Contact
Please contact Phakpoom (Patrick) Chinprutthiwong via email (cpx0rpc@tamu.edu) if there is any question regarding the tool and data set.

Introduction
Our tool, SW-Scanner, runs as a website. Our instructions will include setting up the server, the website environment, and how to use. The source code and the virtual machine for the tool can be downloaded from https://u.tamu.edu/sw-scanner. The service worker parameters can be found at */swscanner/swinfo.lst*. The *new_sw* data set (corresponding to Section 6.1) can be found at */swscanner/web/included_files/new_sw.zip*. There are 7,060 websites that are included in this data set. Among them, only 6,144 are compatible with our tool. The *old_sw* data set (corresponding to Section 6.3) can be found at */swscanner/old_sw/old_sw.zip*. There are 1,886 websites in this data set. The main source code of SW-Scanner, which is a website, can be found at */swscanner/web/*.

Using the VM

1. Import our virtual machine file (found at */vm/swxss.ova*). Then login using the password “swxss”. Our VM has everything set up, so you do not have to install any extra library.

2. To reproduce the result of our taint analysis (Section 6.2 in the paper), simply execute the *analyze.sh* file (at *~/swscanner/analyze.sh*).

```
/home/swxss/swscanner/analyze.sh
```

This script will start spawning Chrome and analyzing all websites. You can inspect the *analyze.sh* file to see how to selectively analyze a website. Ultimately, our tool needs to execute two phases. Therefore, for each website, there will be two URLs that you have to run consecutively. Make sure to wait at least 20 seconds for each phase and restart the browser to clear the cache for each website. The output will be saved in */var/www/html/swscanner/*results.

Alternatively, we also include the output from our own run (at *~/swscanner/*results/). Inside a log file, the “[FATAL]” tag represents a sensitive flow. Two examples of such cases are websites #423 and #20754. We refer to section **How to run SW-Scanner** below for more details in how to analyze websites that are not from our data set. If you want to get the list of all reported websites with the “[FATAL]” tag, simply run the *summarize_result.py* script.

```
python summarize_result.py ./results
```

This will list out 54 websites, but not all of them are vulnerable. As we mentioned in the paper, our final filter includes determining whether the tainted value is a URL. However, in this version, we did not filter to show other general cases as well (i.e., websites that use non-URL related parameters). Furthermore, there are 7 websites that we contacted have asked us to remove them from the results. Therefore, there are only 33 vulnerable websites included in this list.

Note that running the analysis using Terminal may produce a sandbox-related error to the Terminal, but the error does not affect our tool. Furthermore, due to different performance tuning of each environment, you may need to adjust/increase the delay between each analysis as seen in the *analyze.sh* file (sleep command). Currently, we also limit the max file size. To increase it, edit the *__internal__MAX_FILE_SIZE__* in the file *analyze.js* (at */swscanner/web/analyze.js*). Adjusting these variables may help when encountering warnings/errors produced by our tools.

3. To reproduce the result of the service worker freshness (Section 6.3 in the paper), execute the *plot.py* (at *~/swscanner/plot.py*) with the following arguments.

```
python plot.py ./old_sw
```

Setting up web environment (in your own machine)

1. Install Apache2

```
sudo apt install apache2
```

2. Install PHP

```
sudo apt install php libapache2-mod-php
```

3. Restart web server

```
sudo systemctl restart apache2
```

[Optional] Installing Babel
While our tool does not strictly require Babel to use, it is recommended that the source code being analyzed is in a proper syntax. Sometimes web browsers can tolerate certain level of syntax mistakes, but such mistakes may affect our tool.

1. Install NodeJS

```
sudo apt install nodejs
```

2. Install NPM

```
sudo apt install npm
```

3. Install Bluebird

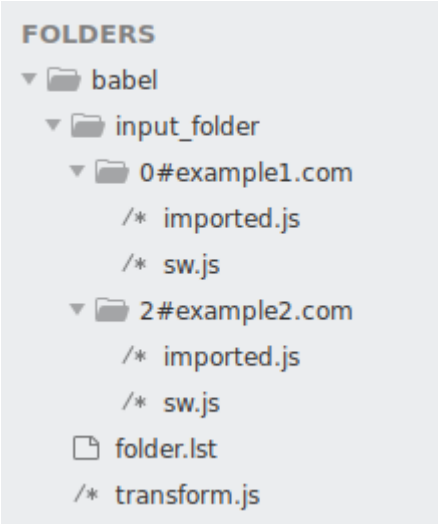
```
npm install bluebird
```

4. Install Babel

```
npm install --save-dev @babel/core
npm install --save-dev @babel/plugin-transform-arrow-functions
npm install --save-dev @babel/plugin-transform-template-literals
```

How to use Babel

1. Prepare a folder to contain the target service worker file(s). Inside the folder, create any number of folders corresponding to each website you want to transform the files.



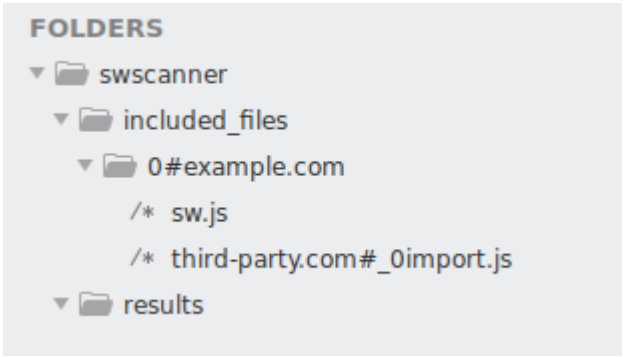
2. Create a file that contains a list of the sub-folders, separated by a new line.

3. Run *transform.js* (at */swscanner/transform.js*), which is a script we prepare for you to transform possibly malformed files into well-formed ones. The resulting files are in the same input folder(s). This script takes two arguments: the path to the folder and the path to the folder listing file.

```
nodejs transform.js ./input_folder ./folder.lst
```

How to use SW-Scanner

1. Download the source code of SW-Scanner (at */swscanner/web/*) and extract it to the web server location. Make sure to (re)name the folder to “swscanner” as shown in the figure below. In Ubuntu, the default path should be */var/www/html*. This should generate the following folder structure.



2. Add write permission for the tool since our tool writes log files into the localhost folder.

```
sudo usermod -a -G www-data [user_name]
sudo chgrp -R www-data /var/www
sudo chmod -R g+w /var/www
```

If you do not want to give write permission, you can simply use Chrome's Devtools to inspect the console while running SW-Scanner. Our tool also logs the output through *console.log*.

3. To analyze a website, first create a folder inside the "included_files" sub-folder. The folder should be named as [number]#[domain] like *0#example.com* as shown in the figure above. We also provide the data set used by our paper named as *new_sw.zip* (at */swscanner/web/included_files*). You can also directly extract the zip file we provided into the *included_files* folder.

3.1. In the case that you want to analyze a website that is not from our data set, download and copy the target service worker file (and other files imported by the service worker) into the folder. Rename the main service worker file to "sw.js" and the imported files should be renamed as how it is used in the importScripts API.

```
<sw.js>
self.importScripts("https://www.third-party.com/import.js");
...
```

For example, given that the service worker file has the above source code, the import.js file should be named as "third-party.com#_0import.js". Note that the preceding protocol (https:// and www.) should be removed, and the "/" should be replaced by "#_0". This is done to make every file into the same folder and reduce the complexity for setting up the sub-folder.
Because SW-Scanner takes URL's parameter as commands, we need to generate the correct URL. The correct URL should look like the following.

```
localhost/swscanner/index.html?
mode=init&filename=[folder_name]&domain=[domain_name]&swurl=[service_worker_url]
&params=[service_worker_parameters]
```

And

```
localhost/swscanner/index.html?
mode=analyze&filename=[folder_name]&domain=[domain_name]&swurl=[service_worker_url]
&params=[service_worker_parameters]
```

For example, given that the target website (www.example.com) registers a service worker as serviceworker.*js?user_id=Alice*, then the command should look like the following.

```
localhost/swscanner/index.html?
mode=init&filename=0#example.com&domain=example.com&swurl=https://www.example.com/
sw.js?user_id=Alice&params=user_id=Alice
```

And

```
localhost/swscanner/index.html?
mode=analyze&filename=0#example.com&domain=example.com&swurl=https://
www.example.com/serviceworker.js?user_id=Alice&params=user_id=Alice
```

Finally, execute both URLs consecutively. Make sure to wait at least 20 seconds for each phase.

3.2 In the case that you want to use our data set, we have already generated all the commands in *analyze.sh* (at */swscanner/analyze.sh*). This file is used to analyze all website in our data set using Chrome. To use this file, make sure you have Chrome installed. Then, copy this file to any folder and execute it. Note that this script may terminate all Chrome windows, not just the one being used for the analysis. To simply analyze a certain case, simply search for the website's name and copy the URL used by our script. Our script has two URLs for each website corresponding to the two phases (determining the path taken and the actual taint analysis). Make sure to use both URLs consecutively and give each phase at least 20 seconds.

4. After the analysis is finished, check the result in the *results* folder. The result file will be named as the folder name (i.e., *0#example.com*). We have also included all result files in the *results* folder (at */swscanner/web/results/analysisresult.zip*) generated from our analysis. When SW-Scanner finds a sensitive flow, it will report with the "[FATAL]" tag.

For example:

```
[FATAL] An importScripts argument contains tainted value.
{"value": "https://prod.webpu.sh/TwCT31YGGaOAVbHmJAlrKUgzYN0aB3BZ/service-worker-source.js?v=1.9.8", "taint": true, "taint_src": 1, "taint_list": ["https://prod.webpu.sh/TwCT31YGGaOAVbHmJAlrKUgzYN0aB3BZ/service-worker-source.js?v=1.9.8"]}
```

Note that the result files we included were run in a different environment, so the path shown may be different, and there could be logging messages that should not be presented in the released version. Additionally, we limit the tool to skip analyzing files that are too large. Because we performed the analysis using a commodity laptop, sometimes analyzing large files can consume too much memory. However, you can change the limit by editing the *__internal__MAX_FILE_SIZE__* in the file *analyze.js* (at */swscanner/web/analyze.js*).

5. To start a new analysis task, please make sure to close and re-open the browser to clear the cache of SW-Scanner.

How to use the old service worker files artifact

1. Install pip

```
sudo apt install python-pip
```

2. Install NumPy

```
pip install numpy
```

3. Install Matplotlib

```
sudo apt install python-matplotlib
```

4. Run the *plot.py* (at */swscanner/plot.py*) with an argument pointing toward the *oldsw* path (at */swscanner/old_sw*). Make sure to extract the zip file *old_sw.zip* before executing the script.

```
python plot.py ./old_sw
```