Volume Thirty, Number Two
DIGITAL EDITION Summer 2013

2600

The Hacker Quarterly

Western Electric
Change PIN
Enter
Clear
Cancel
Audio Jack
For customer assistance, or to report problems with this ATM, please call
Do Not Deposit Paper Towels Into ATM Machine!! Thanks.

Articles

The Road to Safety

by 2600 Magazine - The fallout from the Boston Marathon bombings didn't take long to settle upon all of us and begin to contaminate what remains of a free and open society. This kind of a thing was inevitable and it would be a challenge to find anyone overly surprised by how it's played out so far. What isn't inevitable is wh...

Splunking the Google Dork

by G Dorking - The number of awesome tools for vulnerability assessments is constantly growing. Recently, I was made aware of SearchDiggity by Stach and Liu, which is a nicely bundled tool for search engine dorking. For the uninitiated, "Google dorking" is feeding queries into Google that render interesting results. Two exa...

Fun with the Minuteman III Weapon System

by Bad Bobby's Basement Bandits - Otherwise known as the 21M-LGM30G Intercontinental Ballistic Missile. Your typical Missile Wing consists of 50 Minuteman III ICBMs. Each missile is located on its own plot of ground, usually located on part of someone's farmland. There are many articles and videos of individuals exp...

My First Blackhat

by Pierre LC - I'm a 30-something software engineer who's always been interested in hacking. The earliest code I can remember is writing BASIC programs when I was four years old, just to see if I could tell the computer what to do. This type of thinking naturally led to an interest in computer security, but my career in legitimate...

How to Create and Operate a Temporary Free Autonomous Zone

by lifeguard - This how-to is intended to document a framework of protocols and techniques to organize a large, diverse group of individuals voluntarily gathered together for a shared purpose, and in a public space. For example, a hacker carnival. Or to respond to a community crisis. It is assumed this gathering will happen...

Telecom Informer

by The Prophet - Hello, and greetings from the Central Office! Or at least what passes for a Central Office in my life these days. It has been a whirlwind few months in Rotterdam, and I am still neck-deep in management training. I am preparing for a future life as a silver-haired executive, and it's a huge change of pace. Rather th...

A Broad Spectrum of DRM

by Cybermouse - In my years as a computer user, I've seen quite a wide spectrum of DRM, or digital rights management. I will not be discussing music DRM, as I've not had much experience with it, and the way it is accomplished is fundamentally different than how software DRM is handled. Typically, software DRM is eith...

Getting Free Media - All Without Torrents!

by B4tm@n - Disclaimer: All of the information in this article is for educational use only. If you use this and get sued, don't blame me. Everyone loves media, and I'm guessing a good amount of people love getting it for free. Now, many people love using torrents to get their fill. However, with talk of ISPs subpoenaed by the RIA...

Tech Gets Better, Humans Do Not: A Beginner's Guide to Social Engineering

by jk31214 - Working in IT, I hear people talk about social engineering, and what they think it is. Most of the time, they think it's evil hackers on the Internets trying to gain access to their Facebook accounts, to engage i...

Splunking the Google Dork

by G Dorking

The number of awesome tools for vulnerability assessments is constantly growing. Recently, I was made aware of SearchDiggity by Stach and Liu, which is a nicely bundled tool for search engine dorking. For the uninitiated, "Google dorking" is feeding queries into Google that render interesting results. Two examples are:

```
    big brother status green
intitle:index.of id_rsa pub
```

These types of queries provide a high grade of attacker level visibility, but can be used by a defender to examine their own web presence using the "site:<domain>" param like so:

```
    site:example.com        big
brother status green
```

SearchDiggity supports most major search engines and comes preloaded with several popular query sets.

Another interesting tool is Splunk, a log analysis and intelligence solution. Splunk and its capabilities are extensive and useful enough to warrant their own article but, in brief, Splunk provides access to log data and statistics in seconds via a custom search dialect and indexing engine. The Splunk engine can digest just about any text based log (even tarballs of old logs), making it a great tool for processing text based data.

What If?

What if we digested the results of Google dorking in Splunk? This allows for the creation of dashboards, vulnerability tracking over time, and very very fast searching of the results.

Google provides access to their REST API to allow for programmatic access with a courtesy 100 free requests per day. Additional search volume can be purchased on a charge per use model ($5 per 1000 queries) and at much more significant annual quotas for more significant amounts of money.

Access to the API requires a custom search engine (defined through a Google account) and an API access key (managed through the developer console).

REST API: `https://developers.google.com/custom-search/v1/overview`

Google APIs Console: `https://code.google.com/apis/console/`

Google + Python - SearchDiggity

After working with SearchDiggity a bit and fiddling with some other data in Splunk, it occurred to me that I could readily digest the SearchDiggity results with Splunk (via some minor output modifications). I also wanted to stagger my requests across multiple days as I iterated through the query set (and stay under the 100 free requests limit), which seemed infeasible with SearchDiggity.

A couple of evenings hacking on the Google APIs with Python and I realized it was almost as simple to make the requests myself, as opposed to trying to manipulate the SearchDiggity output. A couple more evenings and some gold plating requests from friends, and the script as it currently stands emerged.

Script

The present Google dorking "script" is a collection of config files and a script to make the Google API requests. Through the config file, the number of requests per run can be controlled and the output format stipulated.

I installed the script on one of my Centos servers and call it daily with a cron job. It writes results to a directory that Splunk monitors and my network intel dashboard updates every day with the results of the most recent query set. Query run statistics are written to syslog for debug and logging purposes.

In the interest of saving space (and making things easy to get at), I've put the scripts on GitHub with their supporting files. They can be downloaded here:

```
https://github.com/-
searchdork/googledorking
```

Installation is as simple as cloning the git repository to somewhere on your server and adjusting the config files to point to the right places. The default install location is: /opt/googledorking

A default installation can be achieved through the following commands (`$` denotes bash prompt. All commands given here assume root privileges for the sake of brevity - feel free to modify permissions as you see fit. If you don't have git installed, run this first):

```
$ yum install git
```

Then:

```
$ cd /opt
$ git clone git@github.com:sea-
rchdork/googledorking.git
```

(This requires that you have your ssh keys added to GitHub.)

The next steps will require a Google custom search engine and API key. To create your custom search engine (which will define what sites you search), go to:

```
http://www.google.com/cse/
```

(1) Select "Create a custom search engine".

(2) Fill out the fields as needed, check and click the "Create" button if you agree to the ToS.

(3) Test your search engine to make sure it can find something on the sites you specified, then click "Edit".

(4) Copy the search engine unique ID field (should be a bunch of numbers, then a colon followed by a bunch of letters).

(5) Save this ID for future use.

To set up a search API key, visit:

```
    http://code.google.com/api-
s/console/?api=customsearch
```

(1) Create a project to associate with the key by selecting the "Create project..." button.

(2) Once again, if you agree to the ToS check the box and hit "Accept".

(3) And one more time... (another ToS).

(4) Select the link on the left for "API Access".

(5) Copy the API key listed in the "API Access" section.

Using the text editor of your choosing, edit the lines for api-key and custom-search-id in etc/googledorking.cfg with your own values from above.

There is more detailed information in the README regarding further customization of the config file.

Splunk

Installing Splunk on Linux is pretty much as simple as downloading the Splunk tarball and extracting it (receiving the download link requires creating a free splunk.com account).

I used wget to download the tarball (at the download link provided by Splunk); if you don't have wget installed, you can add it by issuing the below command (all commands here assume root privileges for the sake of brevity - adjust permissions according to your own tastes):

```
    $ yum install wget
```

To download splunk:

```
    $ wget "http://download.sp-
lunk.com/releases/4.3.3/splunk-
/linux/splunk-4.3.3-<######>-
Linux-x86_64.tgz"
```

where "######" is the Splunk build version (or something of the sort - the link may have changed by the time of publication).

I run everything for this exercise from the /opt/ directory, so I extracted the Splunk tarball there too:

```
    $ mv splunk-4.3.3-######-
Linux-x86_64.tgz /opt/

    $ cd /opt
$ tar xzf splunk-4.3.3-######-
Linux-x86_64.tgz
```

To start Splunk, simply run it from the extracted directory:

```
$ /opt/splunk/bin/splunk start
```

Making sure that Splunk has the right sourcetype is the trickiest part. To add the googledorking sourcetype, insert the following stanzas into the Splunk props.conf and transforms.conf files. (If you have not used Splunk before, you may not have either of these files. Just create them if they do not exist.)

```
file: /opt/splunk/etc/system/local/props.conf


[google_dorking]
CHECK_FOR_HEADER = false
SHOULD_LINEMERGE = TRUE
pulldown_type = 1
TRANSFORMS-headerToNull = google-dork-null-header
REPORT-extractFields = google-dork-field-extract



file: /opt/splunk/etc/system/local/transforms.conf



[google-dork-null-header]
REGEX = ^\#\#.*$
DEST_KEY = queue
FORMAT = nullQueue


[google-dork-field-extract]
DELIMS="\t"
FIELDS=time,query_set,category,search_string,title,url,display_link,cache_id,snippet
```

Once modifications have been made to the transforms.conf file, Splunk requires a restart for them to take effect:

```
$ /opt/splunk/bin/splunk restart
```

Edit Splunk's input types to monitor the directory or files that the Google dorking script will write to, and assign the newly minted googledorking sourcetype to this input. To do this:

(1) Log into the Splunk web interface at http://localhost:8000/ (or wherever you configured it).

(2) Click on "Manager" in top right.

(3) Select "Data Inputs" on the right.

(4) Click the "Add data" button.

(5) Click "A file or directory of files" from the presented links.

(6) Under "Consume any file on this Splunk server," click "Next".

(7) Select the "Skip preview" radio

button (Splunk is bad at previewing data with transforms), then click continue.

(8) Under full path to your data, put the path to the googledorking results folder (config default is /opt/googledorking/results).

(9) Check the box for "More settings".

(10) Under "Set the source type," select "From list".

(11) Under "Select source type from list," select "google_dorking".

(12) Click the save button.

To see your results (if/when you have any), select "Search" from the App pull down menu at the top right. Search for:

```
sourcetype="google_dorking"
```

Cron

Once the script is in place and is verified working, the crontab can be configured as follows:

If your system is missing cron (mine was), vixie-cron can be installed with the below command:

```
$ yum install vixie-cron
```

The crontab can be updated with "crontab -e":

```
$ crontab -e
```

Insert the below line to run the script every day at 2:04 am (arrange to your own personal preference):

```
04 02 * * * /opt/googledor-
king/bin/runGoogleDorking.py
```

Assuming default configuration, this should make 90 queries a day and the results should be immediately visible in Splunk. How you use them is up to you. I strongly encourage checking out Stach and Liu's collection of queries (and others) listed in the README. Happy hacking/splunking/-dorking! ■