KHracker – Known Hosts Entry Decrypter

Written by Kevin Moore (kevinm@cert.org) & Matthew Geiger (mgeiger@cert.org)

Requirements:

Python v2.6 or greater
Netaddr python module (http://code.google.com/p/netaddr)

Application Details:

KHracker - Known Hosts Entry Decrypter

KHracker is a python-based decryption tool for encrypted known_hosts entries. It will attempt to decrypt values stored in SSH known_hosts files, if the encryption option has been enabled for that computer. By default, known_hosts entries are not encrypted, but there is an option to do so. From a forensics perspective, encrypted known_hosts entries can prevent an investigator from seeing other computers a user may have been connecting to. Information about the connections made from a system can be integral to identifying a complete understanding of the systems involved in a network intrusion or incident response case.

***Known_hosts Entry Encryption**:*

Information on hostnames and public keys used in Secure Shell (SSH) connections are stored in what is called the known_hosts file. On Linux/Unix systems information on the computer(s) connected to via SSH is maintained in the user directory under: *~/.ssh/known_hosts*.

SSH allows for the encryption of known_hosts entry values using OpenSSL. The values, which indicate the computers a user connected to via SSH, are then obfuscated to hide their original value. The mechanism used to encrypt/obfuscate the domain or IP address is a SHA1 HMAC (Hash-based Message Authentication Code) of a salt value combined with the domain/IP address value. The salt and SHA1 HMAC encrypted value are both base64 encoded and stored in the known_hosts file. The values in an encrypted known_hosts file will appear similar to the following:

```
|1|zGIqIiAMo03vd9KZA7OQR4oeXgc=|Ar+irD9Q1WQ3UTCvc92hLF3q5+E= ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA[removed]
```

Values in the file are separated by the pipe (|) value. The section highlight in blue is the salt value for this entry. The section highlighted in red is the SHA1 HMAC encrypted value of salt combined with the domain/IP address for the connection entry. Both values are base64 encoded and stored as an entry in the known_hosts file. The value following the SHA1 HMAC value is the public key [truncated] used for the SSH connection.

KHracker Features:

KHracker attempts to decrypt values stored in the target known_hosts file by comparing SHA1 HMAC values to those stored in the known_hosts file. The script provides three options for decrypting the values: brute-force, input-file list, and bulk_extractor export processing.

Brute-force attack: attempts all, or a specified range, of IPv4 addresses. Brute-force attacks support multiprocessing to increase the number of entries attempted per second. By default, the script will utilize all available CPU cores, but that can be limited if the user specifies. Brute-forcing is the slowest of the options, but will test every valid IP address. The user can also limit the IP addresses attempted based on the command line options specified (see help output).

Input file list: Because brute-force attacks may be unfruitful or an attacker may have connected to a system via a hostname rather than an IP address, using a list containing domain or IP addresses of interest may be the most efficient and beneficial mode of attack. The user can generate a list of domains/IP addresses from a disk image or network traffic and attempt each value against those stored in the known_hosts file. This is a much quicker testing mechanism and does not support multiprocessing.

bulk_extractor Output: bulk_extractor is a tool developed by Simson Garfinkel (http://afflib.org/software/bulk_extractor). From the bulk_extractor website: "bulk_extractor is a C++ program that scans a disk image, a file, or a directory of files and extracts useful information without parsing the file system or file system structures." One of the features of bulk_extractor is the collection of domain, IP address and TCP connection information. If the user has run bulk_extractor against an image, the features of KHracker allow the user to point the script at the bulk_extractor export folder. KHracker will then generate a list of domain/IP addresses to test against the encrypted values in the known_hosts file. This method also does not support multiprocessing.

KHracker Usage:

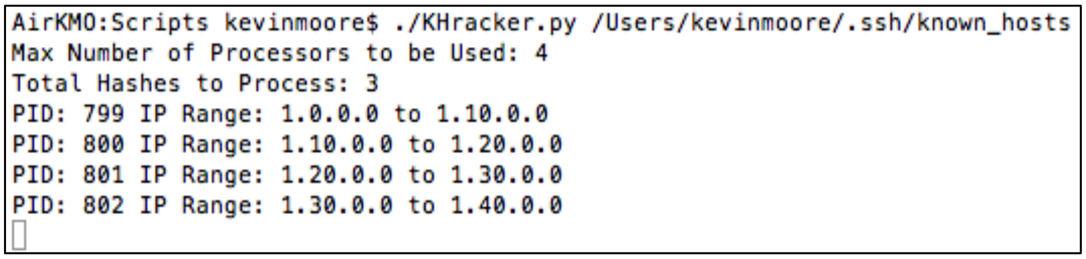
You must first install *netaddr* (http://code.google.com/p/netaddr) and Python v2.6 or greater in your operating system of choice.

To make KHracker executable in Linux/Unix/Mac run the following command:
```
chmod +x KHracker.py
```

To view KHracker's options run the help command at the command line:
```
./KHracker.py –h
```
This will display the options available and expected syntax of the application.

To run a brute-force attack on all standard, non-reserved IP addresses, run the following command and you should see output similar to that pictured below:

```
./KHracker.py <known_hosts_file_location>
```

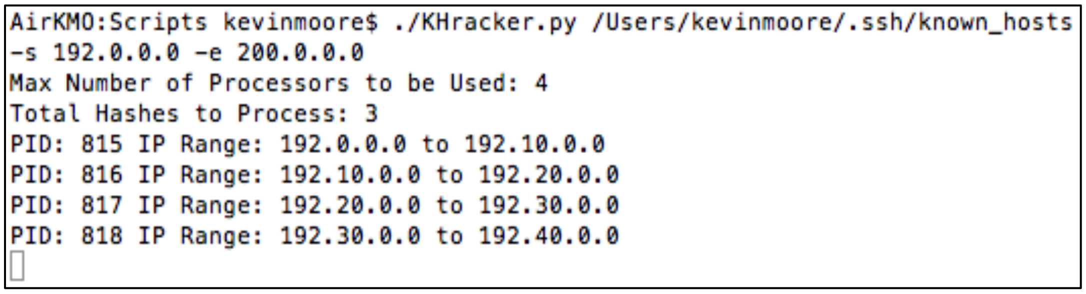


```
AirKMO:Scripts kevinmoore$ ./KHracker.py /Users/kevinmoore/.ssh/known_hosts
Max Number of Processors to be Used: 4
Total Hashes to Process: 3
PID: 799 IP Range: 1.0.0.0 to 1.10.0.0
PID: 800 IP Range: 1.10.0.0 to 1.20.0.0
PID: 801 IP Range: 1.20.0.0 to 1.30.0.0
PID: 802 IP Range: 1.30.0.0 to 1.40.0.0
```

Notice that the script is using 4 processors. This indicates that the computer running the application has 4 CPU cores that will each be utilized for the brute-force attack on the encrypted entries.

To run a brute-force attack on an IP range, specify the starting IP address and ending IP address with the –s and –e options respectively:

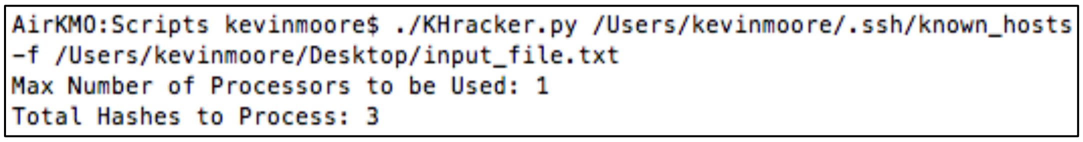
```
./KHracker.py <known_hosts_file_location> -s <starting_IP>
-e <ending_IP>
```



```
AirKMO:Scripts kevinmoore$ ./KHracker.py /Users/kevinmoore/.ssh/known_hosts
-s 192.0.0.0 -e 200.0.0.0
Max Number of Processors to be Used: 4
Total Hashes to Process: 3
PID: 815 IP Range: 192.0.0.0 to 192.10.0.0
PID: 816 IP Range: 192.10.0.0 to 192.20.0.0
PID: 817 IP Range: 192.20.0.0 to 192.30.0.0
PID: 818 IP Range: 192.30.0.0 to 192.40.0.0
```

To run an input file attack on a list of IP addresses and domain names, use the –f option and specify the input to read entries from (one entry per line):

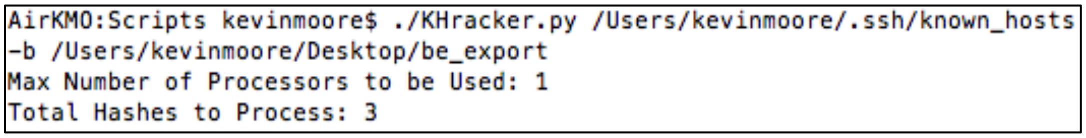
```
./KHracker.py <known_hosts_file_location> -f <input_file>
```



```
AirKMO:Scripts kevinmoore$ ./KHracker.py /Users/kevinmoore/.ssh/known_hosts
-f /Users/kevinmoore/Desktop/input_file.txt
Max Number of Processors to be Used: 1
Total Hashes to Process: 3
```

To run an attack using the output from bulk_extractor, use the –b option and specify the folder containing bulk_extractor output. The script will process the relevant output files and generate a list to test values with:

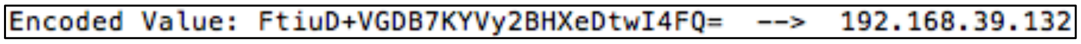
```
./KHracker.py <known_hosts_file_location>
-b <bulk_extractor_export_folder>
```



```
AirKMO:Scripts kevinmoore$ ./KHracker.py /Users/kevinmoore/.ssh/known_hosts
-b /Users/kevinmoore/Desktop/be_export
Max Number of Processors to be Used: 1
Total Hashes to Process: 3
```

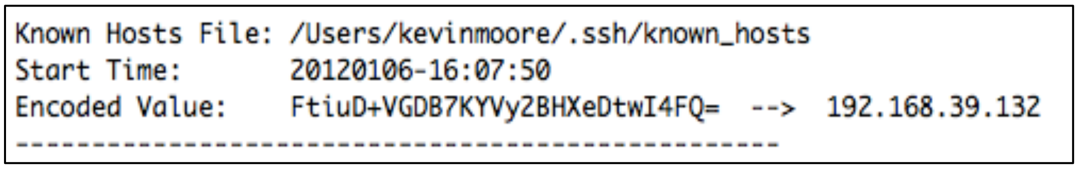
Notice that both the bulk_extractor and input_file options utilize only one CPU core. These options do not support multiprocessing in the current release.

If an entry is identified, it will display the value to the screen, similar to the following:



```
Encoded Value: FtiuD+VGDB7KYVy2BHXeDtwI4FQ=  -->  192.168.39.132
```

Decrypted entries will also be saved in a file named '*decrypt_log.txt*' stored in the same location as where KHracker was run from. The image below shows an entry in the *decrypt_log.txt* file.



```
Known Hosts File: /Users/kevinmoore/.ssh/known_hosts
Start Time:       20120106-16:07:50
Encoded Value:    FtiuD+VGDB7KYVy2BHXeDtwI4FQ=  -->  192.168.39.132
-------------------------------------------------
```

Logging entries can be disabled using command line options for sensitive situations where maintaining a log of entries is not preferred. While this option is available, it is not recommended as it could result in finding an entry, but not maintaining a record of it.

KHracker Help Output:

```
Brute-Force IP Addresses:
   KHracker.py <known_hosts_file> [options]
Read Input File of IPs/Hostnames:
   KHracker.py <known_hosts_file> [options] --read-file=<input_file>
Read Bulk Extractor Export Folder:
   KHracker.py <known_hosts_file> [options] --bulk-extractor=<input_folder>

Known Hosts Entry Decrypter. Copyright (c) CERT, 2012 ALL RIGHTS RESERVED.
Written by Kevin Moore (kevinm@cert.org) and Matthew Geiger (mgeiger@cert.org)

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -b BULK, --bulk-extractor=BULK
                        Bulk Extractor export folder for parsing IP/Hostname
                        from output files
  -c CORES, --cores=CORES
                        Number of CPU Cores to Use - Default is ALL on system.
                        Not for use with Input File or Bulk Extractor Data,
                        Brute Force Only
  -d, --disable-log     Disable decrypted data output log.
                        Console output only - NOT RECOMMENDED
  -e END_IP, --end-ip=END_IP
                        Ending IP Address for Brute-Force attacks
  -f READ_FILE, --read-file=READ_FILE
                        Processes IP Addresses/Hostnames from list in file
  -p PORTS, --ports=PORTS
                        Processes IP/Hostname with Port Numbers
                        (i.e. 192.168.1.1:1000)
                        Enter Range or List (i.e. 1-1024,6666,7777).
                        Leave blank for all ports (1-65535) -
                        SIGNIFICANTLY INCREASES PROCESSING TIME!
  -s START_IP, --start-ip=START_IP
                        Starting IP Address for Brute-Force attacks

  IP Address Specifications:
    Options for filtering IP addresses during Brute-Force processing ONLY
    -i, --private       Exclude private IP addresses
                        (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16)
    -k, --link-local    Include link local addresses (169.254.0.0/16)
    -l, --loopback      Include Loopback address space (127.0.0.0/8)
    -m, --multicast     Include multicast IP addresses
                        (224.0.0.0 to 240.0.0.0)
    -r, --reserved      Include reserved IPs addresses (over 240.0.0.0)
```