Basic Tools

Tools such as SSH, Netcat, Tmux, and Vim are essential and are used daily by most information security professionals. Although these tools are not intended to be penetration testing tools, they are critical to the penetration testing process, so we must master them.

Using SSH

Secure Shell (SSH) is a network protocol that runs on port 22 by default and provides users such as system administrators a secure way to access a computer remotely. SSH can be configured with password authentication or passwordless using public-key authentication using an SSH public/private key pair. SSH can be used to remotely access systems on the same network, over the internet, facilitate connections to resources in other networks using port forwarding/proxying, and upload/download files to and from remote systems.

SSH uses a client-server model, connecting a user running an SSH client application such as OpenSSH to an SSH server. While attacking a box or during a real-world assessment, we often obtain cleartext credentials or an SSH private key that can be leveraged to connect directly to a system via SSH. An SSH connection is typically much more stable than a reverse shell connection and can often be used as a "jump host" to enumerate and attack other hosts in the network, transfer tools, set up persistence, etc. If we obtain a set of credentials, we can use SSH to login remotely to the server by using the username @ the remote server IP, as follows:

```
MichaelLuka@htb[/htb]$ ssh Bob@10.10.10.10

Bob@remotehost's password: *********

Bob@remotehost#
```

It is also possible to read local private keys on a compromised system or add our public key to gain SSH access to a specific user, as we'll discuss in a later section. As we can see, SSH is an excellent tool for securely connecting to a remote machine. It also provides a way for mapping local ports on the remote machine to our localhost, which can become handy at times.

Using Netcat

Netcat, ncat, or nc, is an excellent network utility for interacting with TCP/UDP ports. It can be used for many things during a pentest. Its primary usage is for connecting to shells, which we'll discuss later in this module. In addition to that, netcat can be used to connect to any listening port and interact with the service running on that port. For example, SSH is programmed to handle connections over port 22 to send all data and keys. We can connect to TCP port 22 with netcat:

```
MichaelLuka@htb[/htb]$ netcat 10.10.10.10 22

SSH-2.0-OpenSSH_8.4p1 Debian-3
```

As we can see, port 22 sent us its banner, stating that SSH is running on it. This technique is called Banner Grabbing, and can help identify what service is running on a particular port. Netcat comes pre-installed in most Linux distributions. We can also download a copy for Windows machines from this link. There's another Windows alternative to netcat coded in PowerShell called PowerCat. Netcat can also be used to transfer files between machines, as we'll discuss later.

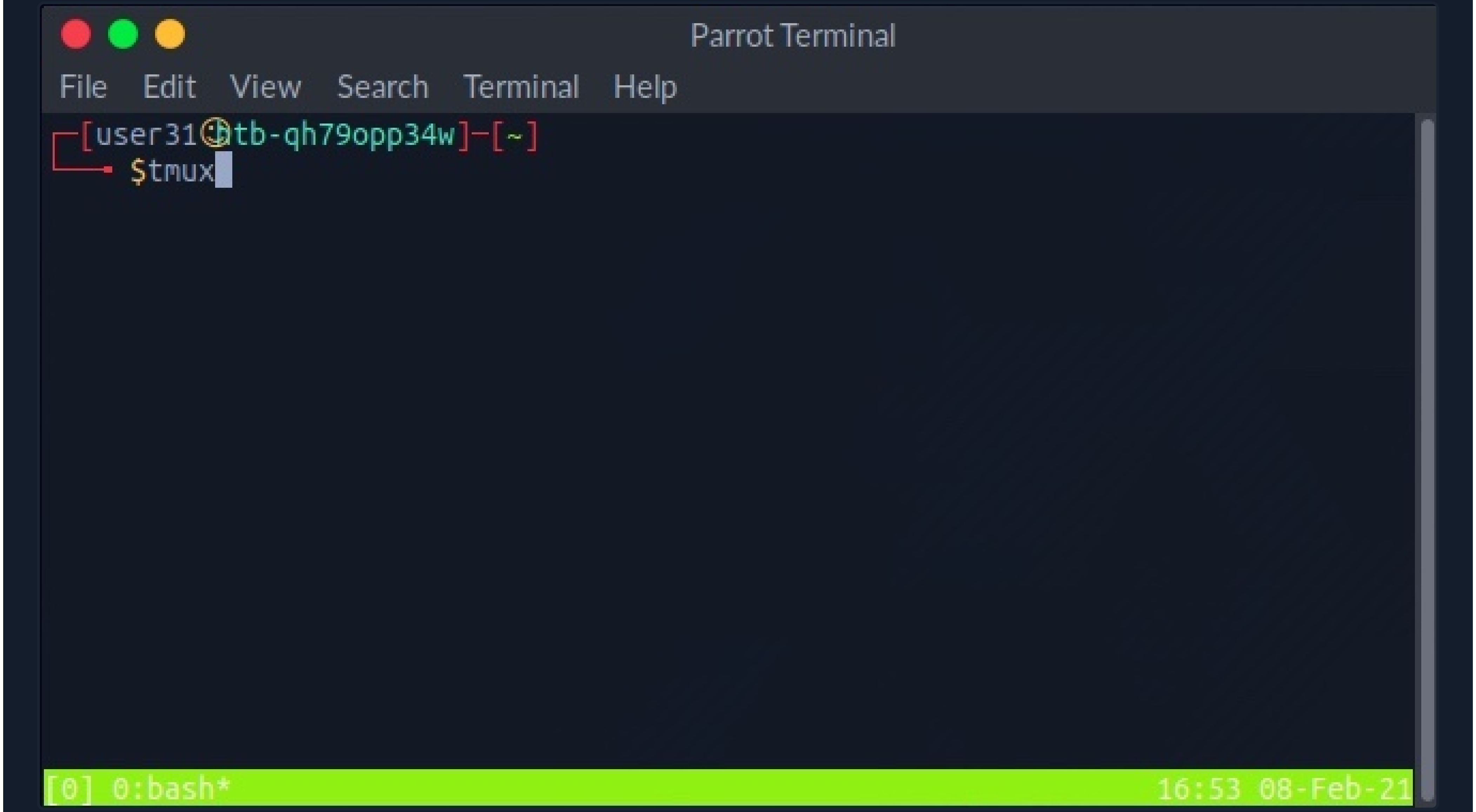
Another similar network utility is socat, which has a few features that netcat does not support, like forwarding ports and connecting to serial devices. Socat can also be used to upgrade a shell to a fully interactive TTY. We will see a few examples of this in a later section. Socat is a very handy utility that should be a part of every penetration tester's toolkit. A standalone binary of Socat can be transferred to a system after obtaining remote code execution to get a more stable reverse shell connection.

Using Tmux

Terminal multiplexers, like tmux or Screen, are great utilities for expanding a standard Linux terminal's features, like having multiple windows within one terminal and jumping between them. Let's see some examples of using tmux, which is the more common of the two. If tmux is not present on our Linux system, we can install it with the following command:

```
MichaelLuka@htb[/htb]$ sudo apt install tmux -y
```

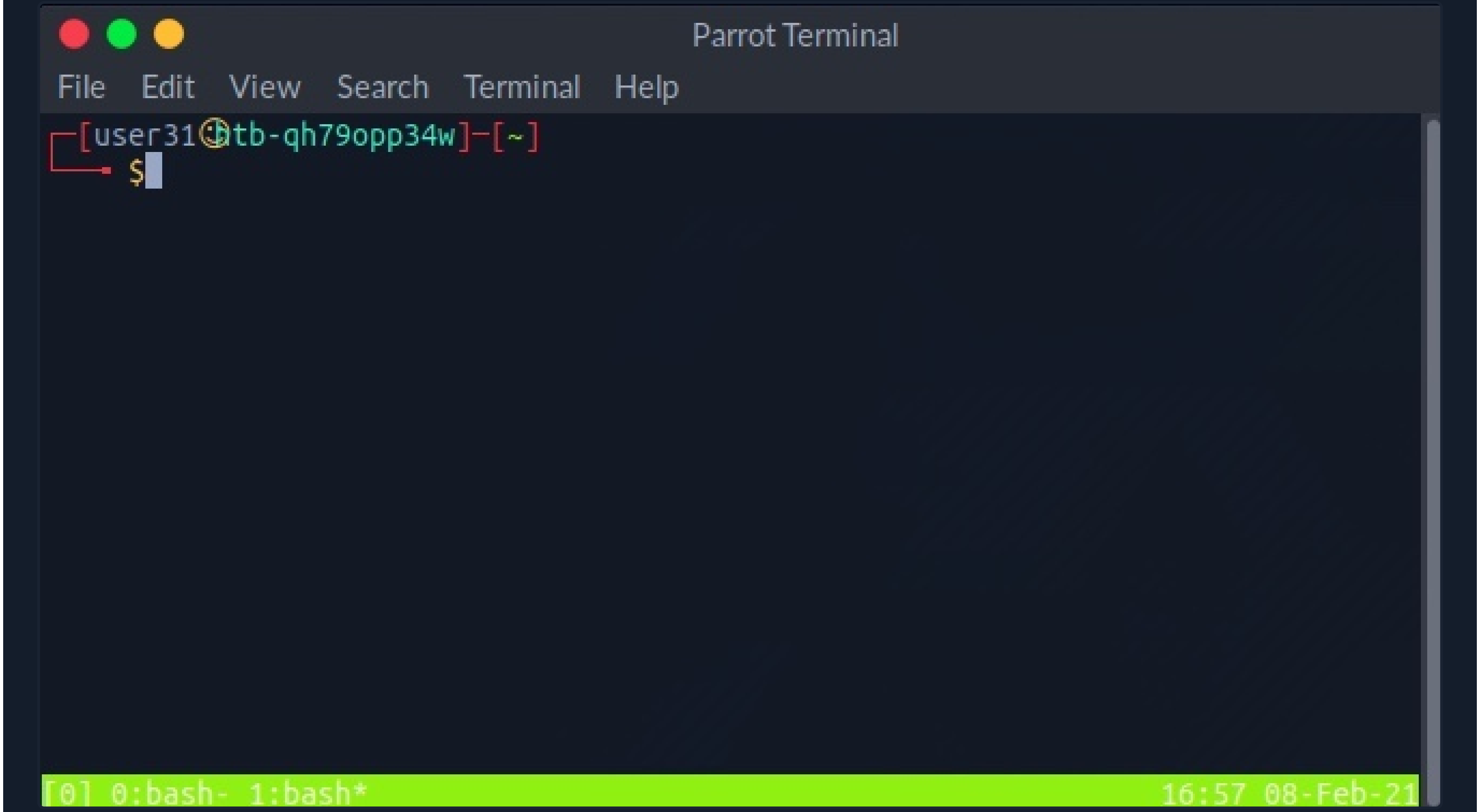
Once we have tmux, we can start it by entering tmux as our command:



```
┌─[user31@htb-qh79opp34w]─[~]
└──╼ $tmux
```

```
[0] 0:bash*                                                    16:53 08-Feb-21
```

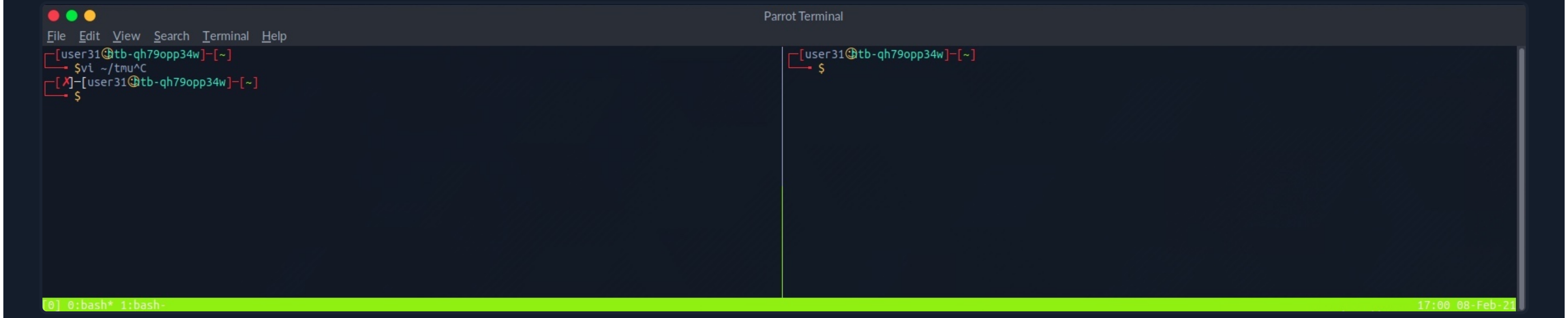
The default key to input `tmux` commands prefix is `[CTRL + B]`. In order to open a new window in `tmux`, we can hit the prefix 'i.e. `[CTRL + B]`' and then hit `C`:



```
┌─[user31@htb-qh79opp34w]─[~]
└──╼ $
```

```
[0] 0:bash- 1:bash*                                                16:57 08-Feb-21
```

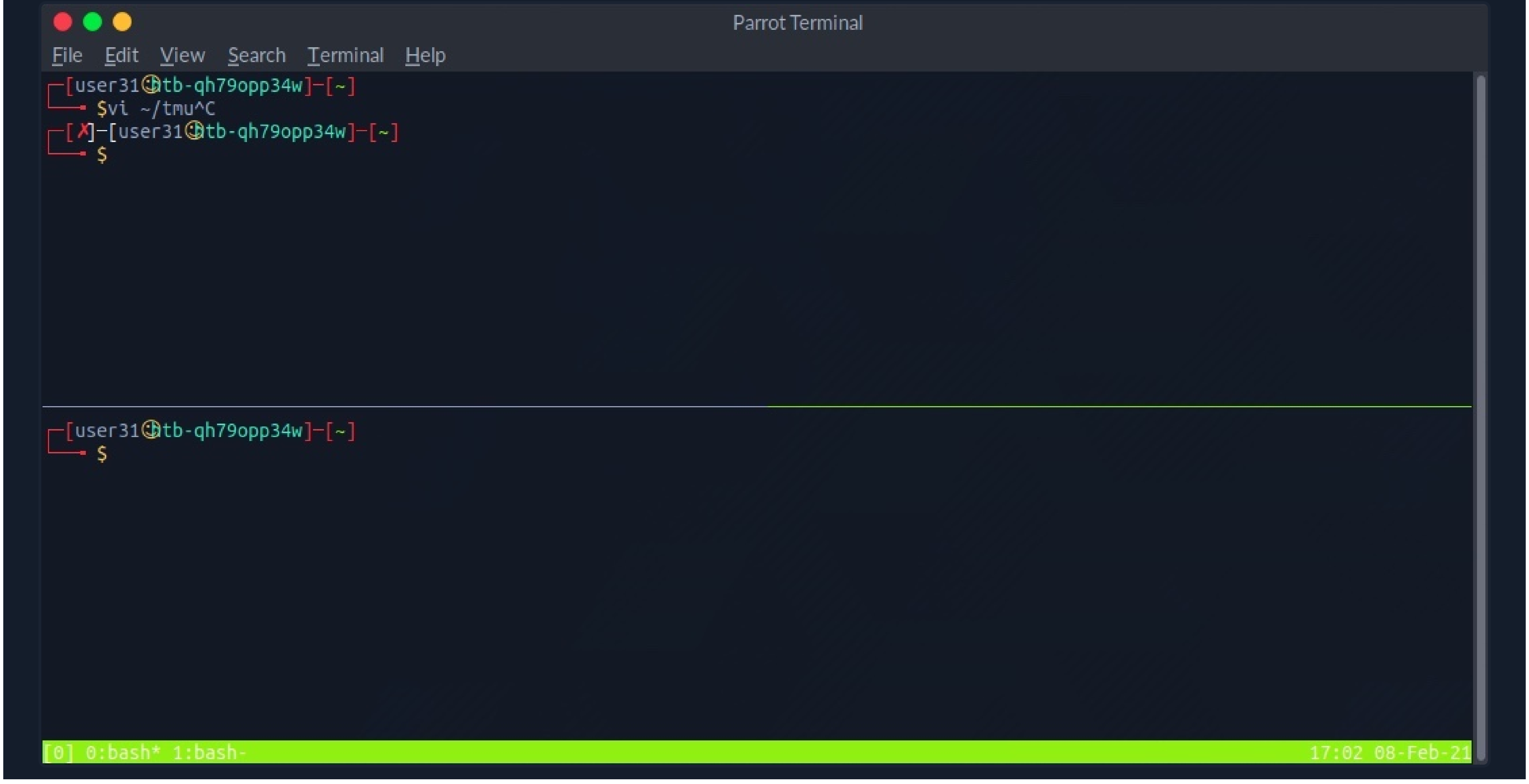
We see the numbered windows at the bottom. We can switch to each window by hitting the prefix and then inputting the window number, like `0` or `1`. We can also split a window vertically into panes by hitting the prefix and then `[SHIFT + %]`:



```
┌─[user31@htb-qh79opp34w]─[~]              │┌─[user31@htb-qh79opp34w]─[~]
└──╼ $vi ~/tmu^C                           │└──╼ $
┌─[✗]─[user31@htb-qh79opp34w]─[~]          │
└──╼ $                                     │
```

```
[0] 0:bash* 1:bash-                                                17:00 08-Feb-21
```

We can also split into horizontal panes by hitting the prefix and then `[SHIFT + "]`:



```
┌─[user31@htb-qh79opp34w]─[~]
└──╼ $vi ~/tmu^C
┌─[✗]─[user31@htb-qh79opp34w]─[~]
└──╼ $
────────────────────────────────────────
┌─[user31@htb-qh79opp34w]─[~]
└──╼ $
```

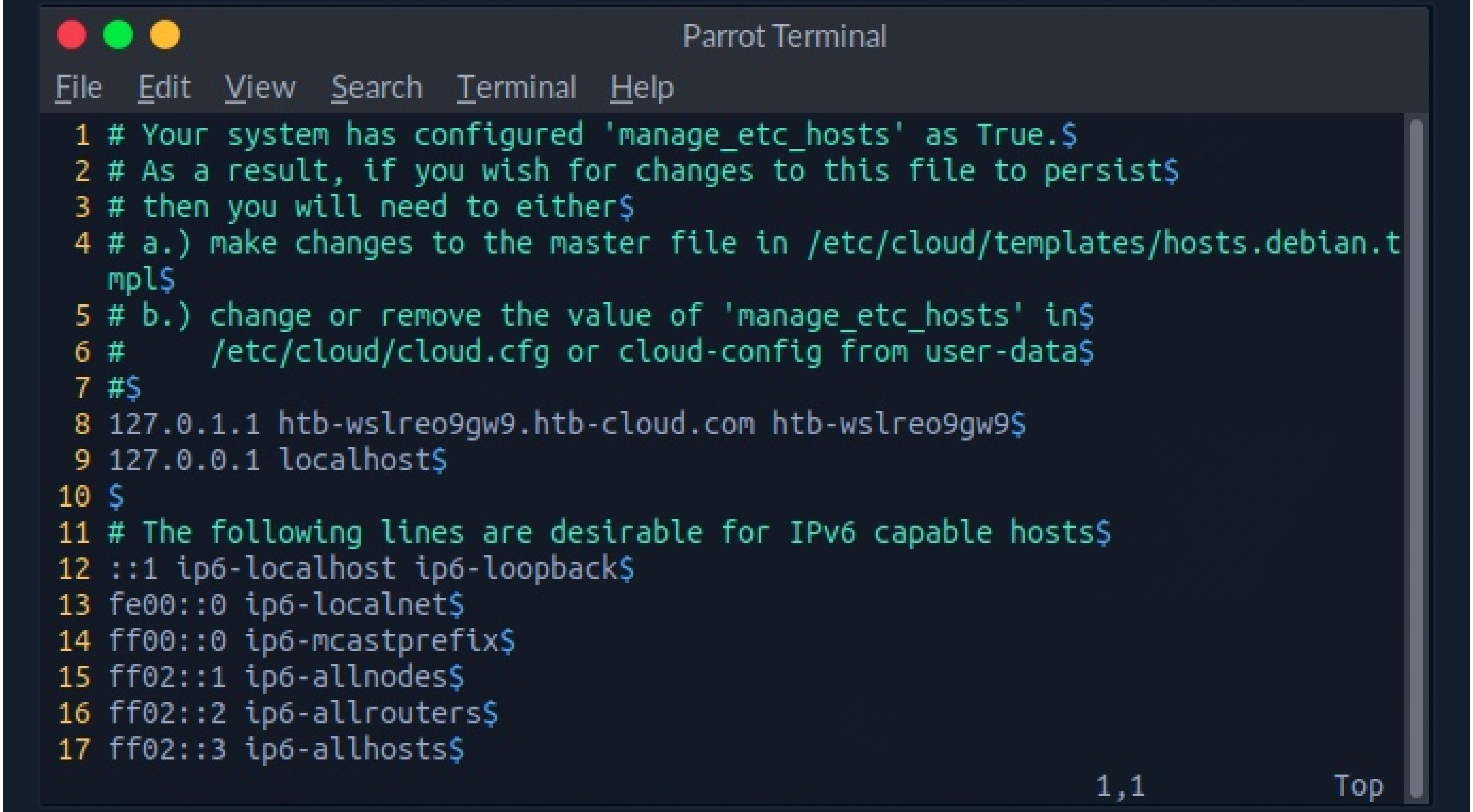
```
[0] 0:bash* 1:bash-                                                17:02 08-Feb-21
```

We can switch between panes by hitting the prefix and then the `left` or `right` arrows for horizontal switching or the `up` or `down` arrows for vertical switching. The commands above cover some basic `tmux` usage. It is a powerful tool and can be used for many things, including logging, which is very important during any technical engagement. This cheatsheet is a very handy reference. Also, this Introduction to tmux video by `ippsec` is worth your time.

Using Vim

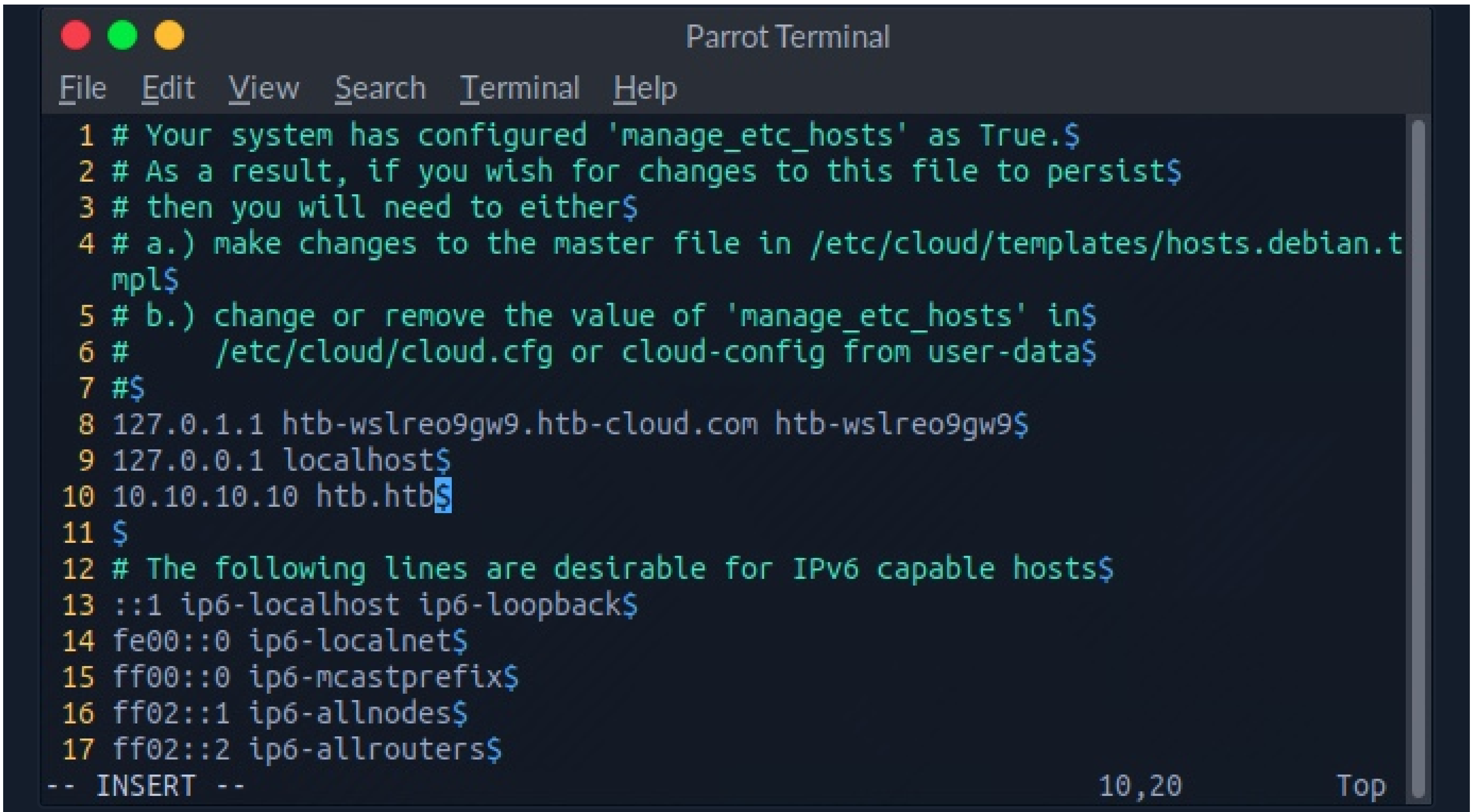
Vim is a great text editor that can be used for writing code or editing text files on Linux systems. One of the great benefits of using `Vim` is that it relies entirely on the keyboard, so you do not have to use the mouse, which (once we get the hold of it) will significantly increase your productivity and efficiency in writing/editing code. We usually find `Vim` or `Vi` installed on compromised Linux systems, so learning how to use it allows us to edit files even on remote systems. `Vim` also has many other features, like extensions and plugins, which can significantly extend its usage and make for a great code editor. Let's see some of the basics of `Vim`. To open a file with `Vim`, we can add the file name after it:

```
MichaelLuka@htb[/htb]$ vim /etc/hosts
```



```
# Your system has configured 'manage_etc_hosts' as True.$
# As a result, if you wish for changes to this file to persist$
# then you will need to either$
# a.) make changes to the master file in /etc/cloud/templates/hosts.debian.t
mpl$
# b.) change or remove the value of 'manage_etc_hosts' in$
#     /etc/cloud/cloud.cfg or cloud-config from user-data$
#$
127.0.1.1 htb-wslreo9gw9.htb-cloud.com htb-wslreo9gw9$
127.0.0.1 localhost$
$
# The following lines are desirable for IPv6 capable hosts$
::1 ip6-localhost ip6-loopback$
fe00::0 ip6-localnet$
ff00::0 ip6-mcastprefix$
ff02::1 ip6-allnodes$
ff02::2 ip6-allrouters$
ff02::3 ip6-allhosts$
                                                   1,1           Top
```

If we want to create a new file, input the new file name, and `Vim` will open a new window with that file. Once we open a file, we are in read-only `normal mode`, which allows us to navigate and read the file. To edit the file, we hit `i` to enter `insert mode`, shown by the "`-- INSERT --`" at the bottom of `Vim`. Afterward, we can move the text cursor and edit the file:



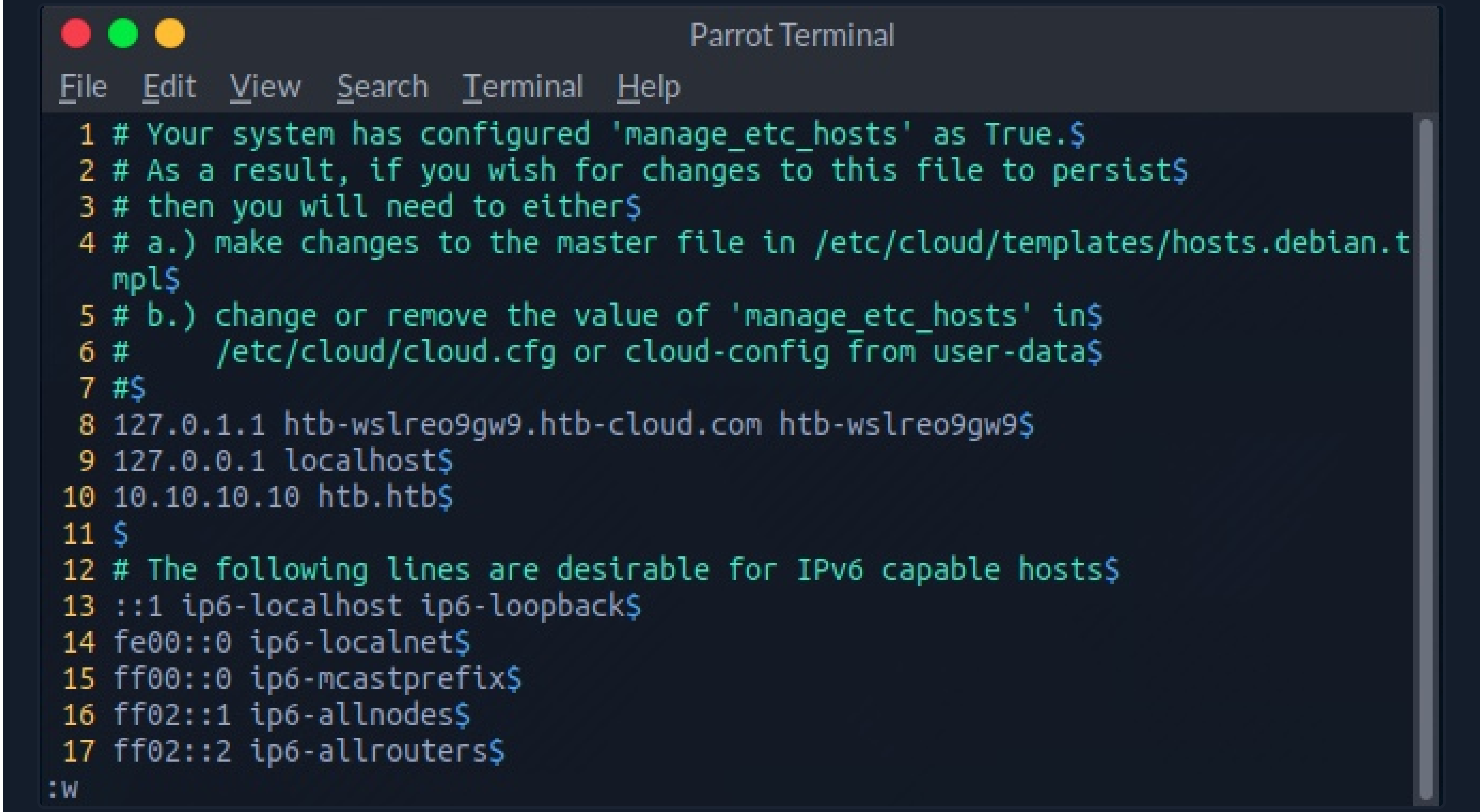
```
# Your system has configured 'manage_etc_hosts' as True.$
# As a result, if you wish for changes to this file to persist$
# then you will need to either$
# a.) make changes to the master file in /etc/cloud/templates/hosts.debian.t
mpl$
# b.) change or remove the value of 'manage_etc_hosts' in$
#     /etc/cloud/cloud.cfg or cloud-config from user-data$
#$
127.0.1.1 htb-wslreo9gw9.htb-cloud.com htb-wslreo9gw9$
127.0.0.1 localhost$
10.10.10.10 htb.htb$
$
# The following lines are desirable for IPv6 capable hosts$
::1 ip6-localhost ip6-loopback$
fe00::0 ip6-localnet$
ff00::0 ip6-mcastprefix$
ff02::1 ip6-allnodes$
ff02::2 ip6-allrouters$
-- INSERT --                                           10,20         Top
```

Once we are finished editing a file, we can hit the escape key `esc` to get out of `insert mode`, back into `normal mode`. When we are in `normal mode`, we can use the following keys to perform some useful shortcuts:

Command	Description
`x`	Cut character
`dw`	Cut word
`dd`	Cut full line
`yw`	Copy word
`yy`	Copy full line
`p`	Paste

Tip: We can multiply any command to run multiple times by adding a number before it. For example, '4yw' would copy 4 words instead of one, and so on.

If we want to save a file or quit `Vim`, we have to press `:` to go into `command mode`. Once we do, we will see any commands we type at the bottom of the vim window:



```
# Your system has configured 'manage_etc_hosts' as True.$
# As a result, if you wish for changes to this file to persist$
# then you will need to either$
# a.) make changes to the master file in /etc/cloud/templates/hosts.debian.t
mpl$
# b.) change or remove the value of 'manage_etc_hosts' in$
#     /etc/cloud/cloud.cfg or cloud-config from user-data$
#$
127.0.1.1 htb-wslreo9gw9.htb-cloud.com htb-wslreo9gw9$
127.0.0.1 localhost$
10.10.10.10 htb.htb$
$
# The following lines are desirable for IPv6 capable hosts$
::1 ip6-localhost ip6-loopback$
fe00::0 ip6-localnet$
ff00::0 ip6-mcastprefix$
ff02::1 ip6-allnodes$
ff02::2 ip6-allrouters$
:w
```

There are many commands available to us. The following are some of them:

Command	Description
`:1`	Go to line number 1.
`:w`	Write the file, save
`:q`	Quit
`:q!`	Quit without saving
`:wq`	Write and quit

`Vim` is a very powerful tool and has many other commands and features. This cheatsheet is an excellent resource for further unlocking the power of `Vim`.

Start Instance

1 / 1 spawns left

Waiting to start...

Optional Exercises

Challenge your understanding of the Module content and answer the optional question(s) below. These are considered supplementary content and are not required to complete the Module. You can reveal the answer at any time to check your work.

Target: Click here to spawn the target system!

Apply what you learned in this section to grab the banner of the above server and submit it as the answer.

Submit your answer here...

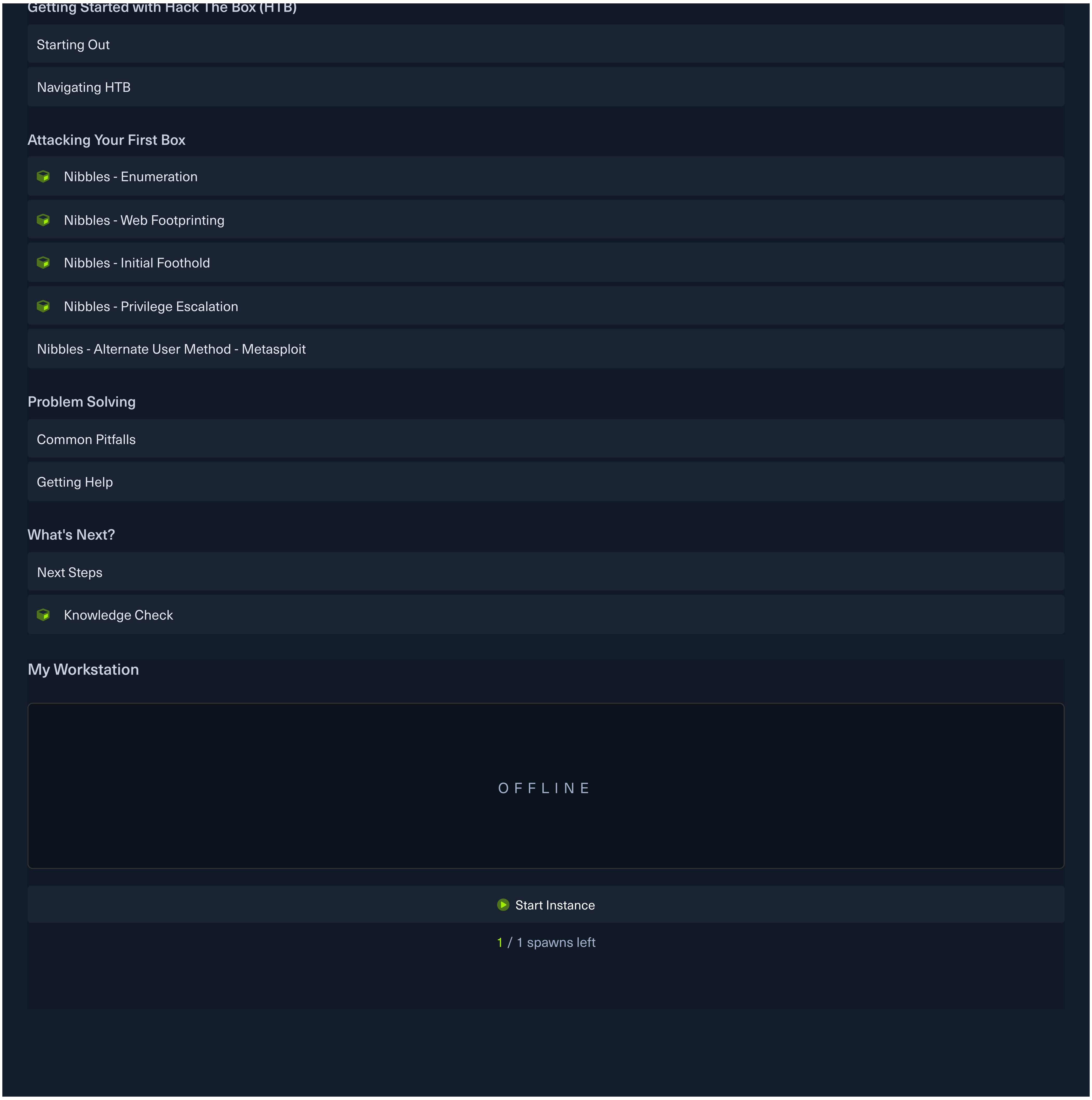
Submit

Reveal Answer

Previous   Next   Mark Complete & Next

Cheat Sheet

Table of Contents



Getting Started with Hack The Box (HTB)

- Starting Out
- Navigating HTB

Attacking Your First Box

- Nibbles - Enumeration
- Nibbles - Web Footprinting
- Nibbles - Initial Foothold
- Nibbles - Privilege Escalation
- Nibbles - Alternate User Method - Metasploit

Problem Solving

- Common Pitfalls
- Getting Help

What's Next?

- Next Steps
- Knowledge Check

My Workstation

OFFLINE

▶ Start Instance

1 / 1 spawns left