OPGen

OPGen is a tool that generates object property graph (OPG) and analyze vulnerabilities for JavaScript.

Installation

OPGen requires Python 3.7+ and Node.js 12+. To set up the environment, simply run `install.sh`.

Alternatively, you can do it manually:

```
pip3 install -r ./requirements.txt
cd esprima-csv
npm install
```

Command line arguments

Use the following arugments to run the tool:

```
generate_graph.py [-h] [-p] [-t VUL_TYPE] [-P] [-m] [-q] [-s] [-a]
                  [-f FUNCTION_TIMEOUT] [-c CALL_LIMIT]
                  [-e ENTRY_FUNC] [input_file]
```

Argument	Description
`input_file`	See subsection Input.
`-p, --print`	Print logs to console, instead of files.
`-t VUL_TYPE, --vul-type VUL_TYPE`	Set the vulnerability type to be checked. (See the Vulterability Types section.)
`-P, --prototype-pollution, --pp`	Shortcut for checking prototype pollution.
`-m, --module`	Module mode. Indicate the input is a module, instead of a script. This implies -a.
`-a, --run-all`	Run all exported functions in module.exports. By default, only main functions will be run.
`-q, --exit`	Exit the analysis immediately when vulnerability is found. Do not use this if you need a complete graph.

`-s, --single-branch`	Single branch mode (or single execution). Do not execute multiple branches in parallel.
`-f SEC, --function-timeout SEC`	Set the time limit when running all exported function, in seconds. (Defaults to no limit.) Do NOT use this parameter as it is very unstable.
`-c CALL_LIMIT, --call-limit CALL_LIMIT`	Set the how many times at most the same call statement can appear in the caller stack. (Defaults to 3.)
`-e ENTRY_FUNC, --entry-func ENTRY_FUNC`	Mannualy set the entry point function. Use this parameter only if you know which function to start the analysis with. This only affects the input module, i.e., dependent packages will not be affected.

Input

The tool accepts any of the three input types.

1. Use `-` to get source code from stdin.
2. Use a path to specify a source code file (or directory).
3. Ignore the argument to analyze AST CSV files (`./nodes.csv` and `./rels.csv`).

Examples

```
$ ./generate_graph.py paypal-adaptive@0.4.2 -mqp -t proto_pollution
```

Vulnerability Types

Currently OPGen can only detect prototype pollution. Use `-t proto_pollution`, `--pp` or `-P` to turn the detection on.

Batch analysis

OPGen includes powerful batch analysis scripts that allows packages to be analyzed in multiprocessing.

Go to the npmtest folder and use the following command to analyze packages with single process.

```
call_function_generator.py [-p] [-t VUL_TYPE] [-P] [-m] [-q] [-s]
                           [-a] [-f FUNCTION_TIMEOUT]
                           [input_directory]
```

Definitions of the arguments are the same as `generate_graph.py`. `input_directory` is

the directory where you store your packages. To analyze with multiprocessing, modify `chunkrun.sh`. Add input directory at the end of the command, modify the other arugments to meet your need.

Technical Details

Language support

OPGen uses Esprima to generate JavaScript AST. However, unlike Esprima, OPGen does not support a number of features, including those introduced in ECMAScript 2015 and later versions.

Compatibility with Joern

CSVs generated by esprima-csv are still compatibile with a modified version of the original Joern.

Directory structure

- `builtin_packages`: JavaScript modeled built-in packages.
- `esprima-csv`: Generate AST in CSV format with Esprima.
- `npmtest`: Scripts to analyze packages in batch.
- `simurun`: OPGen's core module, including simulated execution and vulnerability analysis.

Analysis process

OPGen simulates the execution of JavaScript code to build the OPG. In module mode, it runs every exported function, and the functions in their prototypes. Every time when a function is finished, it will traverse the graph to check if vulnerability is found. OPGen checks if there is a path between the exported function and the sink functions (except prototype pollution). Refer to the paper to know more about OPG and the system.

Output and logs

- `run_log.log`: Main logs. Use `-p` to print them to console.
- `testnodes.csv`: Nodes of the generated graph (including AST).
- `testrels.csv`: Edges of the generated graph (including AST).