HASH MONITORING SETUP

I like working on cracking password hashes (in case you are new here, see this) but nice big data sets don't come along often (good news, bad news) but there are lots of smaller dumps that happen (again.. good news/bad news). Most of these dumps are put on sites like pastebin, and in order to help people stay up to date there are monitoring bots that notify you if something is found. The best of these even notify folks via twitter (for example, @pastebindorks @dumpmon etc). This is very useful and very helpful. But it does mean you need to monitor the accounts. And since some of these paste sites have taken to removing these dumps, you need to act quickly. This causes problems for all of us unable to monitor our twitter feeds 24x7. But in the normal spirit of the security minded community ("I will replace you with a small shell script") the folks at TekDefense have created a python (not quite shell) script to monitor certain twitter accounts and parse through the links provided to download and store the hashes.

Needed
Obviously you need

- a twitter account
- the HashMonitor script itself (see here)
- python 2.7
- some required libraries (sqlite3, re, datetime, httplib2, argparse, sys - use "*pip install <library>*" to get those you need)
- also "*python-twitter*"

Python-twitter
Once you have downloaded "*python-twitter*" (see here using "*git clone git://github.com/bear/python-twitter.git*") you will need to:

- run "*pip install -r requirements.txt*"
- run "*python setup.py build*"
- run "*python setup.py install*"
- go to "*dev.twitter.com*", login and create a new application following instructions. Once you have your app details (consumer secret, tokens, etc) then you run "*get_access_token.py*" and use those details to allow access. Make sure to note the details from this process.

HashMonitor script
Now that the groundwork is done, go to the *HashMonitor.py* script. You will need to edit it, change this:

```
api = twitter.Api()
```

and use the details I told you to note earlier to make it look like this:

```
api = twitter.Api(consumer_key='xx', consumer_secret='xx', access_token_key='xx',
access_token_secret='xx')
```

If all is setup right, you should be able to run the script and see something like:

```
> ./hashMonitor.py

[*] Running hashMonitor.py
https://twitter.com/PastebinDorks
https://twitter.com/dumpmon
http://www.leakedin.com/
[*] Adding links to the DB if they have not been scanned previously.
[+] Adding http://pastebin.com/raw.php?i=LNTgKMa4 into the DB
[+] Adding http://pastebin.com/raw.php?i=nCmrM1gC into the DB
[+] Adding http://pastebin.com/raw.php?i=0Yy7wsXE into the DB
[+] Adding http://pastebin.com/raw.php?i=fskdqypx into the DB
[-] 86 links were previously scanned
[+] Searching for hashes in the new URLs
[*] Inserting new hashes to the DB if any are found.
[+] Adding ea2c66ddeb13ff3360e343b98413de1d to the DB
[+] Adding b60ef33c2d1d9aa2db2271d9bbd5f24b to the DB
..
<snip>
..
[+] Added 1188 Hashes to the Database
```

Final Steps
So we now have a working script. Lets get it automated, to do this we need to remember some points:

- Unless you specifically state the DB to use (the "*-d*" option), the script will create the DB (*hashMon.db*) in the directory from which it is invoked
- Use *crontab* to schedule running at intervals (**/30 * * * * /bot/TekDefense-hashMonitor/hashMonitor.py*)
- Use the "*-s*" option to see what you have gathered so far
- Use the "*-h*" option to see the other options for listing data, etc

So now we have a scheduled task which will download all those nice hashes for us. There are some gotcha's depending on your point of view. One is that there is no attribution, you will get the hashes but you will not know where the leak came from if that was mentioned in the original dump. But what you do get is real hashes, in an easy way, and you can use them to learn and contribute back. Try it out, have fun and learn.