scarf-to-db USER GUIDE

This document's the program `scarf-to-db` that can be used to upload SCARF results into a NOSQL database (MongoDB) or SQL databases (PostgreSQL, MySQL, MariaDB or SQLite3). Uploading SCARF results into any DBMS involves the following steps:

1. Installing the DBMS (see Appendix A)
2. Installing Perl drivers (see Appendix B)
3. `scarf-to-db` configuration
4. Database table creation or deletion for SQL databases
5. Use `scarf-to-db` to add SCARF results to DBMS
6. Example command line execution
7. Database schema

The rest of this document will introduce all the above sections.

`scarf-to-db` configuration

To operate, scarf-to-db requires configuration. The configuration can be set using the command line, or via configuration files. `scarf-to-db` supports two configuration files: *scarf-to-db.conf* for database configuration, and *scarf-to-db-auth.conf* for database credential data (the permissions of this file should be restricted as it contains sensitive information). The use of the configuration files is optional, but recommended.

The value for an option is determined by first one of these that sets the value: 1) command line options, 2) the *scarf-to-db-auth.conf* configuration file, 3) the *scarf-to-db.conf* file, and finally 4) defaults built-in to `scarf-to-db`.

The location of configuration files can be set from an option before the option file is processed. If the configuration file location is explicitly set, it is an error if the file does not exist, but if the value is the default value, the configuration file is skipped if not present.

The remainder of these sections describes each option and is grouped by the most appropriate place to set the option starting with the scarf-to-db.conf options, then scarf-to-db-auth.conf options, and finally options most appropriately passed as command line options.

scarf-to-db.conf

The purpose of `scarf-to-db.conf` file is to configure the database, but other options can also be specified in this file to set their default values. The database related settings that can be configured using this file are as follows:

Option	Description
`db-type=<type>`	Database type - It can be any of the databases supported, default

Option	Description
`db-host=<host>`	Hostname of the DBMS server, default: localhost
`db-port=<port>`	Port on which the DBMS server listens on, default: 27017 (Mongo
`db-name=<name>`	Name of the database in which you want to save scarf results to. I
`db-commits=<max>`	Specifies the number of records or documents to be inserted atomi
`auth-conf=<path>`	Path to *scarf-to-db-auth.conf* file described in the next step (defau
`include-assess-report-file=<value>`	Adds the AssessmentReportFile name for the given bug instance (
`include-buildid=<value>`	Adds BuildId for the given bug instance (default: null)
`include-instance-location=<value>`	Adds the InstanceLocation information for the given bug instance

scarf-to-db-auth.conf

The purpose of `scarf-to-db-auth.conf` is to store database credential information if required by DBMS (New installation of MongoDB and SQLite does not require username or password). Since the information stored is sensitive, the file permission should be set accordingly so that only the owner can read the file. The database authentication related options configured in this file are as follows:

Option	Description
`db-username=<username>`	Username for DBMS
`db-password=<password>`	Password for DBMS

Command line options

The command line options can be used to specify any of the previous configuration file options, plus options that would be unique to each run of `scarf-to-db`. Command line options start with `--` and have same name as in the configuration file. Values are specified as `--key=value` or `--key value`. Some of the frequently used options have shorter key names (`short-key`) which can also be used to specify values using `-short-key value`. The `short-key` is mentioned with the option itself in the following table. The command line options are as follows:

Option	Description
`--scarf=<path>` or `-s <path>`	Path to the SCARF results XML or JSON (parsed_results.xml or parsed_r
`--conf=<path>`	Path to Config file containing database parameters (default location: curren
`--help` or `-h`	Prints out the help menu on the console and exits
`--version` or `-v`	Prints out the version of the program and exits
`--create-tables`	Creates tables for SQL databases and exits
`--delete-tables`	Deletes tables for SQL databases and exits
`--just-print` or `-n`	Prints out the commands used for database execution and exits (**Note:** You
`--test-auth`	Verifies the credential information provided in `scarf-to-db-auth.conf` file
`--pkg-name=<name>`	Name of the package that was assessed (default: null)
`--pkg-version=<version>`	Version of the package that was assessed (default: null)

Option	Description
`--platform=<name>`	Name of the platform on which the assessment was run (default: null)
`--verbose or -V`	Inserts data into the database and prints out the insert statements dependi
`--output-file=<name>`	Saves all the insert statement to the file provided using this option, default:
`--assess-id=<name>`	Unique id (for SQL databases) required when just printing out the the inser

Database table creation or deletion for SQL databases

SQL databases require tables to be created before importing SCARF data. However, MongoDB does not require any tables for storing data. `scarf-to-db` can be used to create or delete SQL database tables using the following command line options:

Option	Description
`--create-tables`	Creates tables for SQL databases and exits
`--delete-tables`	Deletes tables for SQL databases and exits

> The schema for the SCARF tables can be found in the section **Database Schema**.

Saving the SCARF results into a database

To save the SCARF results into a database (Assuming you have the DBMS and appropriate perl drivers installed), only the `scarf` command line option is required (assuming you are using defaults values and default configuration files location present in the program)

Example commands loading SCARF into a MongoDB database

Configure the `scarf-to-db.conf` and `scarf-to-db-auth.conf` files as mentioned in the previous sections. After configuring those files you should have content similar to the following configuration files:

scarf-to-db-auth.conf

```
db-username = user
db-password = password
```

scarf-to-db.conf

```
db-type = mongodb
db-host = my-mongo.swamp.cs.wisc.edu
db-name = scarf
auth-conf = scarf-to-db-auth.conf
```

Execution command

```
bin/scarf-to-db --scarf=./parsed_results.conf
```

> **Note:** If the above command executes successfully you will not see any output but the data will be saved in the database

Output:

For SQL databases:

- You will see similar insert statement only once per SCARF file

```
INSERT INTO assess (assessuuid, pkgshortname, pkgversion, tooltype, toolversion, plat) VALUE
```

- You will see insert statements similar to these per weakness

```
INSERT INTO methods VALUES  ('4', '1', '-1', null, null);
INSERT INTO locations VALUES  ('4', '1', '1', '1', 'lighttpd-1.4.33/src/lemon.c', '857', '85
INSERT INTO weaknesses VALUES  ('4', '1', 'Assigned value is garbage or undefined', 'Logic e
```

For MongoDB:

- You will see an array of documents similar to the following document

```
{
    "BugId" : 1,
    "BugRank" : null,
    "plat" : null,
    "toolType" : "clang-sa",
    "Methods" : [
    ],
    "classname" : null,
    "toolVersion" : "clang version 3.7.0",
    "BugSeverity" : null,
    "Location" : [
            {
                "LocationId" : 1,
                "EndLine" : 857,
                "StartLine" : 857,
                "primary" : true,
                "SourceFile" : "lighttpd-1.4.33/src/lemon.c",
                "StartColumn" : 9
            }
    ],
    "BugMessage" : "Assigned value is garbage or undefined",
    "BugCode" : "Assigned value is garbage or undefined",
    "pkgShortName" : null,
    "pkgVersion" : null,
    "assessUuid" : "138ad1cb-129e-4837-a376-eed3b2ed072f",
    "BugGroup" : "Logic error",
    "BugResolutionMsg" : null,
    "BugCwe" : null,
    "InstanceLocation" : null,
    "AssessmentReportFile" : null,
    "BuildId" : null
}
```

Note: The above output can used to manually import data to any of the supported databases.

Database Schema

MongoDB

- **BugInstance**

```
{
    "_id" : <unique MongoDB generated id>,
    "BugRank" : <String>,
    "plat" : <String>,
    "toolType" : <String>,
    "Methods" : [
        {
            "MethodId" : <int>,
            "name" : <String>,
            "primary" : <Boolean>
        }
    ],
    "classname" : <String>,
    "toolVersion" : <String>,
    "BugSeverity" : <String>,
    "Location" : [
        {
            "EndLine" : <int>,
            "StartLine" : <int>,
            "primary" : <Boolean>,
            "LocationId" : <int>,
            "SourceFile" : <path-String>,
            "StartColumn" : <int>,
            "EndColumn" : <int>,
            "Explanation" : <String>
        }
    ],
    "BugMessage" : <String>,
    "BugCode" : <String>,
    "pkgShortName" : <String>,
    "BugId" : <int>,
    "pkgVersion" : <String>,
    "assessUuid" : <uuid-String>,
    "BugGroup" : <String>,
    "BugResolutionMsg" : <String>,
    "BugCwe" : <String>,
    "InstanceLocation" : {
        "Xpath" : <path-String>,
        "LineNum" : {
                "Start" : <int>,
                "End" : <int>
            }
    },
    "AssessmentReportFile" : <path-String>,
    "BuildId" : <int>
}
```

- **Metric**

```
{
    "_id" : <unique MongoDB generated id>,
    "SourceFile" : <path-String>,
    "Type" : <String>,
    "pkgVersion" : <String>,
    "assessUuid" : <uuid-String>,
    "toolType" : <String>,
    "toolVersion" : <String>,
    "Value" : <String>,
    "plat" : <String>,
    "pkgShortName" : <String>,
    "MetricId" : <int>,
    "Method" : <String>,
    "Class" : <String>
}
```

- **If the package does not contain any BugInstance or Metric**

```
{
    "_id" : <unique MongoDB generated id>,
    "pkgVersion" : <String>,
    "assessUuid" : <uuid-String>,
    "toolType" : <String>,
    "toolVersion" : <String>,
    "plat" : <String>,
    "pkgShortName" : <String>
}
```

Schema (SQL databases)

Below is the schema for SQLite database. All other SQL databases have same schema with few minor changes for primary key. But, the column names and types is same for all SQL databases.

```
CREATE TABLE assess (
        assessId         integer PRIMARY KEY AUTOINCREMENT,
        assessUuid       text       NOT NULL,
        pkgShortName     text,
        pkgVersion       text,
        toolType         text       NOT NULL,
        toolVersion      text,
        plat             text
);
CREATE TABLE locations (
        assessId         integer    NOT NULL,
        bugId            integer    NOT NULL,
        locId            integer    NOT NULL,
        isPrimary        boolean    NOT NULL,
        sourceFile       text       NOT NULL,
        startLine        integer,
        endLine          integer,
        startCol         integer,
        endCol           integer,
        explanation      text,
        PRIMARY KEY (assessId, bugId, locId)
);
CREATE TABLE functions (
        assessId         integer    NOT NULL,
        sourceFile       text,
        class            text,
        method           text,
        startLine        integer,
        endLine          integer
);
CREATE TABLE weaknesses (
        assessId         integer    NOT NULL,
        bugId            integer    NOT NULL,
        bugCode          text,
        bugGroup         text,
        bugRank          text,
        bugSeverity      text,
        bugMessage       text,
        bugResolutionMsg text,
        classname        text,
        bugCwe           text,
        AssessReportFile text,
        BuildId          integer,
        ILXpath          text,
        ILStart          integer,
        ILEnd            integer,
        PRIMARY KEY (assessId, bugId)
);
CREATE TABLE methods (
        assessId         integer    NOT NULL,
        bugId            integer    NOT NULL,
        methodId         integer,
        isPrimary        boolean,
        methodName       text,
        PRIMARY KEY (assessId, bugId, methodId)
);
CREATE TABLE metrics (
        assessId         integer    NOT NULL,
        metricId         integer    NOT NULL,
        sourceFile       text,
        class            text,
        method           text,
        type             text,
        strVal           text,
        numVal           real,
        PRIMARY KEY (assessId, metricId)
);
```

Appendix A: Database Installation

Installing MongoDB

If you do not have MongoDB installed already, please follow the installation guide at https://docs.mongodb.com/manual/installation/

Installing MongoDB on RHEL based platforms

For installation specific to RHEL based platforms please see https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/

> NOTE: On `rhel-6.4-64` platform, executing `sudo yum install -y mongodb` will install an old version (2.4.14) of MongoDB. To install the latest version (3.2.8 or above) of MongoDB, please follow the steps in the section **Configure the package management system (yum)** in the tutorial https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/. This program is tested on

> MongoDB version (2.4.14, 3.0.12, and 3.2.8) with perl MongoDB driver version (1.4.2, and 1.4.4).

Example: To install `3.2.x` version of MongoDB on `rhel-6.4-64`:

Create a file named `/etc/yum.repos.d/mongodb-org-3.2.repo` and add the following content to the file

```
[mongodb-org-3.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.2.asc
```

Execute the following shell command to install MongoDB:

```
% sudo yum install -y mongodb-org
```

Check if the MongoDB server is running

If the installation is successful, please execute the following command to check if the MongoDB server is running.

```
# Invokes Mongo Shell
% mongo
```

If the above command fails with a message **exception: connect failed** then, MongoDB may not be running.

Execute the following command to run MongoDB:

```
% sudo /etc/init.d/mongod start
```

By default, MongoDB server listens on `localhost:27017` network interface. There are various options to access MongoDB across the network. See the MongoDB documentation for more information https://docs.mongodb.com/manual/reference/configuration-options/.

Authentication

By default, MongoDB does not require *root password or user accounts* to create databases and insert documents. If you like to authenticate and authorize users please see https://docs.mongodb.com/manual/tutorial/enable-authentication/.

> **Note:** If you notice any authentication related error messages and you are sure that password and username entered are correct, please check if the `authenticationDatabase` used for the user is same as the database that you are trying to access

Installing PostgreSQL

If you don't have PostgreSQL installed already, please follow the installation guide at https://www.postgresql.org/download/linux/redhat/

Installing MySQL

If you don't have MySQL installed already, please follow the installation guide at https://dev.mysql.com/doc/mysql-repo-excerpt/5.6/en/linux-installation-yum-repo.html

Installing MariaDB

If you don't have MariaDB installed already, please follow the installation guide at https://mariadb.com/kb/en/mariadb/yum/

Appendix B: Perl Drivers Installation

Installing Perl drivers

`scarf-to-db` program uses Perl drivers. Install the following Perl drivers on the machine that you would want to call scarf-to-db from

1. DBI
2. DBD::Pg
3. DBD::MySQL
4. DBD::SQLite
5. MongoDB
6. YAML
7. Config::AutoConf
8. JSON::MaybeXS

On `rhel-6.4-64` platform, execute the following commands to install the drivers using CPAN

```
sudo cpan DBI DBD::Pg MongoDB DBD::MySQL DBD::SQLite
```

> NOTE: On some `rhel-6.4-64` machines, users may also have to install `YAML and Config::AutoConf` packages from CPAN. To install the `YAML and Config::AutoConf` package, execute the following shell command:

```
% sudo cpan YAML Config::AutoConf
```

> NOTE: The above command may ask for user confirmation to install packages and its dependencies too many times. To avoid typing `yes` on the CPAN console too many time, please run the following commands:

```
% sudo perl -MCPAN -e shell  # Opens up a CPAN shell
    cpan[1]> o conf prerequisites_policy follow
    cpan[2]> o conf build_requires_install_policy yes
    cpan[3]> o conf commit
```

For more information on how to avoid the `yes` confirmation dialog please see https://major.io/2009/01/01/cpan-automatically-install-dependencies-without-confirmation/.

> NOTE: Using the CPAN command to install perl MongoDB driver you can install latest version of the driver. This program is tested on versions (1.4.2, and 1.4.4(latest)).