Readme

Thank you for testing our tool. This is the detailed testing instruction.

1. Dockerize the tool

We provide two methods for testing:

- A customized virtual machine

```
Address: 128.220.247.60
Port: 40022
Username: guest
Password: yfeBG83%5wlO

ssh guest@128.220.247.60 -p 40022
```

- A docker image -- The docker image locates in the home directory of the guest user named "ODGen.tar". You can download and load it by:

```
docker load < ODGen.tar
```

Then you can attach to this docker by running

```
docker run -it odgen bash
```

After loading it, you should be able to see the same environment with the virtual machine

2. Instructions on how to run the tool with custom samples

```
python ~/projs/ODGen/odgen.py -t os_command -ma --timeout 300 /PATH/TO/YO
UR/FILE
```

Here the "-t" argument means the type of vulnerability, you can try "os_command", "proto_pollution", "code_exec", "ipt", "xss" and "path_traversal". The "-m" argument means that your input file is a module and "-a" means you want to test all the exported functions. "--timeout" arguement will set a timeout for the testing process.

Besides that, you can also try the "-q" argument to save some time, this arguement means that once ODGen finds any vulnerability, it will quit immediately. The "-s" argument will trigger the "branch-insensitive" mode, which will also speed up the testing.

3. A script that runs the analysis correctly on ReviewerA example

```
cd ~/examples
./run_prototype_pollution.sh
```

4. The dataset of the 180 zero-day vulnerabilities

- Dataset: ~/packages (Note that after our reporting, there are eight packages that are unpublished from NPM. Currently, we only have source code for 172 packages + one package, which is unpublished but cached on our server.)
- the CVEs they got: ~/packages/xx/package-name@version/cve.txt (if exists)
- a script that runs the analysis on each of these folders/projects and detects the vulnerabilities: ~/packages/xx/package-name@version/run.sh

where xx = code_exec, ipt, os_command, path_trasversal, proto_pollution, and xss.

5. Follow the artifact evaluation guidelines

See this readme.md and ~/projs/ODGen/README.md where we have a detailed instruction on different usages of this tool.

File Organization

Once you log into the virtual machine, all the files and folders are organized as follows:

```
.
|--projs: the source code and libs of our tool.
|--packages: all the zero-day vulnerable packages detected by our tool.
  |--code\_exec: packages with zero-day arbitrary code execution vulnerabil
    |--XX: package-name@version
      |--cve.txt: if it exists, it indicates the CVE identifier
      |--run.sh: a script to detect the zero-day vulnerability
  |--ipt: packages with zero-day internal property tampering vulnerabilitie
  |--os\_command: packages with zero-day OS command injection vulnerabiliti
  |--path\_traversal: packages with zero-day path traversal vulnerabilities
  |--proto\_pollution: packages with zero-day prototype pollution vulnerabi
  |--xss: packages with zero-day XSS vulnerabilities
|--examples: a few simple vulnerable examples
  |--pp\_example.js: the prototype pollution example written by reviewer A
  |--run\_proto\_pollution.sh: detect prototype pollution of pp\_example.js
  |--motivating\_example.js: the motivating example mentioned in the paper
  |--run\_ipt.sh: detect internal property tampering of motivating\_example
  |--run\_os\_command.sh: detect taint-style vulnerability of motivating\_e
  |--clean.sh: clean up log files
|--back\_up: recovery files (do not touch)
```