Finer-grained access controls will be added after IOC. Additionally, integrity provisions protect against accidental or malicious changes.

The following section highlights specific Software Developer activities enabled by the SWAMP.

SWAMP Software Assurance Tool Developer Use Case Diagram

Support for the tool developer at IOC

We expect tool developers to be able to perform the following common activities:

● Manage Membership: ○ Apply for, receive, and manage membership in the SWAMP.

- Manage Tools: ○ Add, modify or remove a SwA tool,

versions of the tool, or access permissions. ○ Access the host environment used to build the software to interactively debug the software package build in the case of failure.

- Test Tools:

  ○ Assess one or more software packages (or test suites) using one or more SwA tool on one or more operating system platforms. The tool developer can also import their own software package for testing with the same mechanism used to bring packages into the SWAMP. ○ Schedule one-time, or recurring assessments to support CSwA. ○ Monitor the status of pending, ongoing, or completed assessments via a dashboard of assessment activities. Summary information of successfully completed assessments such as number of weaknesses found are displayed in the dashboard. ○ Modify or cancel recurring or future assessments.

- Analyze Results:

  ○ Retrieve or view raw tool output from an assessment.

  ○ Retrieve or view tool output after minimally parsing the raw tool output into a set of weaknesses. Each weakness is identified as to its location (file and line number) and presented with a textual description of the weakness as provided by the tool. The textual description is used by a human analyst, and for comparison to other textual descriptions. As there is no standard tool output format, comparing textual descriptions from different tools has little meaning.

- Sort by location of weakness to group by function and file.
- Sort by textual description to group by type of weakness

- Combine results from multiple assessments of the same package including the above sorting options.

- Results from different versions of the same tool allows a tool developer to detect changes in results based on location and type of weakness (identical opaque textual descriptions indicate the same type of weakness).

- Results from multiple tools allows a tool developer to determine how the coverage of each tool's reported weaknesses differ by sorting by location.

- View metrics about tool performance from an assessment such as the time to perform the assessment and the number of weaknesses discovered.

- A basic database stores assessment results and provides simple queries to support the analyses described above.

- Assessment results includes labeling information, such as tool name and version, package name and version, platform description, date and time, filename, line number, basic metrics, and textual description of the weakness.

- Assessment results are controlled by the user.

Detailed Narrative of Use Case

Manage Membership

Before using the SWAMP, each user must register an account. Any personal information required is kept in strict confidence and not shared with any other SWAMP user. For access to SWAMP capabilities the user's account must then be associated with an active SWAMP project. The user may request the creation of a new SWAMP project or with the permission of the project owner

join an existing SWAMP project. Each project request is reviewed by SWAMP administration to ensure the requested use is supported by and aligned with the SWAMPs capabilities and mission. Once access is granted, one can use the account management interface to update personal information and change passwords as needed. Projects and users may disassociate with each other at any time. When a user no longer needs access to the SWAMP, they may cancel their account.

Manage Tools

The first step in making an SwA tool available to perform assessments is to register the tool with the SWAMP. Registering a tool involves getting the tool into the SWAMP and configured for operation. The tool software can be uploaded directly via the web interface or pulled from external hosts such as a web server or source code repository. Multiple versions of the same tool can registered and the tool software can be loaded as a binary or as source code which can then be built on demand in the SWAMP. Registration also includes defining various parameters about the tool, such as the version, description, documentation, the language the tool can assess (e.g. C, C++, Java) and on which platforms it will execute (e.g. Linux, Mac, Windows).

The next step is to configure the tool. Tool configuration includes setting access controls to determine which, if any, other SWAMP users can use the tool and identifying the parameters that may be adjusted for a given assessment. At IOC, we expect access control to be based on the project. By default, members of the project are able to access the package(s) and assessment results, and other are prohibited. In the future, we plan to add more granular, group-based access controls, for example private to one user, private to project members or available to all SWAMP users. SWAMP staff reserves the right to disable a tool made available for public use if it functions incorrectly or causes performance issues for other users. The tool owner may disable a tool from public use or remove it completely from the SWAMP. After IOC, the tool developer may be provided with aggregate

usage statistics for their tool such as number of unique tool users, number and size of assessments performed and number of failures (crashes) of the tool, but no identifying user information is included.

Test Tools

Besides making their tool available for other SWAMP users, the main task that tool developers perform in the SWAMP is testing their tool. Basic tool testing involves using an installed tool to perform an assessment of a software package or test suite, and analyzing the results. To simplify creating test cases, the tool developer can select multiple tools, software packages and platforms to test against, and the SWAMP generates tests for all possible combinations. In addition to the corpus of source code from test suites and open source packages, the tool developer can upload their own packages for testing.

The tool developer is able to monitor the status of upcoming, ongoing and completed assessments using a web-based dashboard. Summary information of successfully completed assessments such as number of weaknesses found are displayed in the dashboard. The dashboard enables the developer to schedule recurring assessments thereby enabling CSwA. The tool developer can also modify and cancel future assessments. When an assessment completes, a notification can be sent.

Analyze Results

The SWAMP provides access to the raw build and assessment output. If the build of a software package fails, the software developer is able to debug the failure using the output files, or through interactive access to the host environment where the build failed. If the assessments complete successfully, the results are presented in a results browser. Results from an assessment are, by default, available to all members of the project. Results are maintained in the SWAMP until the user deems them unnecessary or under conditions described in the

SWAMP's User Data Retention Policy.

At IOC, we expect that tool output will be parsed to determine the location of the weakness in the source code and description of the weakness. With this functionality, the tool developer is able to view the raw tool output, view a list of weaknesses and locations, and compare the results of multiple assessments with the same tool (and output format) on the same source code. In the future we expect to more fully decode the tool output to be able to normalize the weakness types based on CWE, which allows output comparisons between different tools.

At IOC, we expect to provide a rudimentary source code viewer to display the source code of the weakness location by clicking the weakness in a web browser. In the future we expect to provide a more full featured code browser with hyperlinking capabilities to find name declaration, definitions, and use locations, along with type information, and other information of use to a developer such as call graphs.

An important feature for analyzing tool output is the ability to triage the results. Shortly after IOC, we will support basic triage capabilities to mark a weakness discovered in a package as a false positive, confirmed or unknown. We will apply these triage results to future assessments of the same software package to remove false positives from display. In the future, we will enable a richer set of information to be entered for each weakness, such as notes, assignment of reviewer, and assignment of repair.

Metrics are collected about the assessment, such as duration and number of weaknesses. In the future, we will collect additional metrics as deemed useful, and also collect metrics about the assessed software package such as the source code size and complexity. This allows data analysis using the collected metrics and results.

Future support for the tool developer

We plan to add the following capabilities in years 2-3 of the

project:

- Manage Tools:
    - Improve the dashboard to include features such as trend lines from previous assessments, so the SwA tool developer can tell at a glance if new weaknesses are introduced, or old weaknesses are corrected.
    - Allow a tool developer to specify configuration parameters that allow tuning of the results or otherwise change the behavior of the tool.
    - Add finer-grained access controls to specify what operations a user can perform on a specific object in the system. Users will be able to grant permissions to specific users or groups of users such as member of projects or other groups. The operations include viewing assessment results, using a SwA tool, performing an assessment, accessing the software of a project, and managing users. The objects in the SWAMP include users, projects, results, summary results, software packages and SwA tools.
- Test Tools:

○ Allow a tool developer to add tool configuration parameters as another dimension when

configuring a set of assessments in addition to the tools, packages and platforms. ● Analyze results

○ Retrieve or view tool output of the fully parsed raw tool output. This is a set of weaknesses with the minimally parsed attributes (location and raw weakness description), and a normalized version of the weakness such as a CWE or a set of CWEs based on the raw weakness description. This allows sorting by CWE, and meaningful comparison between different tools such as:

■ Differences in weakness discovery between tools ● Unique

discoveries

- Missed discoveries
- False positives (using source with triage data)

■ Differences in weakness types discovered between tools

○ Triage support for each weakness discovered. This is human entered data that is useful for future viewings of these same results, or with other assessments of the same package. The types of triage information includes: ■ Status such as *confirmed* (will fix), *ignore* (confirmed but will not fix), *false positive,* and *unknown.* ■ Display filtering: by default, *confirmed* and *unknown* are displayed, *ignore* and *false positive* are hidden, but can be overridden. ■ Free form descriptive text of problem or fix. ■ Developer assigned to fix.

○ Filter results based on triage information display filtering field. This allows a tool developer to focus on new weaknesses deemed important, or new weaknesses.

○ View results of performing an assessment of the corpus of source code in the SWAMP's preloaded test suite, applying triage information of known weaknesses and known false positives. This allows a tool developer to automatically determine which known weaknesses they find, which known false positives they detect, and which undiscovered (or unclassified) weaknesses they detect.

○ Use a source code viewer to assist in the triage effort. The web-based source code viewer will have the ability to browse the source code and to visually see the locations where weaknesses were discovered. It will also assist the developer in tasks such as finding definitions, uses, and calls; see type information; jump to related source files such as include files; and view the call graph.

○ Capture and display metrics of source code assessed.

These are common code metrics such as lines of code (LOC), and various complexity measures.

- View tabular data, or graphs comparing multiple assessments. For instance a SwA tool developer might be interested in the assessment duration versus the tool version of the tool assessing the same package or the assessment duration versus the LOC of each package with the same tool assessing each package.

- A database will store assessment results and provide queries to support the analyses described above.

- The pre-IOC database will be extended to include both the textual descriptions of the weaknesses and the fully parsed and labeled version of the weaknesses.

- The database will include information to support the triage and filtering of results on a per-package level.

- The database will include a description of tool configuration parameters and detailed metrics of executing the tool while performing an assessment.