership

Security leadership included cybersecurity program management and providing leadership on security-related decision making. As part of cybersecurity leadership, the cybersecurity plan for the SWAMP project was developed, evaluated and updated throughout the SWAMP project's time frame. The SWAMP Chief Information Security Officer (CISO) or the Cybersecurity team staff led the weekly/monthly calls between cybersecurity, infrastructure and software development personnel to discuss operational issues and review designs for new (and existing, when needed) SWAMP features.

Security leadership enforced "Separation of Duties" and "Need to Know" principles. For example, only members of the SWAMP Infrastructure team had privileged access to the SWAMP infrastructure needed to access the data. The use of a strict revision control system (GitHub and GitLab) was enforced to prevent malicious or accidental changes to the SWAMP software.

### 3.1.2 Threat Awareness and Risk Assessment

Threat awareness included monitoring various vulnerability reports and security announcements and checking against the SWAMP inventory of installed software packages to see if the SWAMP infrastructure or SWAMP users were affected.

The cybersecurity team maintained knowledge of state of practice and relevant threats and was responsible for reviewing vendor specific and third party vulnerability reports on a daily basis. The cybersecurity team was responsible for responding to data and/or service breach and vulnerabilities. Vulnerability advisories were sent to the SWAMP user community when they were impacted and the cybersecurity team ensured that the SWAMP software itself was regularly scanned, patched and kept up to date.

Risk assessments included both comprehensive and situational risk assessment while triaging vulnerabilities, handling incidents and triaging JIRA tickets with potential security concerns.

### 3.1.3 Control Design and Implementation

The cybersecurity team was responsible for developing policies, and for selecting or designing and implementing technical controls. The cybersecurity team defined and documented the policies for cybersecurity such as:

- Incident Response Policies and Procedures
- Password Policy
- Privacy Policy
- User Data Retention Policy
- Policy for accessing development, integration, and production environments.
- Two Factor Authentication Policy
- Website Security Policy

The SWAMP Cybersecurity team also developed:

- Data Classification Plan
- Intrusion Detection Strategies
- Vulnerability Management Program
- Interactive Virtual Machine (VM) Security Plan
- Data Breach Handling Process

### 3.1.4 Auditing and Monitoring

Auditing and monitoring included operational and network security monitoring. Security controls were audited, and the SWAMP infrastructure, including Internet facing (external) and internal hosts, were scanned regularly for vulnerabilities. The administration and implementation of the SWAMP software was constantly and actively monitored to check for issues that could lead to vulnerabilities in the infrastructure. Monitoring and scanning of SWAMP infrastructure and software included:

- Monitoring the Intrusion Detection and Prevention System (IDS/IPS)
- Responding to alerts from the FortiGate firewall
- Monitoring, triaging, and responding to the vulnerability scan results from OpenVAS and Qualys
- Reviewing, prioritizing and working on JIRA issues related to cybersecurity.

- Monitoring and responding to 2-factor Authentication (2FA) logs for anomalies.

### 3.1.5 Incident Coordination

The cybersecurity team led the security incident response processes. This included coordination during the different phases (for example, during Preparation and Response phases) of the incident handling process.

In order to ensure that the SWAMP teams would be ready to handle any eventual security incident, the SWAMP CISO defined and initiated simulated incidents to exercise incident detection and incident response. Actions were performed in the SWAMP integration environment and the cybersecurity staff, with limited knowledge of those actions, ensured SWAMP logging and forensics capabilities were sufficient to determine actions taken.

### 3.1.6 Communication, Training, and Awareness

The cybersecurity team maintained regular communications with the leadership and key personnel on security matters and held regular cybersecurity meetings.

The cybersecurity team led recurring meetings and responded to cybersecurity related questions posed via emails, the support ticketing system, and Signal™ (secure messaging app) from the rest of the SWAMP personnel. The cybersecurity team was responsible for updating the cybersecurity section of the Readiness Reports for Department of Homeland Security (DHS) on a weekly and then bi-weekly basis.

Various security trainings were provided to SWAMP personnel to make them aware of information security best practices. In addition, training on the SWAMP project's cybersecurity policies and basic cyber hygiene were provided to SWAMP personnel.

## 3.2 Identity and Access Management

The goal of the SWAMP Identity and Access Management (IAM) system was to provide secure yet usable authentication and authorization for SWAMP users.

### 3.2.1 Methods:

The team located at National Center for Supercomputing Applications (NCSA) designed the SWAMP IAM system, with support for account creation, password reset, linked external identities (InCommon, GitHub, Google), and collaborative project groups. When SWAMP users create project groups, they control the group membership and the sharing of assessment results within the group. SWAMP supports invitation-based group enrollment and add/remove of group members by group owners.

### 3.2.2 Procedures:

To support SWAMP educational use, the SWAMP IAM system connects to NCSA's CILogon (www.cilogon.org), which supports authentication with faculty, staff, and student credentials from US universities via the InCommon federation

(www.incommon.org). SWAMP users can access SWAMP services using their university credentials rather than creating a SWAMP-specific password for access.

NCSA developed a Java application programming interface (API) to SWAMP services to support SWAMP plug-ins for integrated development environments and to support SWAMP IAM regression testing. The API enables SWAMP use outside the web browser, directly from the developer's workstation. To further support integration with the local developer environment, NCSA also added the ability to integrate local SWAMP-in-a-Box installations with the local organization's IAM system (e.g., Microsoft Active Directory).

## 3.3 User Interface

The user interface for the Software Assurance Marketplace is the public face of the application and is intended to make the capabilities of the SWAMP available over the web and as easy as possible to use. The front page is shown in Figure 1.


**Figure 1. SWAMP Front Page**

### 3.3.1 User Interface Requirements

The most basic and essential requirements for the user interface (UI) capabilities are as follows:

#### 3.3.1.1 Allow users to manage their identities, profiles, and user permissions

The first requirement is the ability for users to create accounts, which is standard for any web application that allows users to upload and save or retain data.

#### 3.3.1.2 Allow users to upload their software packages for assessment

In order to be able to assess software source code, the source code must be uploaded to the platform. Since the source code typically involves several hundred or thousand

files, we required that the user first combined their source code into a single archive file, which would then be uploaded to the web application.

#### 3.3.1.3 Allow users to create and run assessments

Once the code has been uploaded, a user must be able to create and run assessments on that code. Running an assessment involves selecting a tool to use with that source code and also a platform operating system(OS) to run the assessment on.

#### 3.3.1.4 Allow users to view results from software assessments

Once an assessment has been completed, the application would also allow the user to view the results. It might have also been possible to omit this requirement from the web interface, in which case a user would be required to download the results and view them in a separate application. However, since we wanted the application to have broad appeal, it made sense to have a basic viewer integrated into the web application to make result viewing as easy and convenient as possible.

### 3.3.2 User Interface Goals

The web interface home screen is shown in Figure 2. In addition to the basic requirements above, we had a list of goals that we kept in mind and aspired to during the development process:

#### 3.3.2.1 Be agnostic with regards to tools

One of the aspects about the SWAMP platform that was unique was that it is intended to be an open marketplace for any user, language, or tool. Most static analysis web applications are offered by tool providers and are geared very specifically to the tool that is being sold by that particular vendor. Our goal with the SWAMP project was to never favor one particular tool and to treat all tools as equally as possible.

#### 3.3.2.2 Appeal to both software developers and static analysis tool creators

The target audience for most static analysis tools is software developers because that is by far the largest audience of users. However, there is also a small but important community of static analysis tool developers and researchers who would benefit from the SWAMP platform. The SWAMP project aimed to be useful to this community, in order to assist them with the task of improving the quality of static analysis tools.

#### 3.3.2.3 Make it easier to create packages and specify build information

One of the most difficult aspects of static analysis is that in order for most tools to run, they require the software to go through a complete build process. Building software is inherently a difficult and complicated process. We discovered early on in the SWAMP development that errors in specifying build information was one of the most common sources of problems. The solution that we arrived upon is a sort of “wizard” that inspects the software code and assists the user to make the right decisions in specifying the build information. While it doesn’t always provide all of the answers, in a vast majority of

cases, it provides sensible defaults and build parameters and greatly increases the success rate of assessments.


**Figure 2. SWAMP Web Interface Home Screen**

### 3.3.3 User Interface Main Workflow

The process of assessing software in the SWAMP platform involves the following workflow:

#### 3.3.3.1 Upload Your Source Code

First, you must upload your source code to the application to be analyzed, as shown in Figure 3. Since source code can be many hundreds or thousands of files, this is done as a compressed archive file. The SWAMP platform also allows users to choose between a number of options for where to upload the code from. They can upload the code from the storage on their local machine or they can also use a remote file server or Git repository.

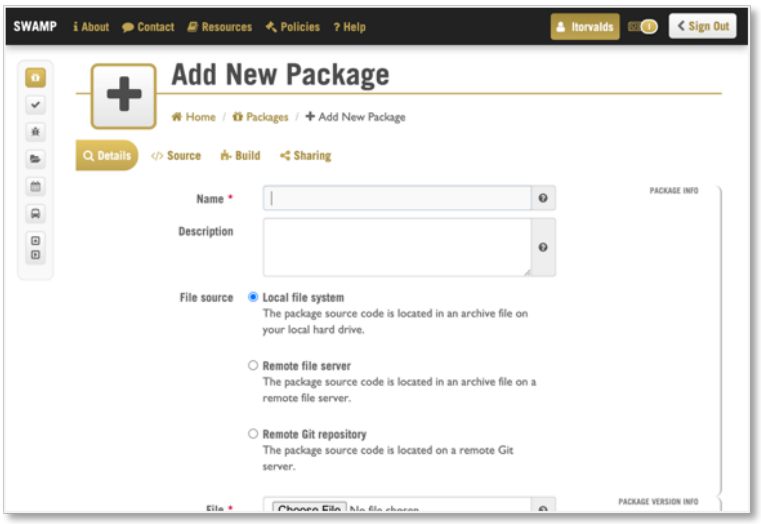


**Figure 3. Adding (Uploading) a New Package**

### 3.3.3.2 Set Your Build Parameters

After uploading your source code, you next need to set your build parameters, as shown in Figure 4. This is necessary because most software needs some sort of build process in order to run and the static analysis tools rely upon the build process in order to most effectively locate all potential weaknesses.

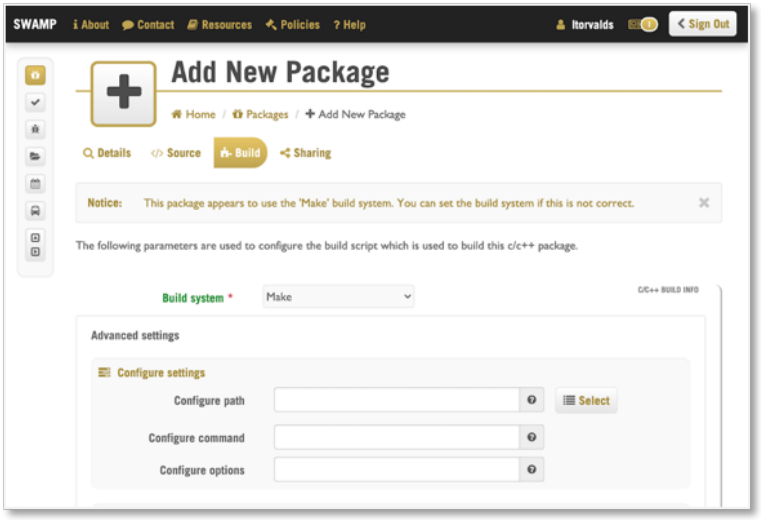


**Figure 4. Setting Build Parameters**

### 3.3.3.3 Select a Tools and Platform

Next, to run an assessment, you need to select the tools and a platform to use, as shown in Figure 5. The SWAMP project's philosophy is to use as many quality tools as possible, instead of relying upon a single tool, and then to compare results among the

tools. The areas where the tools agree are most likely to be valid weaknesses. This approach reduces the number of false positives, which is one of the major problems with static code analysis and one of the main reasons why it has not been more widely adopted. The SWAMP user interface makes it quick and easy to select all available tools for a particular package type.

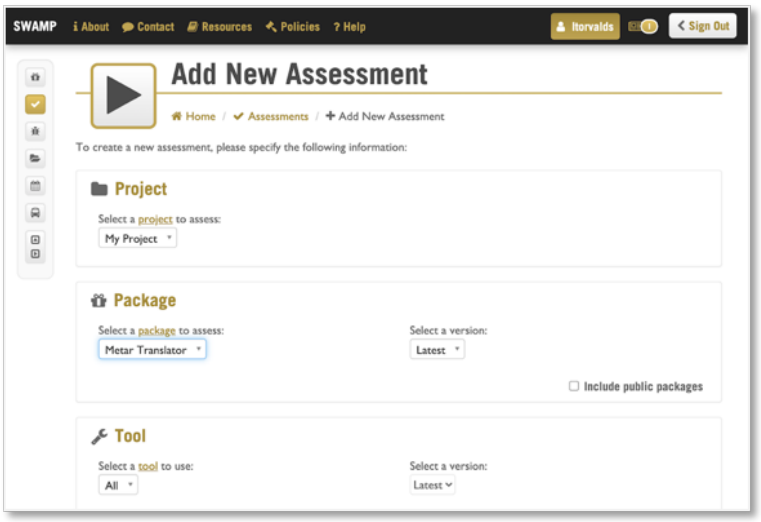


**Figure 5. Select Tool and Platform**

### 3.3.3.4 Run Assessments

Next, we run the assessments, as shown in Figure 6. Using the SWAMP platform, it is easy to run multiple assessments in one easy step. Since assessments can potentially take a while to complete, there is an option to send you a notification email when your assessments have finished.

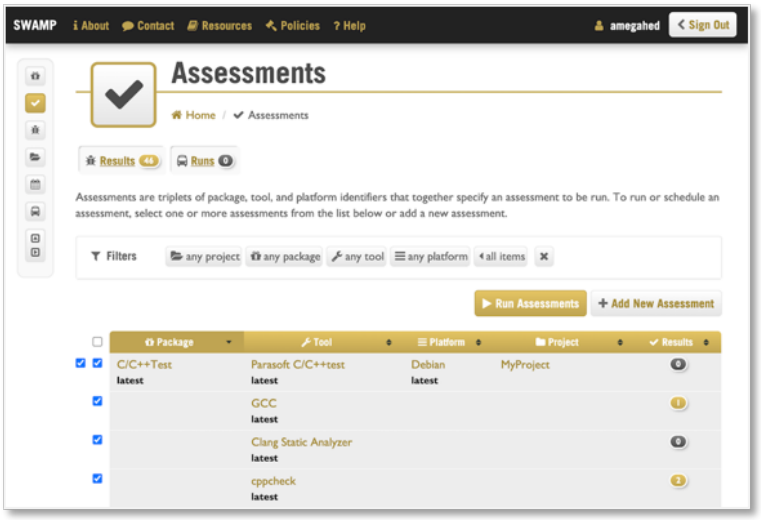


**Figure 6. Running Assessments**

### 3.3.3.5 View Results

Lastly, we view the results. The SWAMP platform offers two methods for viewing results: (1) a lightweight “native” viewer that’s built into the SWAMP platform, shown in Figure 7, and (2) a full featured third party viewer called Code Dx™, which is provided by a commercial partner, shown in Figure 8.

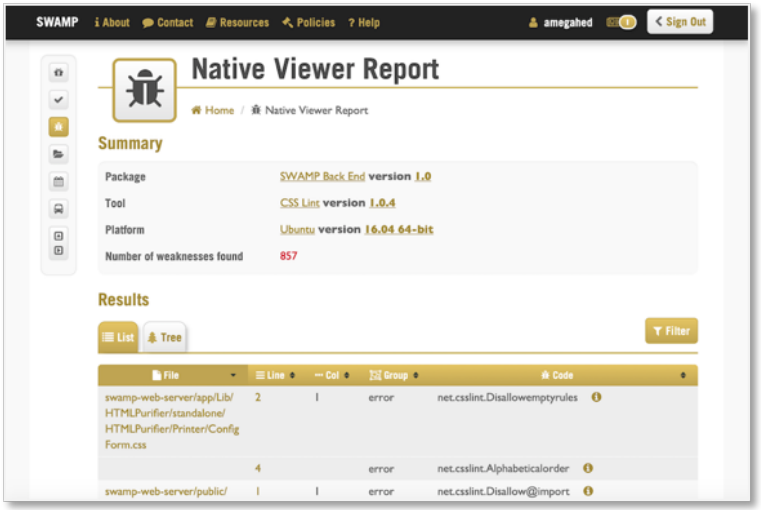


**Figure 7. The SWAMP Native Viewer**

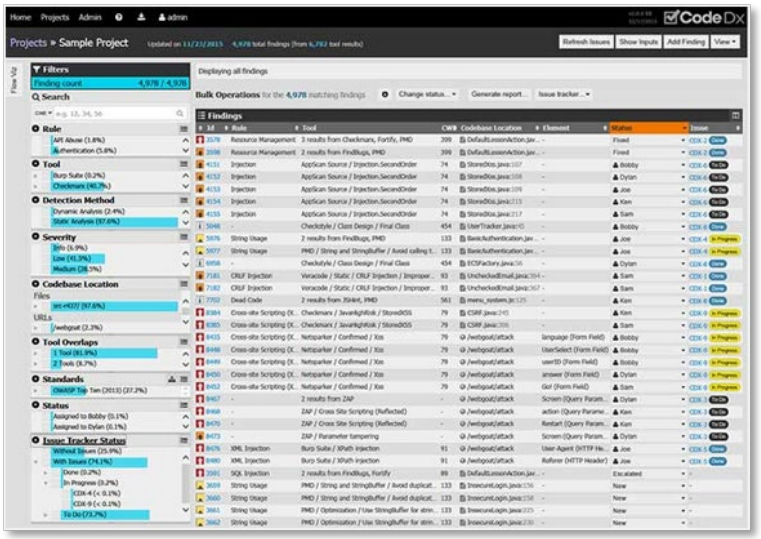


**Figure 8. The Code Dx™ Viewer**

### 3.3.4 User Interface Innovations

Below are some of the innovative aspects of the SWAMP project's user interface:

#### 3.3.4.1 Package Build Settings Wizard

The most significant usability innovation involved helping users provide build settings. Early on we found that a significant number of assessments were failing due to improper build settings. Building software is difficult. Even an experienced developer can have a difficult time figuring out the proper build settings. In the case of open source code, often the person who is assessing the code is not the original developer. In that case, it can be nearly impossible for the user to figure out what the proper build settings should be. At one point we noticed that most assessments were failing due to build errors. Something needed to be done. We looked at several different build systems that we supported and found that in most cases, the build files and structure tend to be relatively standardized. We determined that most of the build parameters can be discovered by automatically examining the code and applying a few simple rules.

**The Automated Package Build Detection Process:**

**a. Determine the source directory**

We next examine the directory structure to find the top level of the source directory. This is the top-level directory that contains source code files.

**b. Determine the package type**

By performing an inventory of the source code files contained in the archive, we look at the counts of the various file types and use these counts to make a guess at what the intended package type is, as shown in Figure 9. For example, if a package archive contains mostly ".c", ".cpp", and ".h" files, then it is most likely a C package.

**c. Determine the language version**

Some package types require that we specify a language version for the build, as shown in Figure 9. In the case of Ruby packages, for example, the language version may be specified inside of a file called "Gemfile". For this package type, we look for the Gemfile and parse the contents to discover the language version.

**d. Determine the build system**

Build systems tend to use specifically named files to specify the build, as shown in Figure 10. For example, the "Make" build system looks for files named "Makefile" or "makefile", the "Maven" build system looks for a file named "pom.xml", the "Ant" build system looks for a file named "build.xml", and so on. We examine the contents of the package to look for the build file or files that are appropriate for that particular build system. If multiple build files are found, the one at the higher level in the directory structure takes precedence.

### e. Determine configure settings

Like the build file, the configuration files for certain build systems tend to have standardized names and require certain standard strings to be entered into the config command to work properly, shown in Figure 10. For example, C packages may contain a file called “configure.ac” which indicates that they are to use a build system called “autotools+configure+make”. In that case, the configure command is automatically set to the string

mkdir -p m4 && autoreconf --install --force || ./autogen.sh && ./configure”

This is an example of a non-trivial setting that most users would not be able to determine without assistance.

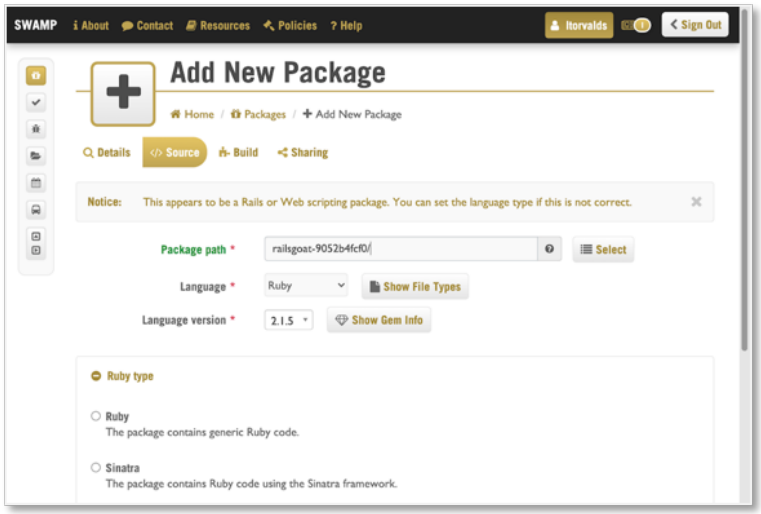


**Figure 9. Automatic Package Type and Language Version Detection**

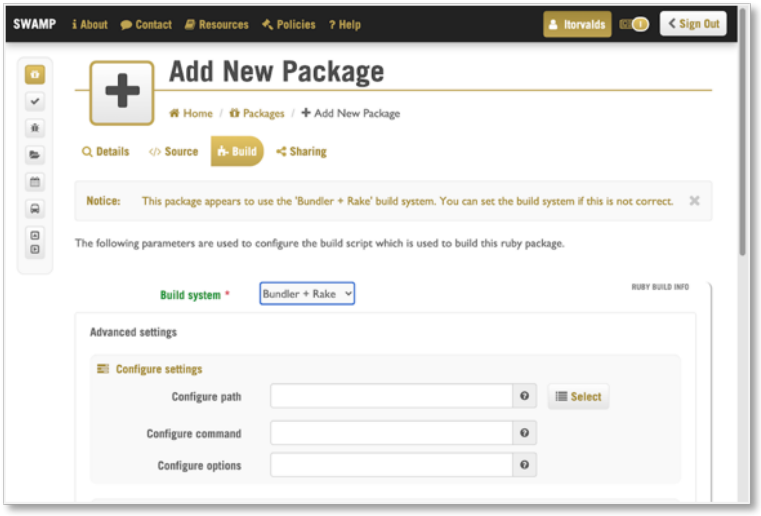


**Figure 10. Package Build and Configure System Detection**

### 3.4 Infrastructure

From the start the SWAMP infrastructure was focused on security and separation, with an eye on scalability. This started with installing dedicated keycard accessed racks in the already secured datacenter, and purchasing hardware to be as flexible as possible to deploy as the project took shape and grew. This was accomplished with Cisco unified computing system (UCS) server hardware and EMC virtual network exchange (VNX) Storage hardware. Security zones were established to limit cross-environment communication in the event of an intrusion, with virtual local area network (VLANs), restrictive host based, and network-based firewall zone configurations. The UCS hardware provided the flexibility and capability to provision and adapt the host machines with desired network and storage configurations to meet the security zone needs and evolutions.

To further increase deployment flexibility and assist in defining the security zones, several virtual machine managers were used; VMware and Red Hat Virtualization Manager (RHVM) (now Ovirt). Each development, testing, and production environment has their own set of RHVM Hypervisors running virtual machines to provide the services needed for each. This allowed us to have redundancy and task separation where we could try new things and upgrades in the development environments with an easy way to roll back changes and migrate between hosts to maintain uptime for the team. With this setup and using the EMC VNX storage system it allowed us to allocate storage resources where and when they were needed as the project grew; more disk space could be allocated to the Hypervisor pool and then provisioned to the VMs that needed it.

With an additional set of VMWare hypervisors to provide standard services across environments such as internal domain name system (DNS), lightweight directory access protocol (LDAP), and network time protocol (NTP), which were also setup as redundant. These hypervisors also hosted services like JIRA, Confluence, Git, and Jenkins, for use by the team. These services were utilized to help maintain the insular configuration of the SWAMP so there was also very little needed from our upstream host of DiscoverIT beyond a network connection.

Within each working environment the services used were further separated into separate virtual machines to keep the various components of the information apart and easier to secure and monitor. Web server, data storage server, identity management server, database server, were all individual installs. As the project went on, and some of the security considerations were adjusted, several of these functions were combined into fewer machines to make management and upkeep easier in the production environment.

To separate the assessments from the rest of the system, they were run as HTCondor Virtual Machine Jobs and were placed into their own internet protocol (IP) space, preventing access to the internals of the SWAMP infrastructure while they were running. The execution hosts used two network interfaces to fully separate the Virtual Machines from the OS. Software Defined Networking was also used to keep the assessments from interfering with each other. When the Defense Advanced Research Projects

Agency (DARPA) Cyber Grand Challenge hardware was added, most of these nodes were added to the production execution pool and increased the number of assessments and viewers that could be run simultaneously.

## 3.5 Assessments

An assessment is the fundamental task and unit of work of the SWAMP: analyzing the source code of a software package using a single Static Analysis tool to produce data about the software package in a usable output format including source code,weaknesses and metrics. A reported weakness is something that a tool determines is a problem in the software package that should be remediated. The weakness types discovered include the following types security, correctness, maintainability, performance, bad practices, and coding standards. A metric is a fact about the software package such as its language, number of lines of code or number of source files.

### 3.5.1 Goals and Motivations

The driving goals and motivations for design and structure of assessments in the SWAMP platform was born from the mission: support the assessment of software packages using multiple static analysis tools with support for multiple operating system platforms, support viewing of the assessment results in a single viewer, and provide a set of curated packages. The mission goals along with making a system that is easy to use and secure drove the following goals for assessments:

#### 3.5.1.1 Ease of Use

An overall goal of the SWAMP platform is ease of performing static analysis testing using multiple tools on multiple platforms. Deploying static analysis tools for an individual or organization generally comes at a significant cost in effort: the tool must be installed and configured; often software build configurations must be changed to meet the needs of the tool; users have to learn how to operate the tool and fit it into their development workflow. This must be done for each OS to be tested on, and all of it must be maintained as software packages, tools, and OS's are updated. Since each tool and its output are unique this process has to be repeated for each tool. This is a significant effort for each combination of tool and software package. Our goal has been to alleviate this cumbersome process to a simple set of settings per software package.

The following principles were followed to support ease of use:

1. No need to modify source code or build configuration. Users should not have to modify their software package in any way. The only requirement is that the software build and they be able to provide package settings on how to configure and build it.
2. No tool knowledge. Users should not have to know about installation and configuration of the tool, or how to decode each tool's output. Users only need to choose the tools they want to use.

3. No OS installation. Users should not have to manage the OS platforms in the SWAMP. Users only need to choose the platforms they want to use.
4. Only one output format to process.

#### 3.5.1.2 Isolation and Repeatability

Isolation of the assessment process is important for security of the facility itself. Performing an assessment requires building the user's software and that means the SWAMP platform needs to execute arbitrary code provided by the user, regardless of whether the user built the software or legally obtained the software to be assessed. Since this code submitted by the user cannot be trusted, it must be run in a way that isolates it from the platform's operational code to prevent damage to integrity of the SWAMP itself. It must also be isolated from the other code and data of other users to prevent the exposure of private data.

Reproducible results for assessing a software package with the same tool and operating system is a desired property. This is achievable if the initial state of the operating system is the same and the tool and build of the software package are deterministic. Most, but not all tools, are deterministic. Likewise, not all software package builds are deterministic as they may download libraries and files from the internet that may change over time.

Isolation and reproducibility resulted in the following principles:

1. Assessments must run in an isolated environment.
2. Each assessment must be started in an environment with the same initial state.

#### 3.5.1.3 Scalability

The SWAMP platform will often need to perform many assessment runs in a short period of time. This is due to there being one assessment per combination of software package, OS platform and tool. This is further increased with a concentration of simultaneous users such as would occur when an educational course is using the SWAMP open facility or during peak business hours. To support this infrastructure and perform isolated assessment runs, the platform must be capable of queueing assessment runs, running them concurrently, and allow the concurrency to meet the desired throughput.

Supporting scalability resulted in the following principle:

1. The isolation mechanism must be capable of being run under a scheduling system that supports the capacity of a single host or multiple hosts.

By coupling SWAMP software with the widely deployed HTCondor system, an open-source workload scheduling and compute management system, the SWAMP platform has the ability to allocate computing resources across the competing needs of the platform’s users.

### 3.5.2 Software Packages

A software package is the unit of code assessed in the SWAMP. In order to assess a software package, a copy of the code to be assessed along with the package settings is required. The package settings are used to determine what to assess in the software package. This section describes software package and package settings.

Software packages often have separate configuration and build steps. The optional configuration step may configure the build step, create necessary files, and compile tools and other ancillary pieces of software. Software compiled during the configure step should not be assessed.

The following principles were followed for software packages:

1. Users are required to provide their software package to the platform along with information (package settings) that describes how to build the package.
2. The software package must build successfully to be assessed.
3. The software package may have an optional configuration step that is run before the build step and files compiled during this step should not be assessed.
4. Only the files that were built during the build step or otherwise specified are assessed.

The assessment requires a copy of the source or binary code along with information that the assessment process uses to determine how and what to assess in the software package. The information required to configure the tool and perform the assessment is quite detailed and includes the complete list of files that should be assessed and their characteristics. This can be beyond what even a well-experienced developer can reasonably be expected to provide accurately. We require the user to provide only basic information, such as code directories and how to build the software. From this information, the SWAMP determines how and what to assess by monitoring the build process for the software and inspecting files in the software package.

The assessment framework expects each software package to consist of a single archive containing the code to be assessed and the package settings files. The package settings files describe how to configure and build the software; and any OS-level software packages that need to be installed before the assessment process begins. The package archive and package settings files are placed in the input directory that is mounted within the OS Platform and are processed by the assessment automation framework.

Both the archive and package settings are uploaded, fetched or otherwise acquired using the SWAMP platform's user interface, scheduling system, or APIs.

The package settings that are used to determine what to assess vary based on the language. Settings common to all languages include the top level directory of the software packages, the directories for the build and configuration, the configuration command, language version, the build system to use, and options that are used to parameterize these steps. Other settings are language specific such as language

version, build system version, and paths to include or exclude for libraries and source files.

For languages that are not compiled, the build step may still exist to fetch library and source dependencies and to set up an environment to run the package. For these packages, the package settings may include a list of files or directories to exclude or include in the assessment.

### 3.5.3 Curated Software Packages

Part of the project's initial mission was to provide a collection of curated software packages for researchers and tool developers to use to evaluate and improve assurance tools. These packages were selected by the SWAMP staff and made available to all users to assess with the assessment tools available in the platform. As such, they could also be used as example and test packages. Curated packages were also used internally for testing of the platform as it was being developed.

The packages chosen are either popular open source packages or packages that are part of test suites injected with specific security flaws.

The following principles were followed to select curated software packages that would:

1. Cover the major combinations of build systems and build settings for each language supported in the SWAMP.
2. Represent a variety of different application categories of software.
3. Present interesting security properties

One criterion for selecting these packages was to represent the supported languages and build systems (and package settings variants). Another criterion was to provide diversity, both in the size and complexity of the software, and in the general purpose and type of the software, such as web application, graphical interface, command line interface, server, and client.

Another criterion was to include packages that had interesting security flaws. These curated packages come in two forms. The first: actual software packages containing a specific security flaw such as the Heartbleed vulnerability in OpenSSL (a later version was created to see if tools could detect the weakness). The second: software packages that are part of synthesized test suites. The later included software packages that are part of the National Institute for Standards and Technology (NIST) Juliet Test Suite for C/C++, NIST Juliet Test Suite for Java, software that was used in one of the NIST Static Analysis Tool Exposition (SATE) competitions, and test suites produced as part of the Static Tool Analysis Modernization Project (STAMP) by GrammaTech's© Bug Injector tool.

Over the duration of the project, thousands of curated test packages have been made available in the SWAMP's open, public facility.

### 3.5.4 Static Assessment Tools

Support for multiple static analysis tools was an initial and continued requirement for the SWAMP platform. There can be one or more static analysis tools available for each programming language, while some programming languages have none. The project's goal to lower the threshold for continuous software assurance was to support assessments with freely available, open source tools. Later we were able to support commercial and academic tools through partnerships with the owners of these tools.

Since tools differ in the types of weaknesses they report and the algorithms they use, the use of multiple tools increases the coverage of weaknesses reported. This also allows for specialized tools to compliment the gaps that tools with broader capabilities miss. Due to the considerable cost of on-site operation, users of tools will typically limit their use to a single tool or a small number of tools.

Many tools support configuration options that are difficult for users to set to reasonable values. The configuration may control the types of weaknesses reported, the algorithms used, or parameters of the algorithms. These tools assess the user's software package and produce results in a native format.

The tools made available for users of the SWAMP are installed, configured, and driven by the assessment frameworks that orchestrate using the data determined from the package settings and from building of the software package. The project configured these tools to support general usage and tried to configure tools to eliminate noisy low-quality results from the output.

To make it easier for users to use multiple tools, the output from the platform must be in a common results format. The common result format used is SWAMP Common Assessment Result Format (SCARF) and is further discussed in section 3.5.8 Assessment Results Output Files. This requires conversion of the native results output of the tools to a common results format that will be universally understood by the SWAMP platform.

Supporting multiple static analysis tools resulted in the following principles:

1. The SWAMP must allow users to easily assess their package with multiple tools.
2. The tools must be able to run successfully in an automated fashion on most packages and must run without interactive human input
3. Tool selection is based on a combination of usefulness of results, popularity of the tool, and limiting the set of tools per programming language.
4. Tools configuration should be appropriate for general usage.
5. The SWAMP platform must convert native tool results output to a common results format.

Once integrated into the SWAMP platform, tools need to be updated as new releases are made available. The expense of these updates ranges greatly: it can be minimal for the project staff (mainly to do testing) for minor updates, or it can have a large cost if the tool, its configuration or output changes significantly.

#### 3.5.4.1 Static Tool Selection and Configuration

The quality of open source static analysis tools varies greatly, so only tools that produced useful output and worked without failure on most packages were considered for integration with the platform. Tools were selected by evaluating their standing in the community and by their suitability for use in the platform. For each language we researched the availability of open-source, static-analysis tools. After selecting the top couple of tools, we further evaluated the tools by making a prototype tool configuration to evaluate how each assessed the SWAMP curated packages.

Some tools were eliminated due to their need to have an interactive human operator that is not supportable in the SWAMP platform. Other tools were eliminated due to their failure to successfully assess most packages in the test suite without crashing. More tools were further eliminated due to the results they produced: high false positive rate or the results being of a low quality that were not useful to a developer. Finally, tools may have been eliminated due to the produced results being largely a subset of another tool that is superior in some way.

Once a tool has been selected, the configuration options were reviewed to determine optimal settings for operating as best as possible for general use. This tuning process is iterative and requires running the tool, reviewing results, and modifying the configurations as needed. Most tools only allow a limited set of tuning options so the results output may not be ideal.

Once the tuning process was complete, the tool was packaged as a SWAMP supported tool. This entails creating an archive of the tool and tool settings data that is used by the assessment frameworks to install, configure and operate the tool. There may be more than one instance of the tool installation files as different versions may be required on different OS platforms, such as tools that have different installations on 32-bit and 64-bit OS platforms. For some tools, additional support for installation, configuration, or operation, was built into the assessment framework code to simplify complex tasks required to operate the tool.

The SWAMP-supported tool packages for some tools that support operational configuration options also allowed for a controlled set of these options to be set as a system-wide SWAMP tool configuration in a SiB deployment. Care was taken to allow only a subset of possible options to be configured in this way to prevent conflict between settings and ensure configurations supported by the assessment frameworks.

SWAMP supported tool packages have been added to the SWAMP as private tools for use only by specific tool developers and researchers, allowing them to use the SWAMP's curated software packages and scalable facilities to do research with their tool.

SWAMP supported tool packages have also been added to the platform for Commercial tools for which the project was able to obtain a license agreement with the vendor. These tools usually introduce further complexities such as license server, remote server access, or additional data not required by other tools. These tools are made available to

SWAMP users based on criteria set by the tool owner such as only providing access for educational use.

Commercial tools are also supported by SiB if the user has access to the commercial tool. This was accomplished by a script that took the tool installation files provided by the vendor and packaged them into a SWAMP tool that could be installed in the user's SiB environment.

Below is the list of tools that are currently supported by the SWAMP or that have been at some point in the past. Experimental tools that were used by their developers only indicated by (e), commercial tools by (c), and removed by (r).

- C/C++
    - Cppcheck
    - Clang Static Analyzer
    - Gcc Warnings
    - Parasoft C/C++test (c)
    - GrammaTech CodeSonar (c)
    - Synopsys Static Analysis (Coverity) (c) (r)
    - HRL Laboratories Tunable Information Flow (TIF) (r)
- Java including Android
    - FindBugs (later SpotBugs)
        - with FindSecurityBugs and fb-contrib plug-ins
    - Error Prone
    - Programming Mistake Detector (PMD)
    - Checkstyle
    - Open Web Application Security Project (OWASP) Dependency-Check (r)
    - Parasoft Jtest (c)
    - Sonatype Application Health Check (c) (r)
    - Indiana University Purdue University Indianapolis (IUPUI) Static Code Analysis Tool Evaluator (SCATE) (e)
    - Archie (e)
    - Virginia Tech CryptoGuard (e)
- Android Java/Android Application Package (APK) only
    - Android-lint
    - Reveal Droid
- Python
    - Bandit
    - Flake8
    - Pylint
- Ruby
    - Brakeman
    - Dawnscanner
    - Reek
    - Rubocop
    - Ruby-lint (r)

- PHP
    - PHP Mess Detector (PHPMD)
    - PHP_Codesniffer
- JavaScript
    - ESLint
    - Flow
    - JSHint
    - Retire.js
- Hypertext Markup Language (HTML)
    - HTML Tidy
- Cascading Style Sheets (CSS)
    - CSS Lint
- Extensible Markup Language (XML)
    - XML Lint
- .NET
    - Code Cracker
    - Security Code Scan
    - Devskim
- Code Metrics Tools (most languages above)
    - Cloc
    - Lizard

### 3.5.5 Operating System Platforms

An operating system platform is an environment running a standard operating system and providing the files and execution environment of the OS. Part of the SWAMP's initial mission was to provide support for running assessments in multiple OS platforms. Each OS and OS version has different versions of installed libraries and development tools. Development tools and libraries can affect how the software is built. This includes the version of the programming language supported, and the library interfaces available. These differences can affect which files are compiled and how they are compiled.

The OS platforms should include a broad set of development tools and libraries pre-installed to make assessments using the OS platform work without the user having to manually add those tools and libraries. This also increases the number of packages that can be assessed easily in OS platforms on a SiB deployment in an isolated environment that does not allow the network connectivity needed to fetch additional OS-level packages.

However, even with a good selection of pre-installed development tools and libraries, some software packages may have unusual requirements. For those, the assessment frameworks provide a means to install user-specified OS-level packages prior to building and assessing a package.

Supporting multiple OS platforms resulted in the following principles:

1. The platform must allow users to easily assess their package on multiple OS Platforms.

2. The OS Platforms should have most common development tools and software development libraries pre-installed.
3. Users should be able install additional OS-level packages prior to the build and assessment of their software package.

All the OS platforms in the platform are Linux-based OS distributions. Multiple 64-bit versions of each distribution are supported along with 32-bit variants of some distributions. The multiple distributions and versions ensure that a wider selection of software can be built and assessed. This is because some software packages will only build (and therefore assess) with specific versions of an OS, its tool chain, and libraries. Also, what and how the software is built (and therefore assessed) can depend on the OS, the build tool chain, availability of libraries or versions of libraries, resulting in flaws that only appear on a subset of the platforms.

The following OS distributions have been supported in the SWAMP platform:

- Debian
- CentOS
- Fedora
- Red Hat Enterprise Linux (no longer supported due to licensing)
- Scientific Linux
- Ubuntu

These OS Platforms are created as virtual machine images that are deployed using a hypervisor. This provides isolation, a consistent starting environment, and scaling. When the project started, this was the only solution to fulfill these requirements. Recently, container technology, such as Docker, is an alternative to the heavier weight VM technology. Containers require less resources and work in VMs rented from services like Amazon Web Services that only allow nested virtualization at a high cost. The SWAMP platform provides Docker assessment containers as an alternative to VMs, allowing for installation and use of SWAMP-in-a-Box in cloud services.

### 3.5.6 Programming Language Support

Programming languages are the primary categorization of software packages and tools. Generally, software packages are written in one programming language and software tools support a single language such as C or Java. Some software packages contain source files written in more than one programming language, and some assessment tools may support assessing more than one type of programming language.

A subcategory of languages are the build systems used to take the source files of the software packages and create a runnable version of the software. The build system is important as it easily allows the user to describe how their software is built (which is one of the requirements for software assessment). The build system is the mechanism to determine the files to be assessed, how the tool should be run, and may also include a package manager that downloads dependent libraries from repositories available on the Internet or in local repositories for strict security situations where Internet access is not permitted.

Supporting programming languages resulted in the following principles:

1. The platform must support assessments from multiple programming languages.
2. Using the software package settings, the platform should be able to build and assess any of the supported languages used by the software package.
3. Selection of supported languages is based on a combination of popularity of the language and availability of appropriate tools.
4. Support for build systems is based on the popularity of the build system and the ease of supporting it.

Programming language support means that the SWAMP can assess the source code or binary files of that language by using static assessment tools. This means that the assessment framework understands how to collect the information required to determine the files to assess, and how to collect and pass the additional information that the tool requires to perform the assessment. This also means that the framework understands how to operate and extract data from the supported build system.

The supported languages and build systems are shown below. In the table below, the build system "any" means any and all build systems that can build using a single command or script; while the build system "files" means that the files are discovered by searching for files with an appropriate file extension. Most languages support some type of "files" build system, while those shown as "files" only support the "files" build system.

Many of the languages supported in the SWAMP have multiple versions that are different dialects of a common language. The SWAMP supports several recent versions of most languages, as shown in Table 1.

**Table 1. Supported Programming Languages and Build Systems**

| Language | Build Systems |
|---|---|
| C | any |
| C++ | any |
| Java | ant, ant+ivy, maven, gradle |
| Java Bytecode | files |
| Android Java APK | files |
| Python | setup tools, PIP, wheels |
| Ruby | rake, gem, bundler |
| PHP | pear, composer |
| JavaScript | npm |
| HTML | files |

| | |
|---|---|
| CSS | files |
| XML | files |
| .Net | msbuild |

### 3.5.7 Assessment Automation and Assessment Automation Frameworks

The whole process of performing an assessment must be automated in the isolated assessment environment. Supporting the assessment automation results in the following principles:

1. There needs to exist an assessment framework that automates the process of performing an assessment given a software package with package settings, an OS platform and a static assessment tool.
2. The assessment run produces a common results output format such as SCARF

The assessment automation framework is a program and other files that automate the entire process to assess a software package and produce an output file containing the results of the assessment. It orchestrates the entire sequence of steps that is performed within the OS platform running in a VM or Container including:

1. Install OS-level packages
2. Install software package
3. Configure the software package
4. Build the software package with monitoring
5. Determine the source files and any other data necessary to run the tool from the monitoring data
6. Install the tool
7. Configure the tool
8. Run the assessment tool
9. Convert the output to the common format

Most of these steps involve simply starting and waiting for processes to start and complete. The step of building the software package and determining the source files and any other information required to run the tools is more involved. There are three different mechanisms used to determine this information.

The first is used for C and C++ packages and involves monitoring kernel calls made by the build command's process and all processes that it started, directly or indirectly, to determine each process’s command line arguments, environment, initial working directory, and exit status. From this data the set of compilation processes and other development tool processes that were started can be determined. From their command lines, environments, and initial working directories, the command line can be decoded to determine information required to assess the software packages.

This is the most general mechanism and highest fidelity, but the overhead of this approach in terms of time and memory is significant, and the expense creating the code to analyze the build tool options is considerable.

The second mechanism is used for Java packages. It involves creating and installing plug-ins into the build systems used by Java; one each for ant, maven, and gradle. When files are compiled by the build system, the build system calls the plug-in with the information and it can then record this information.

This mechanism requires one plug-in per build system and requires them to be maintained and updated if the build system's plug-in architecture is updated. It has the advantage that it introduces minimal overhead. Although rare, another downside is that any activities done outside the build system (calling a script for instance or calling the build system from a script) can cause this technique to fail.

The final mechanism used by the other languages is to run the build and then examine files on the disk to determine the files to assess based on file contents or file names. For instance, all files with an extension of .xml under a certain directory should be assessed as XML files. This technique has relatively low overhead and can incorrectly include or exclude files if the file names or contents are not correct.

### 3.5.8 Assessment Result Output Files

The results from performing an assessment with a static analysis tool is a single file or set of files containing a set of weaknesses along with their location in the analyzed files. Each tool outputs a unique format. With unique tool formats all consumers of SWAMP output, such as SWAMP result viewers and end users, would have to be able to process the more than 30 different file formats. Each consumer and viewer would have to be updated each time a new tool was added or tool output updated. These varied formats also prevent consolidation of the data of the tools, preventing a coherent view of all flaws located in a software package. To solve these issues a common results format is required.

When the project was started, there was not a suitable format that provided a super set of all the results generated by the tools supported in the platform. The SWAMP project created a new format loosely based on the format of the FindBugs tool. This format is the SWAMP Common Assessment Result Format and is described at https://github.com/mirswamp/swamp-scarf-io. This repository also contains libraries to easily consume and produce SCARF files.

After the assessment completes, the assessment framework starts the SWAMP result parser. The result parser converts the native tool output to SCARF. When a new tool or version of a tool is added, it may be necessary for the result parser to be updated to support complete conversion of the native tool output.

Recently, a new initiative was started by Microsoft under the Organization for the Advancement of Structured Information Standards (OASIS) to create and standardize a new more full-featured format called Static Assessment Result Interchange Format. The

project participated as a founding member of the Technical Committee and was a key contributor to completing the standardization process. More information about SARIF can be found at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=sarif or https://github.com/oasis-tcs/sarif-spec/. The assessment frameworks and result parser support the production of SARIF output files in addition to SCARF, but the code does not yet exist in the higher-level to store and allow access to the SARIF files.

### 3.6 Integration with Programmer's Workflow

Initially, the only way for users to interact with the platform was using a web-based user interface. This interface was outside the normal workflow of software development teams and resulted in friction to the adoption of using the platform in their workflow. To reduce this friction, technology was developed to allow the platform to fit more naturally into the developers workflow through the use of a Java API, command line interface (CLI), and plug-ins to support using the platform in the following: integrated development environments, continuous integration systems, source code management systems (SCMS) or on-line repository systems. The rest of this section provides a summary of this work, and a paper "From Continuous Integration to Continuous Assurance" [2] provides further details on this functionality.

The first step was to create an easy to use API in Java and a CLI that allow easy remote access to common features of the platform such as uploading a package, performing an assessment, checking its status, and downloading results. A developer can use the API or CLI to integrate their workflows directly with the platform, or they can use one of the other mechanisms below, if applicable to their development workflow. This API and CLI were used to create the other plug-ins.

The first plug-in was for the Eclipse IDE. It allows users to click a button and have their Eclipse project assessed in the SWAMP platform. All within the Eclipse IDE users can view the status of the assessment, view results, and fix weaknesses.

The second plug-in is for the Jenkins CI system. It provides a means to include an assessment using the platform as one of the tasks performed during the build and test of a software package. The results are visible in the Jenkins dashboard and individual results can be viewed directly in Jenkins.

Another integration is a hook for the SCMS systems git and subversion. These SCMS systems allow a hook to be called when specific actions are done with the repository, such as a check in or merge of new or modified source code. These hooks are used to upload the source code of the new commit to SWAMP and to trigger an assessment. As the repository has no user interface, results can be viewed using the SWAMP project's web-based viewers or downloaded and processed locally.

One final integration is support for a GitHub webhook that can be configured to have GitHub open a web API on the platform and communicate the commit id of the new commit. The SWAMP platform then uses this information to fetch the new code and start an assessment of it.

# 4.0 RESULTS AND DISCUSSION

## 4.1 Initial Operating Capability and Post – Initial Operating Capability

The SWAMP platform was designed and implemented to support continuous assurance by providing a simplified and automated process for running code analysis tools, specifically static code analysis tools. The platform was built from the ground up, in 14 months, by a newly hired team of developers and system administrators, to easily integrate into the continuous software assurance workflow. A high degree of security was put in place to ensure that developers could trust the SWAMP project with their source code and to protect the SWAMP infrastructure from malicious attacks. The requirements listed in the Broad Agency Announcement (BAA) 11-02 Technical Topic Area #14 were met at Initial Operating Capability (IOC), which was to provide a web-based, software assurance facility offering access to multiple platforms, five static analysis tools, and a minimum of 100 open source software packages for testing.

The SWAMP platform went live February 3, 2014 as an open, hosted, continuous assurance platform at mir-swamp.org, supporting eight versions of Linux platforms, five open source tools, three programming languages, and 586 open source packages. As the project progressed and project staff interacted with developers at conferences, it became clear that a local instance of the SWAMP platform was needed for broader adoption. Developers had reservations about uploading their code to the cloud, into a cloud facility funded by the Federal Government, and were restrained by company-set security or intellectual property rules. Project staff created the idea of a single server instance called SWAMP-in-a-Box. With the open source release of SiB on September 27, 2016, the hosted platform would later be referred to as an open, public facility.

As the project continued, more platforms, supported static analysis tools, and programming languages were added. Additionally, new versions of existing platforms and tools were added. Successful collaborations with some commercial companies allowed access to their proprietary static analysis tool through the SWAMP's open, public facility. Access to the commercial tools was determined by a software license agreement between the Morgridge Institute for Research, on behalf of the SWAMP project, and each commercial company. The project's most successful collaborations were with Parasoft and GrammaTech. Both companies allowed educators and students access to their commercial tools and would grant access on a trial basis to all other users on a case by case basis. Some companies felt it was too risky to allow access to their proprietary software through the SWAMP's open, public facility. With the release of SiB, the project positioned SiB as an option for companies to integrate their proprietary software with the platform through SiB only. SWAMP staff would complete development to support the proprietary static analysis tool in SiB. Users of SiB would need to contact the company to purchase a license to integrate the proprietary static analysis tool into their local SiB instance.

As the project evolved from an open, public facility to also include a downloadable, open source solution with SiB, the demand for new functionality and features also grew. The project staff realized the importance of providing direct access to the platform from

integrated development environments such as Eclipse, source code management systems such as git and Subversion, and continuous integration systems such as Jenkins. SWAMP plugins for Eclipse, git, Subversion, and Jenkins were released on the project's GitHub organization page for download. A Java-command line client was also created for developers wishing to bypass the SWAMP project's user interface.

Integration with third-party identity management resulted from the relaxing of security restrictions on who could create an account in the SWAMP's open, public facility. At IOC, only organizational email addresses were allowed to create an account to prevent false or malicious accounts from being created. No free mail addresses were allowed, such as Gmail, Hotmail, yahoo, etc. This was a significant barrier to adoption of the platform. The platform was able to support educators by adding CILogon, a federated identity management software. Universities that became participants in CILogon were able to use their respective university user account credentials to create and login to the SWAMP facility. Integration with Google identity management and linking GitHub accounts also created additional options for users to create accounts without having to create a new set of credentials to remember.

Six commercial tools were integrated with the platform and six software license agreements were executed. Two commercial companies entered into agreements with the project through letters of intent, Parasoft and Code Dx, respectfully. One private tool, Archie, was integrated into the platform for use only by the developer. One commercial tool (Programming Research – Quality Assurance, PRQA) was not successfully added to the platform prior to being acquired by a larger company. The project was not able to successfully add support for VxWorks by Wind River into the SiB platform due to funding and time constraints.

Towards the end of the SWAMP project, the platform supported over 30 static analysis tools, over 20 platforms, 10 programming languages, and offered access to over 3,774 software packages. Since IOC, the project had 40 stable releases of the SWAMP facility software and 12 stable releases of the SiB software.

### 4.2 SWAMP-in-a-Box

SWAMP-in-a-Box is a standalone version of the SWAMP platform. It is, in essence, a local instance of the platform that can be deployed by individual users and organizations onto their own servers. Conversely, the facility can be viewed as a multi-server version of SWAMP-in-a-Box, where the use of multiple servers allows the facility to more easily handle extremely large loads, e.g., hundreds of students in a university class using the SWAMP to assess code for homework assignments and course projects.

SiB allows individual users and organizations to take advantage of the platform capabilities while maintaining full control over and the privacy of their software packages and assessments results, because neither the packages nor results are communicated with a third party, unlike with the SWAMP facility. Furthermore, SiB can be configured to function without access to the Internet, thus making it suitable for use in environments

with restricted network connectivity, such as sensitive compartmented information facilities.

Being open source (source code is available on GitHub), SiB allows individuals and organizations to customize the platform for their own needs, a feature that is not available when using the SWAMP facility. For example, Raytheon was able to customize the assessment platforms used by their SiB instance and to continue using some commercially available tools after the project ceased to provide official support.

This version of the platform started as an aid for developing the platform without needing to replicate the multi-server setup of the SWAMP facility for each developer. It grew into its own, publicly available offering when it was realized that many potential customers of the project were unwilling to share their software packages and assessment results with a third party, even with the strict privacy and security guarantees provided by the SWAMP facility.

SWAMP-in-a-Box consists of the following components running on a single host machine:

- A web server (Apache Hypertext Transfer Protocol (HTTP) Server) that provides the SWAMP web application to users and an API for programmatic access, e.g., via plugins for Jenkins, Eclipse, and source control systems, such as Git.
- An Structured Query Language (SQL) database (MariaDB) that stores information about users, packages, assessments, and assessment results.
- An execution backend that uses HTCondor for scheduling and running assessments.
- A file system that provides persistent storage for packages uploaded to the platform, results from assessment runs, and the assessment platforms and tools used to assess packages.

SWAMP-in-a-Box is distributed as a collection of packages and scripts that install and configure these components of the SWAMP on a host machine. The host must already be set up with an installation of the CentOS Linux distribution prior to installing SiB. The remainder of SiB's software dependencies are handled by the installation scripts. In the event where the host has restricted access to the general Internet, instructions are provided for how to obtain and install the dependencies manually.

The hardware requirements for the host scale with the number of simultaneous assessment runs that the system needs to be able to support. In terms of central processing unit (CPU) cores and memory, this can lead to a substantial load because each assessment is run in its own virtual machine or Docker container that is provisioned with 2 CPU cores and 6 GB of memory, by default. When it is known that each assessment will not make use of its full allocation of cores, SiB can be configured to schedule assessments as if the host has more cores available than it actually does (when using virtual machines for assessments [a technical limitation of the virtual machines requires that they be provisioned with two cores each, even if assessment itself only makes use of one core]) or to provision each assessment with only one core

(when using Docker containers). Similarly, when it is known that each assessment will not make use of its full memory allocation, SiB can be configured to provision each assessment with less memory.

## 4.3 VMs and Docker Containers

This section describes the VM and Docker containers used in the project. It first describes how they were created and then how they were installed, managed, and used to perform assessments.

### 4.3.1 Creation and Content

The VM images used as SWAMP platforms were created by installing an OS image that contains most development tools and libraries required to build software packages. Having a robust set of development tools and libraries increases the size of the VM somewhat, but it ensures that many software packages will build without modification. In addition, virtualized drivers that accelerate input/output (I/O) via the hypervisor may be installed to boost performance. The only requirement for these VM images is that they allow two volumes to be mounted within the VM (an input and output disk), and they start a script located in the input disk, if present, when the VM using the image is started. The input disk is populated with files related to the software package, static analysis tool, result parser, and assessment framework. The output disk is where the output of the build, assessment and results are placed by the assessment framework.

The Docker images were created by copying all the files in each corresponding VM image into a Docker container image. Docker images also have an input and output volumes, and essentially, operate the same as the VM would. Since the SWAMP assessment frameworks require root access to be able to install OS-level packages, the docker container must be configured to support this. Containers generally provide less isolation than a VM, and allowing root access in a container further reduces the isolation between software running in the container and the kernel running the container. Since the build runs arbitrary code, the use of containers instead of VMs is a security concern if the build process can contain untrusted or malicious code.

Updating a VM image is a fair amount of work for minor updates, and supporting a new distribution or major version update can be significant as the system administration between major versions can change significantly. Platforms that require licensing were not supported due to the amount of effort it would take to manage the configuration and communication with licensing infrastructure, in addition to negotiating legal distribution mechanisms.

### 4.3.2 Management, Installation, and Operation

The Morgridge team distributed VM images created and supplied by the University of Wisconsin (UW) team to the execution nodes used by test systems and, after additional testing, by the open SWAMP facility. This was done using a Distributed Gluster file system backed by an EMC VNX system to provide plenty of space to keep multiple

versions of the images, and to ensure that all of the nodes would have access to and be using the same versions at all times.

In order to support updates to and installation of dependencies within running Assessment platforms, an on-site mirror of distribution repositories was maintained for use by assessment virtual machines. This solved the problem of external network interruptions or changes, and, when running hundreds of simultaneous jobs that reference the repositories, to prevent the platform from being blacklisted for trying to flood public repository connections. This also allowed us to easily version lock packages by excluding them from the local repository.

SWAMP-in-a-Box initially included only one VM platform in the installation package. SWAMP-in-a-Box administrators could add additional VM platform images and, eventually, Docker container platform images to their SWAMP-in-a-Box instances after downloading them from the same site providing the SWAMP-in-a-Box installation files.

Per host management was done with Puppet and Ansible to adjust configurations, install packages, and make sure things were configured as intended.

Builds of the software, including running the applicable parts through the SWAMP itself was managed by the Open Source Jenkins software

### 4.4 Use of the SWAMP Platform in the Cloud

As a standalone instance of the SWAMP platform, SWAMP-in-a-Box can be installed on any host meeting its software and hardware requirements. This includes hosts that might be running “in the cloud,” e.g., in virtual machines running in Amazon Elastic Compute Cloud (Amazon EC2), Google Compute Engine (GCE), and Microsoft Azure Virtual Machines. This approach might be appropriate for individuals and organizations that do not own their compute infrastructure or that prefer not to acquire dedicated hardware meeting the requirements for SiB.

The primary limitation in such environments is that running assessments using VMs might not be possible. The assessment VMs require that the host VM supplied by the cloud provider support nested virtualization, which might not be an available feature or incur additional costs compared to VMs that do not provide this feature. However, running assessments using Docker containers should still be viable.

The project has successfully installed and tested SiB on Amazon EC2 instances.

### 4.5 Lessons Learned

The SWAMP project brought together researchers from four academic institutions in response to a vision and IOC requirements articulated by a BAA. As researchers, each and every project serves as a learning experience that advances our knowledge we are committed to share: it reinforces principles; it qualifies and quantifies challenges; and it offers “a-ha moments” that expose unknown aspects of a problem. This knowledge will serve the community as we continue to use open source software and open access

services to advance continuous software assurance and other national challenges. We present the main elements in each of these categories.

### 4.5.1 A-ha moments

- The concept of continuous assurance is significantly strengthened when assessments can utilize multiple tools. To fully leverage this idea, users who invest significant effort to annotate the results of one tool **need a mechanism to transfer these annotations** to other tools. This requires a protocol to exchange annotation between tools and buy-in from the tool developers, which was not in the original scope of the project. An effort to secure funding to develop such a protocol and to get commercial tool developers to expose annotation did not succeed.
- A company's software IP and the result of assessments (potentially highlighting security issues) are highly sensitive data: despite being a project funded by the US government, **distrust of external entities to handle this data runs very deep in the community**. The project even ran into significant reservations about the Federal Government using SWAMP as a means to steal intellectual property. This came to light following the responses to a press release about the project in Network World. The community response prompted the project leadership to receive an official letter from Department of Homaleand Security – Science and Technology Directorate (DHS S&T) stating the Federal Government had no rights or access to data uploaded or produced.
- The wealth of data provided by commercial **result viewers can be overwhelming for students and newcomers** with limited programing or software assurance experience. In an educational setting, viewers should be adjusted to the skills and experiences of the students and hands-on assignments

### 4.5.2 Challenges qualified or quantified

- Most software development projects do not consider (continuous) software assurance an important element of the software development life cycle. Even fewer bother to utilize more than one assessment tool. While we have experienced some increase in interest over the life-time of the project, it often takes a significant encouragement or value proposition to convince teams of the value of software assurance.
- Federated identity management (single sign on) is highly valued by users, especially when starting to use a new service like SWAMP.
- The software assurance ecosystem is strongly influenced by the concerns and interests of commercial tool providers. Open source platforms and open assessment services primarily offer value to users, not providers. Hence, it is difficult to convince the tool providers to invest in open platforms. We did make some headway in helping to standardize a results interchange format (SARIF – an OASIS standard since March 2020) but more time is needed to develop the open platform approach.

- Maintaining a platform with over 30 tools and 19 platforms as tools and technologies evolve requires significant effort. Thus, as the platform matures, this maintenance impacts the ability of the project to add new capabilities and features.

### 4.5.3 Principles Reinforced

- Effort with a long-term commitment and stable funding is required to transform an ecosystem by means of an open-source platform or open suite of services. This principal holds the key for adoption as it underpins the trust relationship with the user; particularly, government or industry users have a much longer time horizon than a typical research project. Namely these users are interested in answering the concern of “will you be here tomorrow?”.
- Talent recruitment and Human Resource procedures and processes are critical. Building, managing and sustaining an effective team of researchers across four institutions has been key to the ability of the project to deliver services, software, education and outreach.
- Committed and engaged users are irreplaceable. They provide the guidance, requirements, and evaluation feedback that is missing from the “we will build and they will come” approach to developing software and services. The group of tool developers funded by S&T - envisioned by the BAA - did not materialize. This resulted in a significant shift in the modality of the project to cast a broad net to attract software developers.
- Building a meaningful community is an adaptive challenge that requires strategic and opportunistic activities. Long term strategic and short-term tactical alignment within the constraints of available effort and on-board talent are critical. Shifting of effort to address new requirements and opportunities has to take place in framework where tradeoffs are recognized and evaluated to address broader impacts. This requires requiring close and frequent communication with the funding agency. In face of the dynamics of the software assurance ecosystem and the lack of a committed core group of users, the project had to continuously adapt to changes in priorities and target opportunities to grow the community and deliver new innovations.

# 5.0 CONCLUSIONS

## 5.1 Project Achievements

The SWAMP project was built and executed on the foundation of a commitment to the goal of promoting effectiveness and adoption of software assurance. The project pioneered the concept of "*continuous* software assurance" and followed a multipronged approach to create an open source, portable continuous assurance platform that addressed the needs of an evolving ecosystem of software assurance practices. The SWAMP strategy targeted software developers, tool developers, educators and researchers. With these users in mind, the project created an open platform that demonstrated the power of continuous software assurance. The SWAMP public facility and SWAMP-in-the-Box software provided a working blueprint for the architecture and functionality of a continuous assurance capability with the ability to be fully integrated into the software development life cycle. The project also identified gaps in available technology that guided and advanced future R&D in software assurance. By operating a public marketplace, the SWAMP project brought the power of hands-on, continuous software assurance to individual developers, small development groups, class rooms and training sessions that would not have otherwise been able to access such resources without being in large organizations with a well establish software assurance program.

### 5.1.1 Built an Open Market Place

The SWAMP project′s initial operating capacity was a cloud-based platform that went live on February 3, 2014. This platform offered 586 open source testing packages, eight Linux platforms, five static analysis tools, supported three programming languages, and later became known as the SWAMP project′s open facility. The public facility was built on robust hardware for the IOC time frame; containing 700 cores, 5 TB of RAM, and 104 TB of disk space. This allowed for up to 275 million lines of code to be analyzed per day. Later the SWAMP project was able to acquire additional computing equipment from DARPA′s Cyber Grand Challenge and add an additional 1,280 cores, 16 TB of RAM, and 128 TB of disk space to further support the computing capacity of the open facility.

Given the computing capacity and proven scheduling management software, HTCondor, supporting the backend software, the open facility provided a testing ground for individual developers and smaller teams, that would normally be resource limited, to incorporate continuous software assurance practices into their software development lifecycle; by lowering their barrier to adoption. The open facility also proved to be a valued resource for educators looking to incorporate new technologies and practices into their curricula, but were also limited by physical resources. The open facility was a way for educators to include continuous software assurance in their classes with full support from the SWAMP staff.

As features and functionality for the open facility progressed, additional tools, platforms, and languages were supported. The open facility would grow to support over 30 static

analysis tools, including six commercial tools, 20 platforms, 10 programming languages, and offered access to over 3,774 testing packages. In addition, the SWAMP project′s own native results viewer provided easily filterable, basic results viewing and was a good stepping stone for new developers and students to learn how to view and interpret results.

The open facility served over 2,870 users, conducted 24,203 user assessments, reported on 76,775,628 potential weaknesses and users uploaded over 5,223 of their own software packages to be analyzed. Through its entire duration, the SWAMP′s public facility scanned close to two billion lines of code.

### 5.1.2 Built the SWAMP-in-a-Box Platform

The SWAMP project created, distributed and supported, a standalone, local instance of the SWAMP′s continuous assurance platform. This allowed individuals and groups with privacy and security concerns to evaluate, and eventually, adopt continuous assurance practices by deploying their own SWAMP platform, under their own firewall and security control measures, reducing the potential for outside individuals or groups from obtaining assessment results. Local deployments featured integration with organizational identity management and computing resources to support scalability.

### 5.1.3 Built a National SWA Community

The SWAMP project was driven by a principle of lowering the barriers to adoption of continuous software practices and software assurance tools. To fulfill this principle, the SWAMP project created a GitHub repository to offer the SWAMP project′s open source software to the greater software assurance community. All SWAMP software was licensed under an Apache 2.0 software license making the software free to download and modify as the user′s needs required.

Leveling the playing field for small software assurance research projects and technology start-ups, the SWAMP project provided a variety of open source components that allowed the tool developer to concentrate on their new technologies and not on the mechanisms needed to make a tool usable and competitive with the large commercial ventures. This software included automated build monitoring and tool launching for C/C++, Java (for all major build systems), and the major scripting languages. The results parsers mentioned below, in Section 5.3, contributed to this leveling effort.

## 5.2 Project Findings

### 5.2.1 The Issue of Trust

Despite offering a robust and secure open facility for static analysis, there are barriers that still exist when adopting continuous software assurance practices. These barriers revolve around building trust between a cloud-based solution and those practicing software assurance. Many do not feel comfortable uploading their software or having assessment results of their software in the cloud or are limited by institutional policies

and security concerns. The problem of trust increased in significance between the time of the project inception (2012) and the time of the project conclusion (2020) due to various well publicized breaches in trust and data security by various large cloud service and social networking data service providers.

The SWAMP project addressed these concerns by creating SWAMP-in-a-Box, a standalone, local instance of the SWAMP′s continuous assurance platform. This allowed individuals and groups with privacy and security concerns to evaluate, and eventually, adopt continuous assurance practices by deploying their own SWAMP platform, under their own firewall and security control measures, reducing the potential for outside individuals or groups from obtaining assessment results. Local deployments also facilitated usage of private assessment tools.

### 5.2.2 Broke Barriers with Interoperability

The SWAMP project brought the software assurance community together by standardizing the results of software assessment tools. In the past software assurance has been driven by individual tools and individual vendors. Each existing tool has its own idiosyncratic format that usually differs wildly from other tools including the tool itself, build process, results format, and viewer; all tightly integrated into a toolchain and non-interoperable with those from other vendors. This limited the ability to use multiple tools and to compare results between tools. In some cases, this ″siloing" is intentional so that users cannot easily compare results between tools, which may be advantageous to vendors, but does not aid the development and progress towards better software tools.

SWAMP was the first, and thus far, the only project that has constructed an open source platform that supports a tool-agnostic workflow that allows multiple tools to be run against a single software package, results to be translated into a tool-agnostic format (SCARF), and results to be displayed in a tool-agnostic viewer. The SWAMP project developed the SCARF standard to put the results of assessment tools into a standardized XML and JSON format. This means that any views, post processors, learning algorithms, or other tools need to only understand SCARF. The SWAMP project provided open source result parsers for a wide variety of tools to convert their output to SCARF.

### 5.2.3 Advanced the State of the Art of SWA

Due to the computing capabilities of the SWAMP hardware and the consistency the computing environment could provide, the SWAMP project was asked to host NIST′s Static Analysis Tool Exposition (SATE) V competition in 2013. All VMs for SATE V were hosted and run on the SWAMP′s computing hardware. Results were then viewed through Code Dx, a DHS S&T funded project at the time, that was integrated into the SWAMP platform as a results viewer by IOC.

Throughout the duration of the SWAMP project, we were approached by different tool developers asking us if the project could incorporate their static analysis tools under

development into the SWAMP platform for testing. We were also asked to incorporate new static analysis tools by tool developers so they could utilize the computing capabilities the SWAMP facility offered to run thousands of static analysis assessments. Some of these tool developers kept their tools private for their own testing, while others such as Red Lizard Goanna, wanted to collect usage data on their tool from SWAMP end users to improve their static analysis tool.

The SWAMP platform became a resource to widely distribute new testing suites for tool developers. These testing suites were developed under a DHS S&T funded project called the Static Analysis Tools Modernization Project (STAMP). The STAMP team utilized a private hosted instance of the SWAMP platform to test the newly developed software test suites prior to making them public in the SWAMP project′s public facility. The SWAMP staff uploaded and made public over 3,774 test packages as part of the Bug Injector test suite developed by STAMP. These Bug Injector test suites were in addition to the NIST Juliet test suites offered through the SWAMP′s open facility as a resource for tool developers.

## 5.3 Project Legacy

### 5.3.1 Demonstrating the Viability of an Open SwA Toolchain

The SWAMP project has proven that a tool-agnostic approach is technically viable and that the toolchain need not be tied to a particular tool and vendor. In the future, a tool agnostic approach will be required for SWA tools to work together and for the advancement of better software assurance tools.

### 5.3.2 Developing Software Assurance Standards

During the course of the SWAMP project, the project team played a key part in the development of tool agnostic standards and participated in the development of Open standards that continue to be in use today. SWAMP developed the SWAMP Common Assessment Results Format standard, which was a stepping stone to a newer format, embraced by industry and led by Microsoft, called SARIF. The SWAMP project had a major influence on the SARIF standard by serving as active members of the technical committee that led to the adoption and standardization of SARIF through OASIS. SARIF allows an integrated approach to results viewing and allows results to be concatenated and compared to see where tools agree and disagree.

### 5.3.3 Promoting Software Assurance Practices

As an outcome of the SWAMP project, the concept of continuous software assurance (continuous assurance) is now a widely accepted practice. Our team, from four academic institutions, pioneered this concept; extending the modern software engineering practice of continuous integration to include the analysis of the software by static analysis (security assessment) tools. Such a practice catches software flaws (weaknesses) at their earliest possible moment, reducing the incidence of these and reducing the cost of handling such weaknesses.

Our goal was and is to lower the barrier of adoption of software assurance tools by making these tools fit seamlessly into the programmer's regular activities. In practice, this was implemented in the platform by automating the launching of assessments from (1) the IDE with the push of a single button, (2) the continuous integration framework on each integration cycle, and (3) the software repository on each commit. As a result, the SWAMP introduced software assurance into the programmer's workflow, offering a practical open source example of applying the concept of continuous assurance in the software development lifecycle.

### 5.3.4 Training the New Work Force

The SWAMP staff proselytized software assurance and offered training to bring new teams to the practice of continuous software assurance. When the SWAMP project started, software assurance tools were used by a minority of developers, usually only in the biggest companies with established security programs. The SWAMP project team, by its steady appearance at conferences, workshop, PI meetings, and other venues, helped make software assurance a commodity and expected practice. In collaboration with Trusted CI, The National Science Foundation (NSF) Cybersecurity Center of Excellence, the SWAMP team developed training materials for practitioners to understand software assurance practices, understand the structure and purpose of such tools, and how to apply these concepts by using the SWAMP platform. Through these trainings, SWAMP staff also learned one of the biggest issues with software assurance practices was helping software analysts determine how to fix the weakness or vulnerability found in the analysis results. Trainings were adjusted and proved invaluable to attendees because they offered guidance in how to interpret and fix weaknesses and vulnerabilities found.

Through networking at conferences, meetings, workshops, and other venues, the SWAMP team continues to receive inquiries from educators asking about using the SWAMP platform in their software security or software assurance classes. Due to this continued interest in the SWAMP platform, the SWAMP software will remain available to the community through the Morgridge Institute for Research, where hosted instances of the continuous software platform will be made available to educators wishing to incorporate continuous assurance into their curricula.

# 6.0 RECOMMENDATIONS

## 6.1 Future of Continuous Software Assurance and the SWAMP Platform

Over seven years ago, the principal investigators of the SWAMP project saw the need to pioneer the concept of “continuous software assurance,” the practice of integrating security assessments into the entire SDLC, a natural extension of continuous integration. Through interactions with commercial vendors, established professional networks, and conferences, many saw the need for continuous software assurance, but did not have a way of implementing continuous software assurance practices in their own SDLCs or on a large scale. The SWAMP project was seen as a resolution to their software assurance needs, but did not easily fit into their SDLC. Other groups and companies began to improve their software assurance offerings. For example, GitHub released a new static analysis tool June 18, 2020 that has the ability to scan multiple languages. GitHub refers to this tool as a super linter [3] and has a well-established platform for easily incorporating static analysis into software development.

The principal investigators also see the need to continue to support the educational community as it builds curricula that includes continuous software assurance. Support for the educational community will continue with the Morgridge Institute for Research offering to host instances of a continuous software assurance platform. Professors Bart Miller and Elisa Heymann have developed video tutorials on secure coding practices and will continue to develop more tutorials as a resource to educators as well as providing in-person tutorials upon request. They were also successful in adding a secure coding course to the undergraduate curriculum at the University of Wisconsin - Madison. We anticipate that more universities will continue to follow by adding secure coding classes with an emphasis on software assurance in the undergraduate curricula; teaching the value of static analysis and instilling a habit of running static analysis tools regularly as a natural part of the development lifecycle. This in turn trains the future workforce in software assurance practices, better equipping them for their future jobs.

## 6.2 Unmet Continuous Software Assurance Needs

A current unmet need is workforce development in continuous software assurance practices. Professors Bart Miller and Elisa Heymann have taught tutorials at professional conferences, but these tutorials only reach a very small percentage of the current workforce. There needs to be more offerings for workforce development on what continuous software assurance is, the importance of continuous software assurance, and how to incorporate continuous software assurance in the current workforce’s SDLC.

## 7.0 REFERENCES


1. NIST Information Technology Laboratory, “Special Publications,” Computer Security Resource Center, URL: https://csrc.nist.gov/publications/sp, last modified June 16, 2020. Accessed January 2, 2013.
2. J. A. Kupsch, B. P. Miller, V. Basupalli and J. Burger, "From continuous integration to continuous assurance," *2017 IEEE 28th Annual Software Technology Conference (STC)*, Gaithersburg, MD, 2017, pp. 1-8, DOI: 10.1109/STC.2017.8234450.
3. Gravley, Lucas, “Introducing GitHub Super Linter: one linter to rule them all,” GitHub, URL: https://github.blog/2020-06-18-introducing-github-super-linter-one-linter-to-rule-them-all/. Accessed June, 24, 2020.

## APPENDIX A - Publications and Presentations

All publications are in PDF format and are not able to be inserted into the Word document version of this report. Publications can be provided in PDF format upon request.

### Presentations

2019

- Bart Miller and Elisa Heymann taught their tutorial, "Securing Coding Practices & Automated Assessment Tools," at the Internet2 2019 Technology Exchange in New Orleans, LA on December 9, 2019.
- Bart Miller and Elisa Heymann taught a tutorial, "Secure Coding Practices and Automated Assessment Tools," at Supercomputing 2019 in Denver, CO on November 17, 2019.
- Bart Miller and Elisa Heymann taught a tutorial, "Web Security and Automated Assessment Tools – Theory & Practice," at the NSF Cybersecurity Summit in San Diego, CA on October 15, 2019.
- Bart Miller and Elisa Heymann taught a tutorial for the CyberCorps SFSCon 2019 at California State Polytechnic University in Pomona, CA on September 27, 2019.
- Bart Miller and Elisa Heymann taught a tutorial, "Secure Coding Practices and Automated Assessment Tools," at the Gateways 2019 conference in San Diego, CA on September 23, 2019.
- Bart Miller and Elisa Heymann taught a secure coding tutorial, "Think Like an Attacker!," at The University of Iowa on September 17, 2019.
- Barton Miller gave a Jones Seminar titled "Why Johnny and Janie Can't Code Safely—Bringing Software Assurance to the Masses" at Dartmouth's Thayer School of Engineering on May 17, 2019.
- Miron Livny presented "Software Assurance Marketplace: Continuous Assurance for Critical Infrastructure Software" at the Department of Homeland Security's 2019 S&T Cybersecurity and Innovation Showcase on March 18, 2019 in Washington, D.C. The SWAMP team also provided demonstrations and discussions about SWAMP software at the Technical Demonstration portion of the Showcase.

2018

- Von Welch presented at Software Assurance Conference (SwACon) 2018 at the NRECA Conference Center in Arlington, VA on November 27, 2018.
- Bart Miller and Elisa Heymann gave a tutorial, "Secure Coding Practices and Automated Assessment Tools," at Supercomputing 2018 in Dallas, TX on November 11, 2018.
- Bart Miller and Elisa Heymann presented the session "Critical Infrastructure Software Security: A Maritime Shipping Study Case" at the O'Reilly Velocity conference in London on November 2, 2018.

- Bart Miller and Elisa Heymann taught a tutorial, "Secure Coding Practices, Automated Assessment Tools and the SWAMP" (Parts I and II), at the IEEE Cybersecurity Development Conference (SecDev) in Cambridge, MA on September 30, 2018.
- Bart Miller and Elisa Heymann from the University of Wisconsin-Madison taught a training session, "Automated Assessment Tools – Theory & Practice," at the 2018 NSF Cybersecurity Summit for Large Facilities and Cyberinfrastructure on August 21, 2018 in Alexandria, VA.
- SWAMP attended BlackHat USA 2018 on August 8-9, 2018 in Las Vegas, NV. The SWAMP gave 3 software demos in the DHS Science & Technology booth.
- SWAMP exhibited at DevOpsDays DC on June 6-7, 2018 in Alexandria, VA.
    - Miron Livny gave an Ignite talk, "Bringing Continuous Assurance to the Developer's Finger Tips," on June 7, 2018. To view the video, https://youtu.be/hYiKtlykLpE
- Bart Miller presented "Bringing Continuous Assurance to Your Code with the SWAMP" at UW-Madison's IT Professionals Conference 2018 on May 31, 2018.
- The SWAMP supported HACK NYC 2018 on May 8-10, 2018. Miron Livny gave a talk on May 8, "Bringing Continuous Assurance to the Software Developer."
- SWAMP supported Women in CyberSecurity at the WiCyS 2018 conference and career fair in Chicago, IL on March 23-24, 2018.
- The SWAMP was a Gold Sponsor for DevOpsDays Baltimore in Baltimore, MD on March 22, 2018.
- Bart Miller presented about software assurance to the US military European Command (EUCOM) in Germany on March 22-23, 2018 at the 2018 International Cyber Summit.
- Bart Miller gave a secure programming and tools tutorial at the Leibniz Supercomputer Center in Germany on March 21, 2018.
- The SWAMP and Parasoft teams hosted a webinar on March 8, 2018 about Parasoft tools in SWAMP-in-a-Box. A video recording is available https://alm.parasoft.com/supercharge-application-security-with-static-analysis-for-swamp.

2017

- Bart Miller taught the tutorial, "Secure Coding Practices and Automated Assessment Tools," at the O'Reilly Security Conference in New York City on October 30, 2017.
- Miron Livny presented at the Wisconsin State Employee Trust Fund IT Director Council meeting on October 25, 2017.
- The SWAMP team from Indiana University presented "SWAMP: Software assurance at IU" at the 2017 Statewide IT Conference at IU Bloomington on October 20, 2017.
- The SWAMP presented a paper, "From Continuous Integration to Continuous Assurance," at the 2017 IEEE Software Technology Conference (STC) at the NIST Headquarters in Gaithersburg, MD on September 26, 2017.

- The SWAMP staff taught a tutorial, "Automated Assessment Tools and the Software Assurance Marketplace (SWAMP)," at IEEE SecDev 2017 in Cambridge, MA on September 24, 2017.
- The SWAMP team presented a webinar for the CSIAC (Cyber Security & Information Systems Information Analysis Center) on September 12, 2017 titled "Overview of the Software Assurance Marketplace (SWAMP) & SWAMP-in-a-Box (SiB)".
- SWAMP was included in the Keynote by Kevin Greene, "Innovative R&D – A True Enabler For Achieving Security Testing 'At-Speed'", as part of DevSecCon Boston on September 11, 2017.
- Bart Miller taught an Afternoon Training Session called "Automated Assessment Tools – Theory & Practice" at the 2017 NSF Cybersecurity Summit for Large Facilities and Cyberinfrastructure in Arlington, VA on August 15, 2017.
- The SWAMP project presented at the 2017 Cyber Security R&D Showcase and Technical Workshop in D.C. on July 12, 2017.
    - Slides are available for review
- Bart Miller taught a 3-day course on secure programming and using the SWAMP at the FAA technology center on June 6-8, 2017.
- The SWAMP team gave a demo at the Madison, Wisconsin OWASP Chapter meeting on May 31, 2017.
- The SWAMP exhibited at OSCON 2017 in Austin, TX on May 10-11, 2017.
    - Bart Miller and Elisa Heymann taught a tutorial called "Secure coding practices and automated assessment tools" on Monday, May 8, 2017.
    - Susan Sons presented two sessions:
        - "Rebuilding a plane in flight: Refactors under pressure" on Tuesday, May 9, 2017.
        - "Finding your way in the dark: Security from first principles" on Wednesday, May 10, 2017.
- Jim Basney presented "CTSC+SWAMP: Cybersecurity Resources for Your Campus" at the CASC Spring 2017 Meeting in Alexandria, VA on March 24, 2017.
- Rob Quick gave an IUPTI Seminar, Software assurance at IU: A journey into the SWAMP, on March 1, 2017.

2016

- The SWAMP Project gave a webinar presentation to the DOE's Critical Infrastructure Supply Chain Working Group on December 13, 2016.
- SWAMP staff attended and presented at the Digital Infrastructures for Research 2016 in Krakow, Poland, September 28-30, 2016.
    - Slides from the 3rd WISE Workshop – Wise Information Security for collaborating E-infrastructures on September 27, 2016 are available for review.
- The SWAMP team hosted a cybersecurity station for children at the August 6, 2016 Saturday Science in the Discovery Building.

- Miron Livny presented to the General Dynamics Information Technology (GDIT) Software Technology Summit on July 20, 2016 in Falls Church, VA.
- SWAMP gave 2 presentations at the NIST Workshop on Software Measures and Metrics to Reduce Security Vulnerabilities on July 12, 2016.
    - Jim Kupsch presented "Dealing with Code that is Opaque to Static Analysis".
    - Miron Livny presented "Measure Early and Measure Often – SWAMP".
- Bart Miller presented "Software Assurance Issues and Threats in Maritime Shipping Systems" at Port of Valencia, Spain on March 2, 2016.
- SWAMP presented at the Cyber Security Division's 2016 R&D Showcase and Technical Workshop in D.C. on February 19, 2016.
    - Slides are available for review
- SWAMP demonstrated the alpha version of SWAMP-in-a-Box for the DHS S&T, CyberSecurity Division's 2016 R&D Showcase and Technical Workshop in D.C. on February 17, 2016.

2015

- SWAMP presented at the 2015 International ImageJ User and Developer Conference on September 4, 2015, at the Discovery Building in Madison, WI.
    - Abstract and slides are available for review.
- SWAMP attended the 2015 RSA Conference in San Francisco, CA, April 20-24, 2015, as part of the DHS S&T booth.
    - Two demos of Code Dx and the SWAMP were given on Wednesday, April 22, 2015.
- The SWAMP presented remotely to the OWASP-Montreal chapter at their meeting on February 3, 2015.
    - Presentation video is available for review.

2014

- The SWAMP presented at the DHS S&T R&D Showcase and Technical Workshop on December 18, 2014.
- The SWAMP Project hosted a panel discussion with our partners on September 18, 2014 entitled "Riddle Me This, Batman: DHS, Open Source, and the SWAMP."
        - Video of the panel is available for review
- Von Welch, Chief Security Officer, spoke at Circle City Con 2014 on June 15, 2104.
    - Slides/video are available for review.
- SWAMP and T.E.N. hosted a webinar with a virtual tour of the SWAMP on May 22, 2014, entitled "Your Guide Through the SWAMP: Avoiding Predators in a Murky World."
    - Slides from the presentation are available to review or to watch the video recording.

- SWAMP hosted a Principal Investigators Meeting at the Morgridge Institute for Research in Madison, WI, May 12-14, 2014.
    - Click here for slides from the sessions.
- Barton Miller, Chief Scientist, spoke at the Second Shonan Meeting on Grid and Cloud Security in Japan, March 24-27, 2014.
- Chief Security Officer, Von Welch, spoke at the Bloomington OWASP chapter on January 28, 2104. Slides are available for review
- SWAMP and T.E.N. hosted two virtual town hall meetings on January 22-23, 2014
    - Developer meeting, "Good Security Starts with Software Assurance," on January 23, 2014. Audio and slides are available for review.
    - Townhall Executive meeting, "Shaping Your Approach: The Executive's Role in Software Assurance," on January 22, 2014. Audio and slides are available for review

2013

- Barton Mill, SWAMP's Chief Scientist presented on "Secure Coding Practices" at the NSF Security Summit on September 30, 2013 in Arlington, VA
- SWAMP PI's attended and present at the DHS Principal Investigators Meeting and Software Assurance Forum on September 16-19, 2013 in Crystal City, VA
- Birds of a Feather presentation at the USENIX Security Symposium on August 14, 2013 in DC
- About SWAMP presentation at CSET 13 on August 12, 2013 in DC
- Miron Livny spoke on Cybersecurity 101 at the Wisconsin Innovation Network Luncheon on February 26, 2013
- SWAMP Unified Overview, https://www.swampinabox.org/doc/SWAMP-Unified-Overview-130206.pdf
- ITTC California presentation, https://www.swampinabox.org/doc/SWAMP-Overview-ITTC-CA-130207.pdf

2012

- Panel discussions at the DHS SWA Workshop on November 27-29, 2012 in Washington D.C.

**Publications**

"SWAMP Capabilities," SWAMP whitepaper 01, December 2013.

https://www.swampinabox.org/doc/SWAMP-WP001-Capabilities.pdf

Von Welch, "SWAMP Confidentiality and Privacy for User Data," SWAMP whitepaper 04, June 2014.
https://www.swampinabox.org/doc/SWAMP-WP004-Privacy.pdf

"SWAMP Privacy Policy, January 2014.
https://www.swampinabox.org/doc/SWAMP-Privacy-Policy.pdf

SWAMP Frequently Asked Questions (FAQ),
https://www.swampinabox.org/doc/SWAMP-FAQ.pdf

James A. Kupsch and Barton P. Miller, "Why Do Software Assurance Tools Have Problems Finding Bugs Like Heartbleed?", SWAMP Technical Report WP003, April 2014.
https://www.swampinabox.org/doc/SWAMP-WP003-Heartbleed.pdf

James A. Kupsch, Barton P. Miller, Vamshi Basupalli, and Josef Burger, "From Continuous Integration to Continuous Assurance", IEEE Software Technology Conference, Gaithersburg, Maryland, September 2017.
https://www.swampinabox.org/doc/SWAMP-WP005-DevProcess.pdf

James Basney, Irene Landrum, Miron Livny, Bart Miller, Von Welch, "A Vision for the Software Assurance Marketplace: A Transformative Force in the Software Assurance Ecosystem", Version 2.0, December 2017.
https://www.swampinabox.org/doc/SWAMP-VISION-12.22.17.pdf

James A. Kupsch, Elisa Heymann, Barton P. Miller and Vamshi Basupalli, "Bad and Good News about Using Software Assurance Tools", Software: Practice and Experience, 2016. DOI: 10.1002/spe.2401.
ftp://ftp.cs.wisc.edu/paradyn/papers/Kupsch_et_al-2016-Software__Practice_and_Experience.pdf

Josef Burger, "Heartbit -- an abstract of the openssl library, illustrating that tools cannot locate the actual Heartbleed flaw, even when the complexity of openssl is reduced to bare essentials", April 2014
https://www.cs.wisc.edu/mist/heartbit/

Miron Livny and Barton P. Miller, "The Case for an Open and Evolving Software Assurance Framework", SWAMP Technical Report WP002, November 2013.
https://www.swampinabox.org/doc/SWAMP-WP002-Framework.pdf

James A. Kupsch, "Status.out and Debugging SWAMP Failures," January 2017.

**Press Releases**

National Cybersecurity Effort Launched to Strengthen Software Infrastructure (November 1, 2012)
http://www.news.wisc.edu/21224

Software Assurance Marketplace to Host Exposition (May 23, 2013)
http://news.wisc.edu/software-assurance-marketplace-to-host-exposition/

UW-Madison Poised to Become National Hub for Software Security (October 14, 2013)
https://continuousassurance.org/blog/2013/10/15/press-release-swamp-poised-to-be-national-hub-for-software-assurance/

National, Shared Software Assurance Facility, 'SWAMP,' Launches (February 3, 2014)
http://news.wisc.edu/national-shared-software-assurance-facility-swamp-launches/

SWAMP and Secure Decisions Partner to Enhance Software Security (July 14, 2014)
http://www.prweb.com/releases/2014/07/prweb12014632.htm

Gaining Industry Momentum with Improved Interface Enhancements (July 22, 2014)
http://www.prweb.com/releases/2014/07/prweb12036108.htm

SWAMP Announces Expert Panel on Software Assurance at OWASP's AppSec USA 2014 Show (September 17, 2014)
http://www.prweb.com/releases/2014/09/prweb12176972.htm

SWAMP Announces Partnerships with Commercial Vendors (September 19, 2014)
http://www.prweb.com/releases/2014/09/prweb12181918.htm

SWAMP Wins ISE North America Project of the Year Award (November 06, 2014)
http://www.prweb.com/releases/2014/11/prweb12342084.htm

Year One: SWAMP a Catalyst for Improving Cyber-Security (February 13, 2015)
https://morgridge.org/story/year-one-swamp-a-catalyst-for-improving-cyber-security/

SWAMP Increases Scope with Python Functionality and Security Scanning of Code Snippets (March 3, 2015)

http://www.prweb.com/releases/2015/03/prweb12553109.htm

SWAMP Expands Portfolio of Open-Access Software Security Tools (July 20, 2015)
http://globenewswire.com/news-release/2015/07/20/753489/10142286/en/Software-Assurance-Marketplace-Expands-Portfolio-of-Open-Access-Software-Security-Tools.html

SWAMP Continues to Expand Platform of Software Assurance Resources (September 24, 2015)

http://globenewswire.com/news-release/2015/09/24/770858/10150552/en/SWAMP-continues-to-expand-platform-of-software-assurance-resources.html

Can cybersecurity crack the undergraduate curriculum? (May 16, 2016)
https://morgridge.org/newsarticle/can-cybersecurity-crack-undergraduate-curriculum/

'SWAMP-in-a-Box,' an on-premises continuous software assurance capability (October 13, 2016)

http://globenewswire.com/news-release/2016/10/13/879300/10165597/en/SWAMP-in-a-Box-an-on-premises-continuous-software-assurance-capability.html

SWAMP integrates assurance tools into the software continuous lifecycle (May 2, 2017)
http://www.marketwired.com/press-release/swamp-integrates-assurance-tools-into-the-software-continuous-lifecycle-2213473.htm

Madison-based SWAMP and Synopsys join forces to educate the future cybersecurity workforce (December 21, 2017)

http://www.globenewswire.com/news-release/2017/12/21/1269116/0/en/Madison-based-SWAMP-and-Synopsys-join-forces-to-educate-the-future-cybersecurity-workforce.html

With SWAMP-in-a-Box, 'Bring Your Own License' and turbo-charge software assurance (February 28, 2018)

https://globenewswire.com/news-release/2018/02/28/1401131/0/en/With-SWAMP-in-a-Box-Bring-Your-Own-License-and-turbo-charge-software-assurance.html

Want to fight cyberthreats? Start with clean code (August 8, 2018)
https://morgridge.org/story/want-to-fight-cyberthreats/

GrammaTech Adds Real World Benchmarks to SWAMP (January 31, 2019)
https://www.prnewswire.com/news-releases/grammatech-adds-real-world-benchmarks-to-swamp-300787603.html

## APPENDIX B - Software Downloads

**Assessment Frameworks** orchestrate setting up an environment to run an assessment, and then the process of performing an assessment of a software package using a static analysis tool to produce a SCARF output file. It accomplishes this by monitoring the build of the software and using this data to drive the process. There are four assessment frameworks based on the programming language to be assessed: C/C++, Java, Ruby and other languages (Script).
https://github.com/mirswamp/c-assess
https://github.com/mirswamp/java-assess
https://github.com/mirswamp/ruby-assess
https://github.com/mirswamp/script-assess

**Result Parser** is the code that converts the native tool output into the SCARF output format (and/or SARIF format).
https://github.com/mirswamp/resultparser

**SCARF library** contains functions to both read and write SCARF files for code written in C, Java, Perl or Python.
https://github.com/mirswamp/swamp-scarf-io

**SARIF library** contains functions to write a SARIF for Perl users.
https://github.com/mirswamp/swamp-sarif-io

**SCARF TO SARIF converter** is a program to convert existing SCARF output files to the SARIF format.
https://github.com/mirswamp/swamp-scarf-sarif

**SWAMP Java API and SWAMP CLI** is a package that contains a Java library and a command line interface that can programmatically perform high-level tasks on a SWAMP instance. These include the uploading for software packages, starting an assessment, monitoring its progress and retrieving SCARF output files.
https://github.com/mirswamp/java-cli

**SCARF Diff** is a program to compare the contents of two SCARF files.
https://github.com/mirswamp/swamp-scarf-diff

**Eclipse Plug-in** is the code for an Eclipse IDE plug-in that allows C, C++ or Java code being developer in Eclipse to use the SWAMP for its assessment functionalityand to view the results within Eclipse.
https://github.com/mirswamp/swamp-eclipse-plugin

**Jenkins Plug-in** is the code for a Jenkins Continuous Integration (CI) system to use the SWAMP for its assessment functionality and to view the results using Jenkins dashboard.
https://github.com/mirswamp/swamp-jenkins-plugin
https://github.com/jenkinsci/swamp-plugin

**Git and Subversion SCMS Plug-in** is the code for a program that can be used as a plugin the Source Code Management Systems git and subversion. The plug-in supports trigging an upload and assessment in the SWAMP of commits or merges to the SCMS repository.
https://github.com/mirswamp/swamp-scms-plugin

## LISTS OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

2FA: two-factor authentication

API: application programming interface

APK: Android application package

AWS: Amazon Web Services

BAA: Broad Agency Announcement

CI: continuous integration

CISO: Chief Information Security Officer

CLI: command line interface

COTS: commercial-off-the-shelf

CPU: central processing unit

CSS: Cascading Style Sheets

DARPA: Defense Advanced Research Projects Agency

DHS: Department of Homeland Security

DHS S&T: Department of Homeland Security – Science & Technology Directorate

DNS: domain name system

EC2: elastic compute cloud (part of AWS offerings)

EMC: not an acronym, DELL EMC, formerly EMC Corporation

GCE: Google compute engine

IAM: identity and access management

I/O: input/ouput

IDE: integrated development environment

IDS/IPS: Intrusion Detection and Prevention System

IOC: initial operating capabilities

IP: internet protocol

IUPUI: Indiana University Purdue University Indianapolis

JIRA: not an acronym, software by Atlassian

JSON: JavaScript object notation

LDAP: lightweight directory access protocol

NCSA: National Center for Supercomputing Applications

NIST: National Institute for Standards and Technology

NSF: National Science Foundation

NTP: network time protocol

OASIS: Organization for the Advancement of Structured Information Standards

OpenSSL: not acronym, open source full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols

OS: operating system

OWASP: Open Web Application Security Project

PHP: Programming language; recursive acronym for Hypertext Preprocessor

PHPMD: PHP Mess Detector

PI: Principle Investigator

PMD: Programming Mistake Detector

PRQA: Programming Research – Quality Assurance

R&D: Research & Development

RAM: Random Access Memory

RHVM: Red Hat Virtualization Manager

SARIF: Static Analysis Results Interchange Format

SATE: Static Analysis Tool Exposition

SCA: static code analysis

SCARF: SWAMP Common Assessment Results Format

SCATE: Static Code Analysis Tool Evaluator

SCMS: source code management system

SDLC: software development lifecycle

SiB: SWAMP-in-a-Box

SQL: Structured Query Language

STAMP: Static Tool Analysis Modernization Project

SWA: software assurance

SWAMP: Software Assurance Marketplace

TIF: Tunable Information Flow; specific to HRL Laboratories, LLC static analysis tool

UCS: unified computing system

UI: user interface

UW: University of Wisconsin

VLAN: virtual local area network

VM: virtual machine

VNX: virtual network exchange

XML: Extensible Markup Language