Automating your reconnaissance workflow with 'meg'

Apr 13, 2018

For the past few months, I have been playing around with a tool developed by Tom Hudson called ‘meg’ and I have fallen in love with this tool. meg is a lightweight URL fetcher that stores the output in organised directories and files. This tool has become the quintessential element in my reconnaissance workflow and has been incorporated into many of my personal tools.

Tom comes from a developing background and therefore understands full well the issue with mass-scanning. This is why meg was designed with the primary focus of not being resource intensive. By default, the tool behaves like a normal user browsing the web and is less likely to take down a service as a result.

The Basics

In bare essence, all it takes to run meg is to specify an endpoint and then a host.

```
$ meg <endpoint> <host>
$ meg / https://edoverflow.com
```

The latter command requests the top-level directory for https://edoverflow.com (`https://edoverflow.com/`). It is important to note, that protocols most be specified; meg does not automatically prefix hosts. If you happen to have a list of targets without protocols, make sure to `sed` the file and add the correct protocol.

```
$ sed 's#^#http://#g' list-of-hosts > output
```

By default meg stores all output in an `out/` directory, but if you would like to include a dedicated output directory, all it takes is appending the output directory to your command as follows:

```
$ meg / https://edoverflow.com out-edoverflow/
```

Discovering Interesting Files On The Web

Say we want to pinpoint specific files that could either assist us further while targeting a platform or be an actual security issue in itself if exposed to the public, all it takes is a list of endpoints (`lists/php`) and a series of targets (`targets-all`). For this process, storing all pages that return a “200 OK” status code will help us sieve out most of the noise and false-positives (`-s 200`).

```
$ meg -s 200 \
  lists/php targets-all \
  out-php/ 2> /dev/null
```

Once the command above has finished running, all it takes is navigating through the `out-php/` directory to find saved responses. You can start by manually grepping for specific strings — make sure to automate part of this process in Bash with `grep` and arrays — or you can use a neat little trick that Tom showed me.

```
$ cat ~/bin/vimprev
#!/bin/bash
VIMENV=prev vim $@
```

You guessed it — vim preview windows! For each file in the output directory, we can preview it inside of vim. All the “exiting vim” jokes aside, this is a brilliant idea and I cannot thank Tom enough for it.

```
$ cat /path/to/vimrc
if $VIMENV == 'prev'
 noremap <Space> :n<CR>
 noremap <Backspace> :N<CR>
endif
$ export PATH=$PATH:~/bin
$ cd out/
$ vimprev $(find . -type f)
```

The `vimrc` trick allows you to use the spacebar to move to the next file and the backspace key to the previous one. Hold that spacebar down and go wild! Interesting-looking files should jump out fairly quickly.

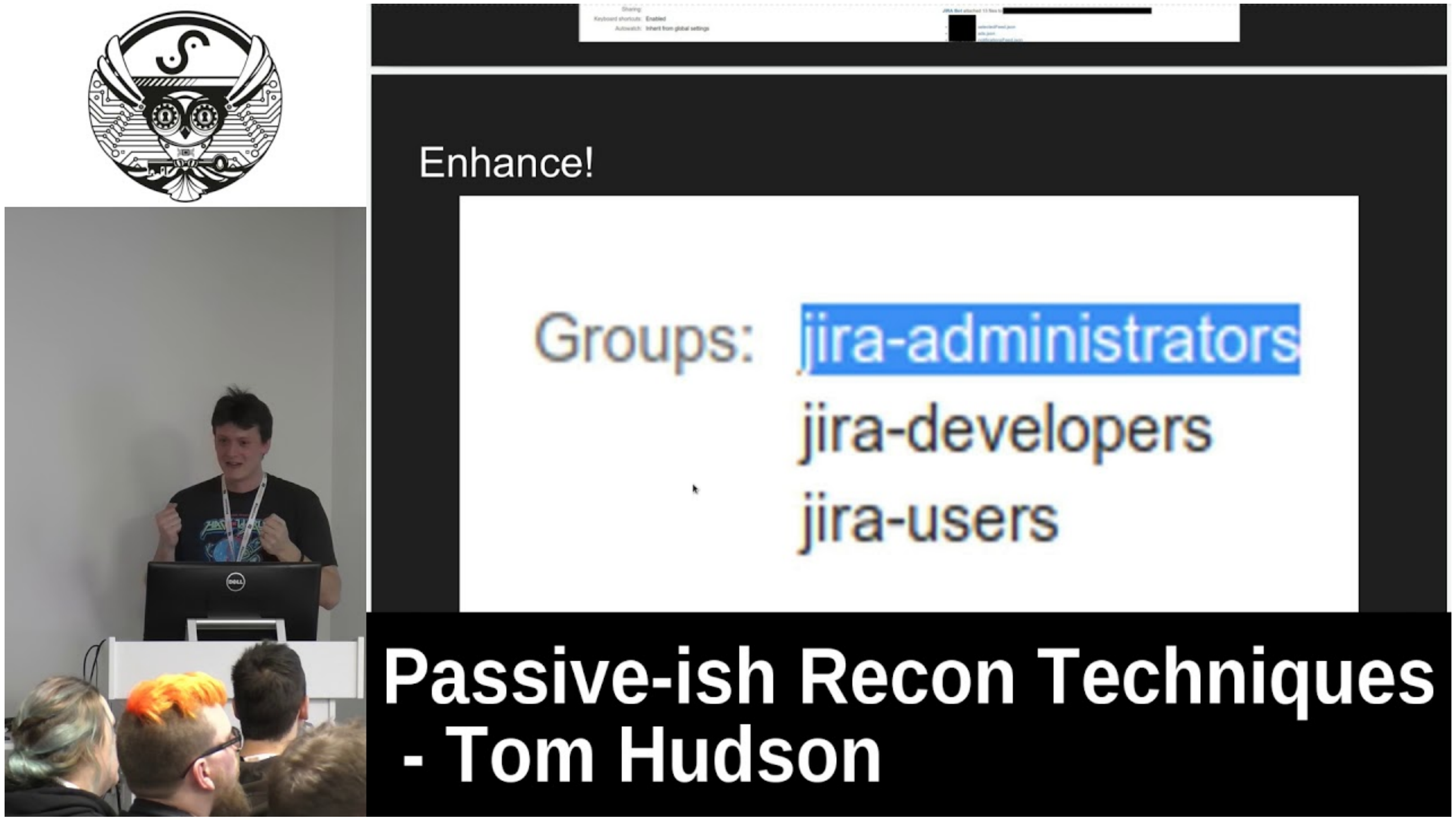
Live Preview

If you want a live preview of saved files in the output directory, all you need to do is follow the `index` file in the output directory the same way you might follow your logs on a web server.

```
$ cd out/
$ tail -f index
out/example.com/2d676fb9c99611db7e6cb75ffa1b137673f4ca04 http://example.com/.well-known/sec
```

Time To Experiment

I am going to cut this write-up short and encourage readers to watch Tom's talk "Passive-ish Recon Techniques". All the techniques described in this talk can be automated using meg and as Tom demonstrates they can be extremely rewarding.



Enhance!

Groups: jira-administrators
jira-developers
jira-users

Passive-ish Recon Techniques
- Tom Hudson

Support my work

If you enjoy reading my write-ups and would like to support my work, please check out my "Buy me a coffee" page.

☕ Buy me a coffee

Subscribe to my newsletter 📨