Security Oracle - User Manual

July 3, 2016

Contents

Part I
Security Oracle Service

1 Overview

Security Oracle is the main component of the Security Oracle ecosystem. It is a RESTful, Swagger-compatible API for determining if an application has any known vulnerabilities, and if so, it may propose solutions to these vulnerabilities and assist the user in evaluating and applying them. Version 1 of Security Oracle may only do so for NodeJS applications, and may only identify vulnerabilities if they had been reported on nodesecurity.io.

2 Usage

As of this writing, an instance of Security Oracle is deployed and publicly accessible at `https://security-oracle.eu-gb.mybluemix.net`. As Security Oracle is fully Swagger-compatible, it also includes an embedded documentation browser that also supports dispatching fully-formed requests to the server. You may view this system at `https://security-oracle.eu-gb.mybluemix.net/docs/`. While this section of the document will briefly outline the supported operations, the exact expected input and output formats and location of the endpoints is better detailed on the documentation browser.

2.1 The App Endpoint

The App endpoint is likely where an application will first interact with Security Oracle. It supports two operations - an HTTP POST to submit an app for review, ultimately reporting whether any vulnerabilities were detected and linking to a full report, and an HTTP GET to retrieve a list of all reports associated with an app. Besides some standard metadata describing the submitted application, the App endpoint more importantly expects a serialized version of the application's dependency tree. The exact structure of this serialized form is given more clearly in the Swagger docs. It is also explored and compared with the native formats exposed by common package management systems, in the peripheral report Extracting Dependency Tree from Common Package Managers.

2.2 The Report Endpoint

The Report endpoint is likely the second interaction an application will have with Security Oracle. Using a report ID returned by the the App endpoint, the user may then retrieve a more detailed report of the vulnerabilities found. This report will include the vulnerable package and its version, its path down the dependency tree, a description of the vulnerability, a hyperlink to the vulnerability's page on the vulnerability database and, if found, a proposed solution. A proposed solution will always be the name of a direct dependency of the submitted application and a version of that dependency that is no longer vulnerable. Security Oracle will never suggest updating dependencies further down the tree, as updates of this nature may cause unexpected changes in behavior, nor will it suggest a version that isn't compatible with the currently installed version or a version where the vulnerability isn't fixed.

2.3 The Solution Endpoint

The Solution endpoint records the user's decision regarding proposed solutions. While the two previous endpoints were technically sufficient to detect vulnerabilities and remediate them if possible, in real-world use, the user may choose to reject a proposed solution. The goal is to allow the user to perform this entire process using Security Oracle, therefore this endpoint exists to store the user's choice regarding proposed solutions and to allow retrieval of this information at a later time. Specifically, the information stored is a list of accepted solutions. Any solutions not listed are implicitly rejected and will not be applied.

3 Installation and Setup

You may also install and run your own instance of Security Oracle. Security Oracle is written in NodeJS and utilizes NPM packages, so you must have NodeJS and NPM installed. It has been most extensively tested using NodeJS version 4, but should have no issues running under versions 5 or 6. It has not been tested under NodeJS 0.12. While Security Oracle is a fully self-contained NodeJS application and NPM package, it is not currently published publicly on NPM. To install it locally, you must check the source code out of its GitHub-hosted git repository. A typical installation procedure may be:

```
$ git clone https://github.com/SecurityOracle/SecurityOracle.git
$ cd SecurityOracle
$ npm install
```

Security Oracle requires it's own local version of the vulnerability database maintained by nodesecurity.io. For your convenience, you may download this database and immediately start the service by running:

```
$ npm run dl-start
```

You may then start Security Oracle again without downloading the vulnerability database by running:

```
$ npm run start
```

Security Oracle is also compatible and will behave well in a Cloud Foundry environment. Assuming that you have all the relevant tools installed and configured, you may run:

```
$ cf push
```

to deploy Security Oracle to a Cloud Foundry environment. This may require a manifest.yml file or additional configuration.

Part II
Security Oracle NodeJS Buildpack

4 Overview

While the Security Oracle Service provides the needed information to identify and remediate security vulnerabilities, it does not immediately provide a way to use this information to easily improve the security of deployed applications. The Security Oracle NodeJS Buildpack is a drop-in replacement for the standard Cloud Foundry NodeJS buildpack, that integrates Security Oracle into the standard deployment workflow for Cloud Foundry NodeJS applications.

5 Usage

The Security Oracle NodeJS Buildpack is a standard Cloud Foundry-compatible buildpack for NodeJS. In fact, it was directly forked from the official NodeJS buildpack. As such, using it for your project is as simple as using any other buildpack. Assuming Cloud Foundry tools are installed and configured correctly on your machine, simply deploy your application by running

```
$ cf push -b https://github.com/SecurityOracle/nodejs-buildpack
```

to push your application using the Security Oracle NodeJS Buildpack.

5.1 Buildpack Output

In addition to the output that is to be expected from the standard NodeJS buildpack, Security Oracle NodeJS Buildpack will print two additional sections; Applying previous solutions and Querying Security Oracle. The former will not contain any additional information - lacking any errors, you may assume that any proposed solutions that have been explicitly accepted have been successfully applied. The latter will also contain two additional outputs; the vulnerability status and a link to the full report.

5.2 Method of Operation

Security Oracle NodeJS Buildpack is a copy of the standard Cloud Foundry NodeJS buildpack, with the addition of two new steps. First, before NPM dependencies are installed, the Buildpack queries the Security Oracle Service for any relevant, accepted solutions for the current application. If found, the Buildpack alters the application's package.json, recording the dependency version suggested by the Security Oracle Service. Second, after the installation of all dependencies, the Buildpack compiles the application metadata and dependency tree into the format expected by Security Oracle Service, sends this information to it and reports the results.

5.3 Limitations

As buildpacks run remotely, on the Cloud Foundry platform, any changes made by the Buildpack to the application's package.json will not be persisted to the user's local package.json file. Additionally, the Buildpack is essentially useless without a Security Oracle to query. In it's current version, the Security Oracle NodeJS Buildpack is hard-coded to refer to the Security Oracle hosted at security-oracle.eu-gb.mybluemix.net. If this instance of Security Oracle were to become unavailable, the Buildpack will have to be altered accordingly. Please refer to the Developer's Manual for more information about the Buildpack's implementation.

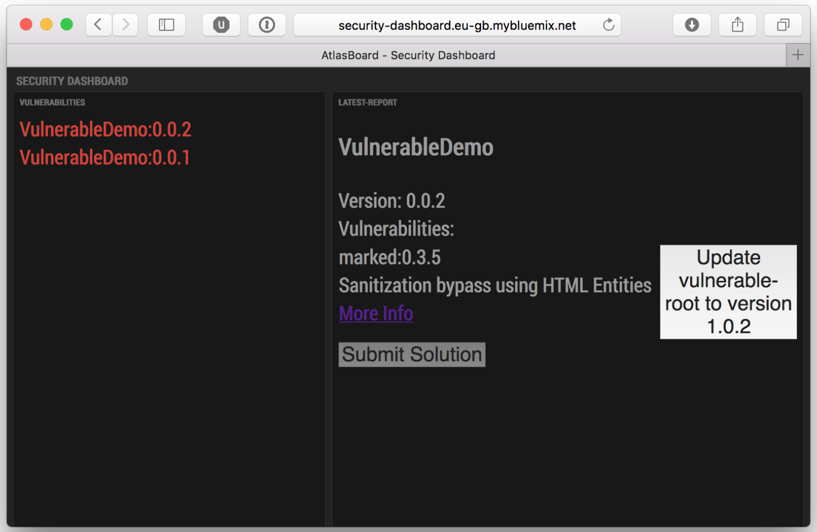
Part III
Security Dashboard

6 Overview

The combination of the Security Oracle Service and Buildpack provides all the required information for identifying vulnerabilities, suggesting solutions to them and applying them. However, they fail to provide a mechanism to continuously monitor the vulnerability status of an application, or even to accept or reject proposed solutions. Buildpacks are by nature not interactive - they cannot accept user input, they can only provide textual output and they may not run continuously in the background unless an application is being pushed to Cloud Foundry. Security Dashboard is a dashboard for Atlasboard, an open source framework for developing and running dashboards. It comes complete with the relevant jobs - background processes to continuously supply up-to-date information, and widgets - methods to transform the acquired information into meaningful visual representations for the user. Security Dashboard can be configured to monitor any application submitted to Security Oracle, identified by its unique application ID.

7 Usage

Security Dashboard has two panels. The left panel will list all deployed versions of the application, color coded by their vulnerability status. The right panel will always show the detailed vulnerability status of the latest deployed version of the app. If Security Oracle also has a proposed solution, you'll be presented with a big button to accept it. The button will turn green once accepted and you may then submit your accepted solution back to the Security Oracle Service. A typical workflow is shown in the figures below.

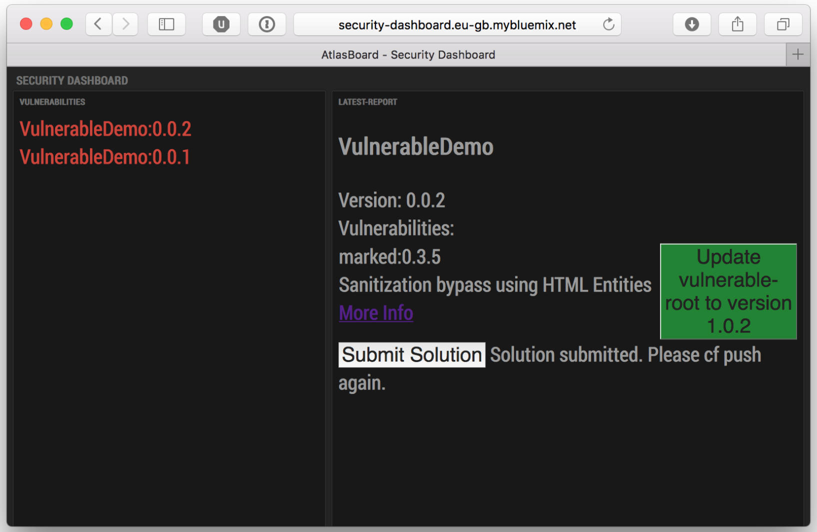

security-dashboard.eu-gb.mybluemix.net
AtlasBoard - Security Dashboard
SECURITY DASHBOARD
VULNERABILITIES
LATEST-REPORT
Update vulnerable-root to version 1.0.2
More Info
Submit Solution

VulnerableDemo:0.0.2
VulnerableDemo:0.0.1

VulnerableDemo

Version: 0.0.2
Vulnerabilities:
marked:0.3.5
Sanitization bypass using HTML Entities

Security Dashboard showing two vulnerable deployed versions of the application


security-dashboard.eu-gb.mybluemix.net
AtlasBoard - Security Dashboard
SECURITY DASHBOARD
VULNERABILITIES
LATEST-REPORT
Update vulnerable-root to version 1.0.2
More Info
Submit Solution

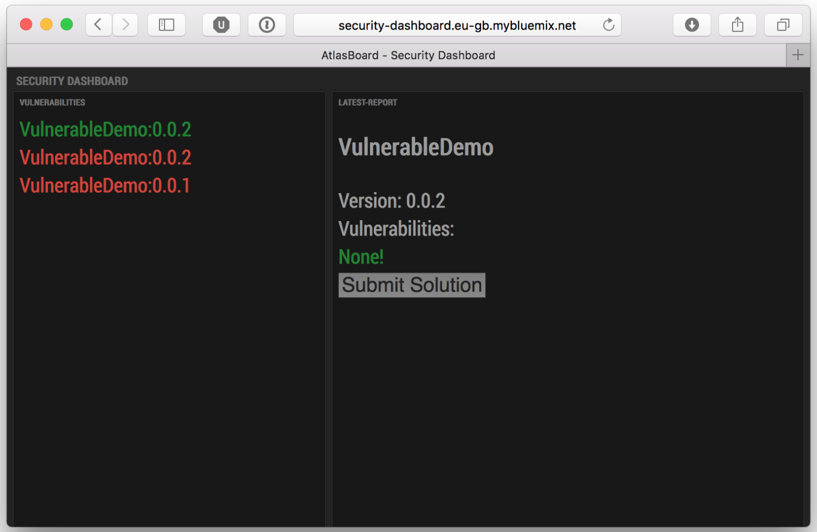
VulnerableDemo:0.0.2
VulnerableDemo:0.0.1

VulnerableDemo

Version: 0.0.2
Vulnerabilities:
marked:0.3.5
Sanitization bypass using HTML Entities

Solution submitted. Please cf push again.

Suggested solution accepted by clicking the "Update" button, and solutions submitted by clicking the "Submit Solution" button


security-dashboard.eu-gb.mybluemix.net
AtlasBoard - Security Dashboard
SECURITY DASHBOARD
VULNERABILITIES
LATEST-REPORT
Submit Solution

VulnerableDemo:0.0.2
VulnerableDemo:0.0.2
VulnerableDemo:0.0.1

VulnerableDemo

Version: 0.0.2
Vulnerabilities:
None!

Security Dashboard showing the latest version free of any detected vulnerabilities

8 Installation and Setup

Security Dashboard is a comprised of a standard Atlasboard dashboard, as well as a set of Atlasboard jobs and widgets. Thus, it can be easily installed into any existing Atlasboard instance. It is, however, distributed as a ready to run, standalone NodeJS application that already includes a default Atlasboard installation. Much like the Security Oracle Service, installing and running Security Dashboard locally involves checking it out of GitHub using git and installing NPM dependencies. Atlasboard, and by extension Security Dashboard, are compatible with all version of NodeJS starting at 0.10. Assuming all local dependencies are installed, local installation would typically be as follows:

```
$ git clone https://github.com/SecurityOracle/SecurityDashboard.git
$ cd SecurityDashboard
$ npm install
```

You must then configure your Security Dashboard to monitor your application by entering its application ID in Security Dashboard's configuration files. Navigate to packages/default/dashboard/security.json and find the "appId" parameter. It's default value should be "02f007e1-8719-4ec0-aca9-49eeec7aa69d". Replace this value with your application ID. The easiest way to get your application id is to run

```
$ cf env "[application name]"
```

where [application name] is the name of an application as deployed to Cloud Foundry, and look for the “application_id” parameter in the resulting output.

Security Dashboard’s local installation of Atlasboard was also altered slightly to play nicely with Cloud Foundry. It should be possible to deploy Security Dashboard to Cloud Foundry using the standard

```
$ cf push
```

command, assuming Cloud Foundry is installed and configured correctly and that a standard manifest.yml file is present.