**SWAMP Software Developer Use Case Diagram**

## Support for the Software Developer at IOC

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

- Manage Membership:
  - Apply for, receive, and manage membership in the SWAMP.
- Manage Software Packages:
  - Add, modify or remove a software package, versions of the software package, or access permissions.
  - Access the host environment used to build the software to interactively debug the software package build in the case of failure.

- Assess Software Packages:
  - Assess one or more versions of a software package using one or more SwA tools on one or more operating system platforms.
  - Schedule one-time, or recurring assessments to support CSwA.
  - Monitor the status of upcoming, 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 can be 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 SwA tools assessing the same package including the above sorting options.

- Number and types of weaknesses discovered increases due to differences in weakness detection techniques.

- The same defect discovered by multiple tools, may improve the confidence in the discovery, and only requires a single inspection by an analyst to assess its correctness.

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

- Assessment results include 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.

- Access to assessment results are controlled by the user.

## Detailed Narrative of Use Case

### Manage Membership

Before using the SWAMP, each user must register for 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 SWAMP's 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 themselves from each other at any time.

When a user no longer needs access to the SWAMP, they may cancel their account.

Manage Software Packages

A project is a group of related developers. A project may have multiple software packages which they wish to assess, so the SWAMP supports multiple packages per project. The first step in making a software package available to be assessed is to create the package within the project. The package acts as the umbrella under which different versions of the software package are placed. Registering a software package involves describing how to get the software into the SWAMP along with information required to build the software, and other details about the version, such as the version string, description, documentation, programming languages, and platforms. The software package can be uploaded directly via the web interface, or pulled from external hosts such as a web server or source code repository.

The software package configuration includes access controls that determine which, if any, other SWAMP users can access information about a project, package or package version. 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 finer-grained access controls, for example private to specific users, private to project members or available to all SWAMP users. After IOC, we may report to tool providers the aggregate usage statistics of their tool, but no identifying information of the tool users will be included.

**Assess Software Packages**

The main task that software developers perform in the SWAMP is to assess their software package with a SwA tool. Once a software package builds in the SWAMP, the next step is to select the combination of SwA tools and platforms to perform the assessments. Reasonable default tool and platform selections

are provided based on known characteristics of the software package. The software developer is able to start the assessment immediately, or to schedule assessments as a one time or recurring task. The SWAMP infrastructure automatically builds and invokes the selected tools on the selected platforms to perform the assessments.

The software 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 software developer are also able to 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, are 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 document.

At IOC, we expect that the output of the tools will be parsed to determine the location of the weakness in the source code and description of the weakness. With this functionality, the software developer is able to view the raw tool output, view a list of weaknesses and locations, and view the combined results of different tools 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 (Common Weakness

Enumeration, http://cwe.mitre.org), 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. The triage results can be used to filter 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 Software Developer

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

- Manage Software Packages:
    - Improve the dashboard to include features such as trend lines from previous assessments, so the software developer can tell at a glance if new weaknesses are introduced, or old weaknesses are

corrected.

- ○ Allow a software developer to specify the build configuration parameters. Most software packages allow build configuration parameters to control the features present in the software, change implementation details, or compiler settings.

- ○ 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.

- Assess Software Packages:

  - ○ Allow a software developer to add build configuration parameters as another dimension

when configuring a set of assessments, in addition to the tools, packages and platforms. This will allow a comparison of how the build configuration affects the weaknesse discovered.

- 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 include: ■ 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. This allows a software developer to focus on previous weaknesses deemed important, or new weaknesses.

○ Compare assessment results from different versions of the package. This allows a software developer to easily determine those weaknesses that have been eliminated, and those that are newly discovered. This will require the automatic ability to determine how the code has changed between the two versions so the location of a weakness can be accurately mapped between the two versions.

○ 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 software developer might be interested in the types and numbers of weaknesses discovered versus the version of the package, or the number of weaknesses discovered in each source file versus the LOC of the source file.

- 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.