HashTag.py

The Parsing and Identification of Password Hashes

**Introduction**

HashTag.py is a tool written in python which parses and identifies various password hashes based on their type. HashTag was inspired by attending [PasswordsCon 13](http://www.youtube.com/playlist?list=PLdIqs92nsIzTphHrDlIucuMP2vatvd4UL) in Las Vegas, KoreLogic’s ‘[Crack Me If You Can](http://contest-2013.korelogic.com/)’ competition at Defcon, and the research of [iphelix](http://thesprawl.org/iphelix/) and his toolkit [PACK](http://thesprawl.org/projects/pack/). HashTag supports the identification of over 250 hash types along with matching them to over 110 [hashcat](http://hashcat.net/oclhashcat-plus/) modes. HashTag is able to identify a single hash, parse a single file and identify the hashes within it, or traverse a root directory and all subdirectories for potential hash files and identify any hashes found.

**Hash Identification**

One of the biggest aspects of this tool is the identification of password hashes. The main attributes I used to distinguish between hash types are character set (hexadecimal, alphanumeric, etc.), hash length, hash format (e.g. 32 character hash followed by a colon and a salt), and any specific substrings (e.g. ‘$1$’). A lot of password hash strings can’t be identified as one specific hash type based on these attributes. For example, MD5 and NTLM hashes are both 32 character hexadecimal strings. In these cases I make an exhaustive list of possible types and have the tool output reflect that. During development I created an excel spreadsheet which contains much of the hash information I used:

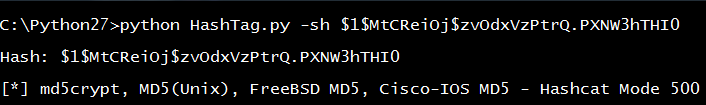
**Usage**

HashTag.py {-sh hash |-f file |-d directory} [-o output\_filename] [-hc] [-n]

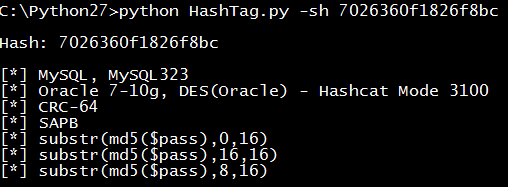
|  |  |
| --- | --- |
| -h, --help | show this help message and exit |
| -sh SINGLEHASH, --singleHash SINGLEHASH | Identify a single hash |
| -f FILE, --file FILE | Parse a single file for hashes and identify them |
| -d DIRECTORY, --directory DIRECTORY | Parse, identify, and categorize hashes within a directory and all subdirectories |
| -o OUTPUT, --output OUTPUT | Filename to output full list of all identified hashes.  --file default filename: HashTag/HashTag\_Output\_File.txt  --directory default filename: HashTag/HashTag\_Hash\_File.txt |
| -hc, --hashcatOutput | --file: Output a file per different hash type found, if corresponding hashcat mode exists  --directory: Appends hashcat mode to end of separate files |
| -n, --notFound | --file: Include unidentifiable hashes in the output file. Good for tool debugging (Is it Identifying properly?) |

**Identify a Hash**

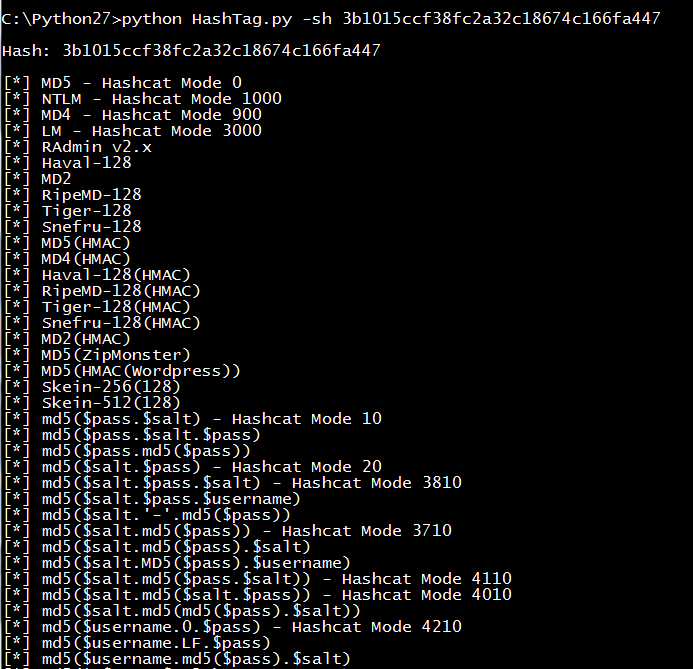
HashTag.py -sh $1$MtCReiOj$zvOdxVzPtrQ.PXNW3hTHI0



HashTag.py -sh 7026360f1826f8bc

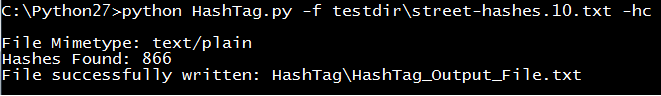


HashTag.py -sh 3b1015ccf38fc2a32c18674c166fa447



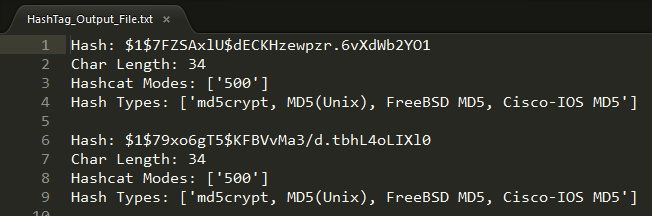
**Parsing and Identifying Hashes from a File**

HashTag.py -f testdir\street-hashes.10.txt -hc

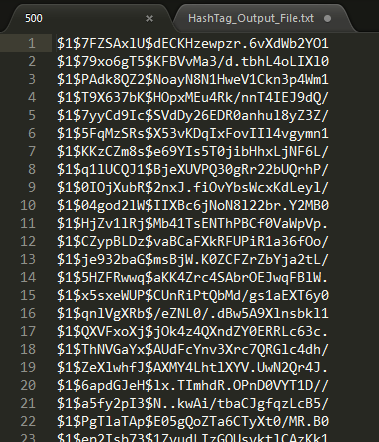
****

Output:

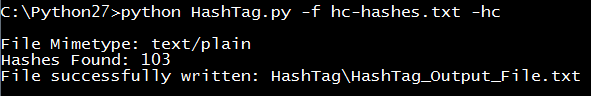
Each identified hash outputs the hash, char length, hashcat modes (if found) , and possible hash types.



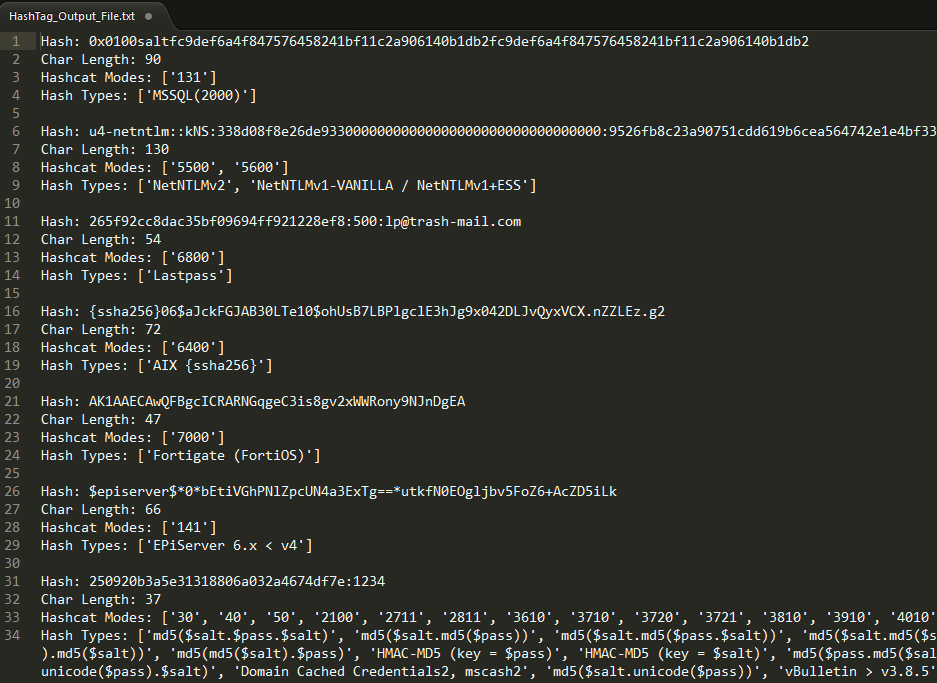
Using the -hc/--hashcat argument we get a file for each hash type if a corresponding hashcat mode is found. This makes the process of cracking hashes with hashcat much easier as you immediately have the mode and input file of hashes.



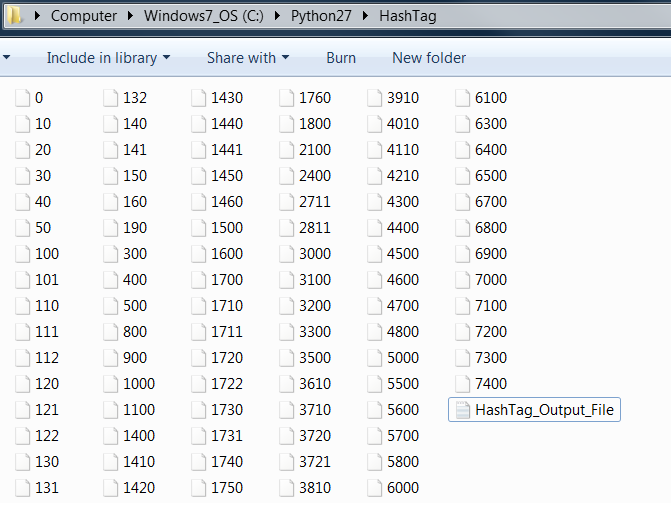
HashTag.py -f hc-hashes -hc



Output:

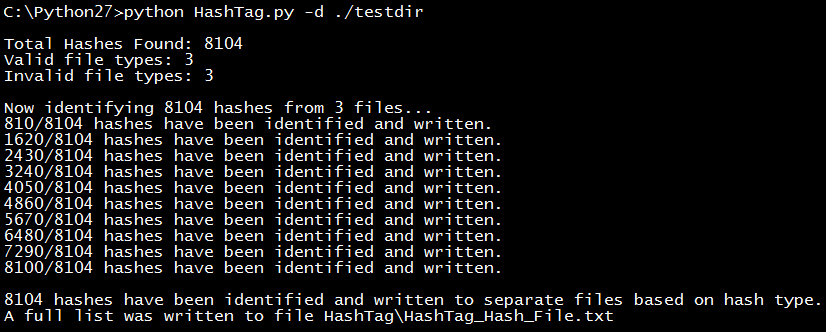


Again, using the -hc/--hashcat argument we get a file for each hash type if a corresponding hashcat mode is found. This is useful if you have a large file with different types of hashes.



**Traversing Directories and Identifying Hashes**

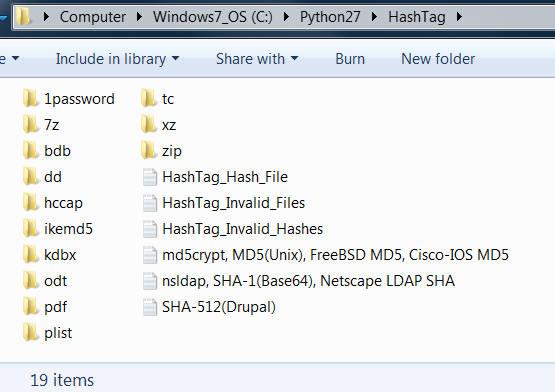
HashTag.py -f testdir\street-hashes.10.txt -hc



Output:

There are three main things included in the output:

* Folders containing copies of potentially password protected files. This makes it easy to group files based on extension and attempt to crack them.
* HashTag default files - List of all hashes, password protected files the tool doesn’t recognize, and hashes the tool can’t identify (good for tool debugging).
* Files for each identified hash type - each file contains a list of hashes. The -hc/--hashcat argument will append the hashcat mode (if found) to the filename.



**Resources**

Quite a bit of research went into different password hash types. During this research I found a script called Hash Identifier which was actually included in one of the Backtrack versions. After looking it over I feel my tool has a lot more functionality, efficiency, and accuracy. My other research ranged from finding different hash examples to generating my own via passlib. I would like to give credit to the following resources which all had some impact in the creation of this tool.

<http://wiki.insidepro.com/index.php/Main_Page>

<https://hashcat.net/wiki/>

<http://openwall.info/wiki/john/>

<http://pythonhosted.org/passlib/index.html>