SSH - Brute Force Detection using Bro Network inside a Vagrant Virtual Environment

Christopher De Jesus
christopher.dejesus@upr.edu
Computer Science Department
University of Puerto Rico - Río Piedras

Advisor:
Humberto Ortiz-Zuazaga
humberto@hpcf.upr.edu
Computer Science Department
University of Puerto Rico - Río Piedras

Abstract

Secure Socket Shell, or SSH, is a network protocol that provides administrators with a secure way to access a remote computer. Secure Shell provides strong authentication and secure encrypted data communications between two computers connecting over an insecure network such as the Internet. SSH is widely used by network administrators for managing systems and applications remotely, allowing them to log in to another computer over a network, execute commands and move files from one computer to another.

The main tool used for this research is called Bro Network. Bro is a passive, open-source network traffic analyzer. It is primarily a security monitor that inspects all traffic on a link in depth for signs of suspicious activity. More generally, however, Bro supports a wide range of traffic analysis tasks even outside of the security domain, including performance measurements and helping with trouble-shooting.

1 Introduction

For this research, the main approach was to install inside a virtual environment the Bro Monitoring System and set up, using Vagrant, a main node which will be the one monitoring its leaves. After setting this up, then create a Python-based script using the module Paramiko to SSH-Brute Force the leaves. Then, the main node should be able to detect this, with the help of TCPDump, and create a log with all the SSH tries to later on take the necessary steps to deal with this problem.

2 Methodology and Tools Used

The tools used for this research were: TCPDump, Python, Paramiko, Vagrant and Bro Monitoring System

1. The first approach used for this research was to install the Bro Monitoring System inside Hulk. This was a complete fail at first, because the install for this IDS needs root access which can be dangerous when experimenting in a server that it is used daily by other researchers.

2. The second approach of installation was to use a personal MacBook Pro. While installing this inside a personal machine, it only installed most of the files and could not execute the monitoring when using 'Broctl', the application inside Bro that monitors the workers, the proxy and the children nodes.

3. The last approach for installation was to use Vagrant. Vagrant is a tool for building complete development environments created by Mitchell Hashimoto in 2010. When using this, we had to set up the VagrantFile that configures how many virtual environments it will be running and the hierarchy of each of these nodes. Dividing them by main node, proxy node and children node. Figure 1 shows how the VagrantFile configures a multi-machine, each with different private IP addresses.

Figure 1: Configuration file provided by Vagrant

```
Vagrant.configure(2) do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.
#Multi-Machine config
  config.vm.define "padre" do |padre|
    padre.vm.box = "ubuntu/trusty64"
    padre.vm.network "private_network", ip: "10.1.1.10"
end

 # config.vm.define "hijo" do |hijo|
  #  hijo.vm.box = "ubuntu/trusty64"
  #  hijo.vm.network "private_network", ip: "10.2.2.11"
 # end

end
```

4. Vagrant was installed successfully with 3 environments running Ubuntu 12.04 LTS 64-bit. The purpose of this was to designate a main node with 2 children and install Bro Network inside this main node and monitor the 2 children. To use this we just needed to do "Vagrant up" and all the machines will be ready to access and then "Vagrant ssh *" where * is the

name of the machine you want to SSH. Figure 2 shows the process used inside the personal machine.

Figure 2: Starting up Vagrant

```
ChristohersMBP2:bro Christopher$ vagrant up
Bringing machine 'padre' up with 'virtualbox' provider...
==> padre: Checking if box 'ubuntu/trusty64' is up to date...
==> padre: A newer version of the box 'ubuntu/trusty64' is available! You currently
==> padre: have version '20160406.0.0'. The latest is version '20160509.1.0'. Run
==> padre: `vagrant box update` to update.
==> padre: Resuming suspended VM...
==> padre: Booting VM...
==> padre: Waiting for machine to boot. This may take a few minutes...
    padre: SSH address: 127.0.0.1:2222
    padre: SSH username: vagrant
    padre: SSH auth method: private key
==> padre: Machine booted and ready!
==> padre: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> padre: flag to force provisioning. Provisioners marked to run always will still run.
ChristohersMBP2:bro Christopher$ vagrant ssh
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-79-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

 System information disabled due to load higher than 1.0

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

22 packages can be updated.
10 updates are security updates.


Last login: Thu May  5 19:39:58 2016 from vagrant-ubuntu-trusty-64
vagrant@vagrant-ubuntu-trusty-64:~$
```

5. The next steps will be a simple guide on how to install Bro inside the Ubuntu 12.04 LTS 64-bit.

   (a) We need to install all the dependencies. Make sure the OS is up-to-date.

   ```
   sudo apt-get install cmake make gcc g++ flex bison
   libpcap-dev
   libgeoip-dev libssl-dev
   python-dev zlib1g-dev
   libmagic-dev swig libgoogle-perftools-dev
   ```

   (b) Then we create a directory that will save the logs created by Bro:

   ```
   sudo mkdir -p /nsm/bro
   ```

   (c) For the main installation:

   ```
   cd ~
   wget https://www.bro.org/downloads/release/bro-2.4.1.tar.gz
   ```

```
tar -xvzf bro-2.4.1.tar.gz
cd bro-2.4.1
./configure --prefix=/nsm/bro
make
sudo make install
export PATH=/nsm/bro/bin:$PATH
```

(d) Now we need to set up the multi-machines by hierarchy:

```
sudo vim /nsm/bro/etc/node.cfg
```

Figure 3: Configuring the nodes

```
# Example BroControl node configuration.
#
# This example has a standalone node ready to go except for possibly changing
# the sniffing interface.

# This is a complete standalone configuration.  Most likely you will
# only need to change the interface.
#[bro]
#type=standalone
#host=localhost
#interface=eth1
#
## Below is an example clustered configuration. If you use this,
## remove the [bro] node above.
#
[manager]
type=manager
host=10.1.1.10

[proxy-1]
type=proxy
host=10.1.1.10

[worker-1]
type=worker
host=10.1.1.11
interface=eth1
#
#[worker-2]
#type=worker
#host=host3
#interface=eth0
#
#[worker-3]
#type=worker
#host=host4
#interface=eth0
```

(e) Lastly, we install BroControl for the active monitoring:

```
sudo /nsm/bro/bin/broctl
install
exit
```

Figure 4: Installing BroControl and status of the nodes

```
[BroControl] > vagrant@vagrant-ubuntu-trusty-64:~$ sudo /nsm/bro/bin/broctl

Welcome to BroControl 1.4

Type "help" for help.

[BroControl] > install
removing old policies in /nsm/bro/spool/installed-scripts-do-not-touch/site ...
removing old policies in /nsm/bro/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating cluster-layout.bro ...
generating local-networks.bro ...
generating broctl-config.bro ...
generating broctl-config.sh ...
updating nodes ...
[BroControl] > status
Getting process status ...
Getting peer status ...
Name         Type    Host             Status    Pid    Peers  Started
manager      manager 10.1.1.10        running   21010  2      05 May 19:12:41
proxy-1      proxy   10.1.1.10        running   21052  2      05 May 19:12:43
worker-1     worker  10.1.1.10        running   21089  2      05 May 19:12:44
[BroControl] >
```

6. After this, installation was successful inside the main node. Many tries later, it was noticed that the permissions when monitoring one machine from another one were getting complicated and after troubleshooting with the Bro Network team on-line, an advice given by one of the staff was to monitor the machine from itself. This was the setup that started to work when 'Broctl' was activated.

7. Now that we have a successful Bro installation inside the Vagrant environment using Ubuntu, a script using Python and the module Paramiko has to be created in order to 'attack' by SSH the child node (which is also the parent node). The script created to do the testing can be seen in Figure 6 and an example of how it works on Figure 5:

Figure 5: An example of how the Python-based script works

```
vagrant@vagrant-ubuntu-trusty-64:~/script_brute$ python attack_ssh.py
Enter Address: 10.1.1.10
Enter Username: vagrant
Which dictionary?: input_file

Na que ver
Na que ver
Na que ver
Na que ver
Na que ver
Na que ver
Na que ver
```

Figure 6: The script used for testing SSH using a dictionary attack

```
import paramiko, sys, os, socket

global host, username, line, input_file

line = "\n----------------------\n"


def ssh_connect(password, code = 0):
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        try:
                ssh.connect(host, port=22, username=username, password=password)
        except paramiko.AuthenticationException:
                code = 1
        except socket.error, e:
                code = 2
        ssh.close()
        return code

try:
        host = raw_input("Enter Address: ")
        username = raw_input("Enter Username: ")
        input_file = raw_input("Which dictionary?: ")

        if os.path.exists(input_file) == False:
                print "\n Didnt find the dictionary"
                sys.exit(4)
except KeyboardInterrupt:
        print "\n\n Interrupted"
        sys.exit(3)


input_file = open(input_file)

print ""

for i in input_file.readlines():
        password = i.strip("\n")
        try:
                response = ssh_connect(password)
                if response == 0:
                        print("%s Usuario: %s [*] Password: %s%s" %(line, username, password, line))
                elif response == 1:
                        print("Na que ver")
                elif response == 2:
                        print("No pude establecer coneccion")
                        sys.exit(2)
        except Exception, e:
                print e
                pass
input_file.close()
```

8. When the script was created, after reading the documentation of Bro Network, the approach of manually analyzing the logs was the best to go because the monitoring used by BroControl uses every script possible instead of the "detect-bruteforcing.bro" which is the script capable of analyzing the brute force using SSH. TCPDump comes to play when trying to do it manually because we need a .pcap file.

```
sudo apt-get install tcpdump
```

9. In this part, the testing actually begins. The first step is to start attacking using the script created:

```
python attack_ssh.py
Enter Address: 10.1.1.10
Enter UserName: vagrant
Which dictionary?: input_file
```

10. Later, we use this command to create a .pcap file that contains the ssh connections and the "tries":

```
sudo tcpdump -i lo -w mycap.pcap
```

In this instruction we tell the parameter *-i* to take the interface *lo* which is the local host and then create a .pcap file called "mycap.pcap".

11. Now, we need to take this .pcap file and run the detect-bruteforcing.bro script inside the directory "../protocol/ssh":

```
sudo /nsm/bro/bin/bro -Cr mycap.pcap
/nsm/bro/share/bro/policy/protocols/ssh/detect-bruteforcing.bro
```

12. After using this, we will have an "ssh.log" that contains all the brute force SSH connections based on the .pcap file created by TCPDump.

Figure 7: The output of ssh.log after the attack

```
vagrant@vagrant-ubuntu-trusty-64:~$ cat ssh.log
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   ssh
#open   2016-05-12-22-33-10
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       version auth_success    direction       client  server  cipher_alg      mac_alg compression_alg kex_alg host_key_alg    host_key
#types  time    string  addr    port    addr    port    count   bool    enum    string  string  string  string  string  string  string  string
1463092346.283506       CkTqjq1hp0cO5Db5p6      10.1.1.10       55396   10.1.1.10       22      2       -       -       SSH-2.0-paramiko_1.10.1 SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6       aes128-ctr      hmac-md5        none    diffie-hellman-group-e
xchange-sha1    ssh-rsa 71:b3:8a:78:61:71:18:39:e8:2e:c8:d3:f2:22:d6:23
1463092348.371456       C1FY8jXFFMsGtf6Sg       10.1.1.10       55397   10.1.1.10       22      2       -       -       SSH-2.0-paramiko_1.10.1 SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6       aes128-ctr      hmac-md5        none    diffie-hellman-group-e
xchange-sha1    ssh-rsa 71:b3:8a:78:61:71:18:39:e8:2e:c8:d3:f2:22:d6:23
1463092350.068688       ChP8gU2tHatmKBJBE4      10.1.1.10       55398   10.1.1.10       22      2       -       -       SSH-2.0-paramiko_1.10.1 SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6       aes128-ctr      hmac-md5        none    diffie-hellman-group-e
xchange-sha1    ssh-rsa 71:b3:8a:78:61:71:18:39:e8:2e:c8:d3:f2:22:d6:23
1463092352.043698       CdUhbZ1DjHA4pMJf0f      10.1.1.10       55399   10.1.1.10       22      2       -       -       SSH-2.0-paramiko_1.10.1 SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6       aes128-ctr      hmac-md5        none    diffie-hellman-group-e
xchange-sha1    ssh-rsa 71:b3:8a:78:61:71:18:39:e8:2e:c8:d3:f2:22:d6:23
1463092353.955561       Clqn2946HnCCYVXdGj      10.1.1.10       55400   10.1.1.10       22      2       -       -       SSH-2.0-paramiko_1.10.1 SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6       aes128-ctr      hmac-md5        none    diffie-hellman-group-e
xchange-sha1    ssh-rsa 71:b3:8a:78:61:71:18:39:e8:2e:c8:d3:f2:22:d6:23
1463092356.471740       C8uD5Ub9H7dM4hyw8       10.1.1.10       55401   10.1.1.10       22      2       -       -       SSH-2.0-paramiko_1.10.1 SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6       aes128-ctr      hmac-md5        none    diffie-hellman-group-e
xchange-sha1    ssh-rsa 71:b3:8a:78:61:71:18:39:e8:2e:c8:d3:f2:22:d6:23
#close  2016-05-12-22-33-10
vagrant@vagrant-ubuntu-trusty-64:~$
```

As we can see in Figure 7 there are references to the tools used for attacking, paramiko. This could be useful in the future to create or edit the already existing script for detecting bruteforcing and either ban, grant access, etc. to these types of tools when accessing a machine.

3 Future Work

For the next semester, the proposal is to use this data to create a machine connected to the Science DMZ and let it monitor for these types of attack. This can only be done if we can fix the problem of creating multi-machines inside Vagrant and monitor them from the main node. Also, these finds can be used to recreate the same tests when using other protocols like http, ftp, ssl, etc.

4 Conclusion

The use of Bro Network, even though it complicates simple tasks since it is mostly designed for programmers, is essential when trying to make an intrusion detection system that monitors the network traffic live. This test was done with semi-live data but the actual logs created are on the /nsm/bro/logs, which saves all the logs after using all the scripts possible. This test was created so that we can see that this script works and then edit it based on the need. The use of Vagrant was a must when trying to install an IDS like this since we want to test with specific packets without other network junk that could confuse the testing in any way.

5 Acknowledgement

This work is supported by the scholarship Academics and Training for the Advancement of Cybersecurity Knowledge in Puerto Rico (ATACK-PR) supported by the National Science Foundation under Grant No. DUE-1438838.

References

[1] The Bro Network Security Monitor

`https://www.bro.org/index.html`

[2] Vagrant

`https://www.vagrantup.com/`

[3] Paramiko

`http://www.paramiko.org/`

[4] SPLOIT: How to Make an SSH Brute-Forcer in Python

`http://null-byte.wonderhowto.com/how-to/sploit-make-ssh-brute-forcer-python-0161689/`

[5] TCPDump

`http://www.tcpdump.org/`

[6] Bro Cluster using Vagrant Issues

`http://comments.gmane.org/gmane.comp.security.detection.bro/9326`