Performance Evaluation of a Raspberry Pi Bramble Cluster for Penetration Testing

M. Ganesh [1], D. Berrios-Vasquez [1], A. Menmou [1], M. Arthur [1], A. Carranza [1], and C. DeCusatis [2]

[1]New York City College of Technology, Brooklyn, NY USA, acarranza@citytech.edu
[2]Marist College, Poughkeepsie, NY USA, Casimer.decusatis@marist.edu

***Abstract* – There is a growing need for low cost, high performance penetration testing hardware and software platforms to counteract the increasing number and severity of cyberattacks, as well as to provide education and training tools to address the significant shortage of cybersecurity professionals. In this paper, we investigate parallel computer clusters using the low cost Raspberry Pi platform as an alternative to conventional penetration testing systems. We investigate 2-node and 4 –nobe Bramble clusters for wireless password cracking using open source Kali Linux tools, and compare the performance with conventional desktop and laptop systems.**

Keywords: Raspberry Pi Cluster, Parallel Computation, Pyrit, John the Ripper, Aircrack-ng.

I. Introduction

The increasing severity of cybersecurity attacks has led to an interest in developing more low cost, efficient penetration testing platforms for practitioners and educators. There was a record of 535 data breeches in 2017, that affected over 1.9 billion records [1]. Major companies have poured billions of dollars in research and development to help create newer and stronger means of protecting data [2]. Companies are losing proprietary data and exposing their clients' personal details through these security holes [3]. Penetration testing involves various software to attack and find exploits in a computer system's security, and is used to help system designers identify problem areas within their system in order to make them more secure and harden them against attack [4]. We will investigate whether a cluster of low cost Raspberry Pi processors can conduct penetration testing using Kali Linux open source tools, and compare the performance with higher cost platforms such as laptops and desktops.

The rest of this paper is organized as follows. Following the introduction, Section II describes how to create a Bramble cluster or Raspberry Pi processors and install specific types of Linux penetration testing tools. Section III summarizes experimental results from password sniffing and cracking attempts using a 2-node and 4-node Bramble cluster. Finally, Section IV summarizes our conclusions and plans for future work.

II. Implementation of a Bramble Cluster

To begin implementing the bramble cluster, we first install the necessary software and dependencies needed. We start by installing the following package names: MPICH, Pyrit, DISPY, NMAP, Psutil, John the Ripper (JtR), and lastly the Wi-Fi adapter that is used to sniff and inject packets. MPICH is a portable and high-performance implementation of the Message Passing Interface (MPI), that allows for parallel computations between computer systems [5]. JtR is an open source password hash cracker that supports the open MPI standards and supports a wide range of hash and cipher types including but not limited to zip, rar, lanman, wpa-psk, sha1, sha256, sha512, md5, pdf, openssh, and hundreds more [6]. Aircrack-ng is a suite of networking tools for assessing Wi-Fi network security, which focuses on monitoring, attacking, testing and cracking wireless network security [7].

We will create a cluster of Raspberry Pi's suitable for penetration testing based on the MPICH framework and software's such as Aircrack-ng, John the Ripper and Pyrit. The cluster will utilize four to six Raspberry Pi's, a wired switch and a powered USB hub, connected to create the Bramble Cluster (see Figure 1). In the following subsections A to H, we will install the necessary software required to create a cluster of interlinking Raspberry Pi's and then create an environment that is consistent on each of the slave nodes and the master node to enable the sharing of a common resource.

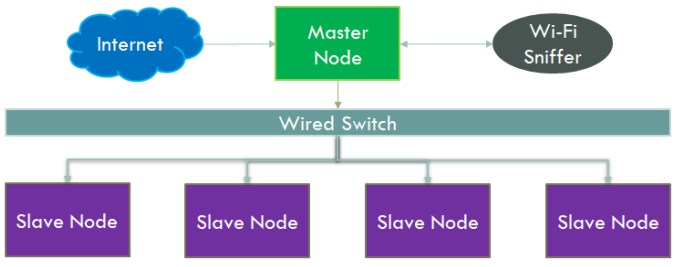

Figure 1 - Bramble Cluster

A. *MPICH Installation*

To install MPICH we first must refresh the list of packages in the cache by issuing the following command in the terminal.

```
$ sudo apt-get update
```

Next, we will make a directory to put the source code in

```
$ mkdir /home/pi/mpich3
$ cd ~/mpich3
```

After creating the necessary folder to house the installation source code, we will use "Get" to download the tarball.

```
$ wget
http.debian.net/debian/pool/main/m/mpich/mpich-
3.2.orig.tar.gz
```

Once the tarball has been downloaded we used the command to unpack the package:

```
$ tar xfz mpich_3.2.orig.tar.gz
```

Next, we issued the following command to create a place to put the compiled files in:

```
$ sudo mkdir /home/rpimpi/
$ sudo mkdir /home/rpimpi/mpich3-install
```

Next, we created a build directory, which will allow us to keep the source directory clean.

```
$ mkdir /home/pi/mpich_build
```

Change to the BUILD directory

```
$ cd /home/pi/mpich_build
```

Enter the following command to configure MPICH.

```
$ sudo /home/pi/mpich3/mpich_3.2.orig/configure-
prefix=/home/rpimpi/mpich3-install
```

The following command is used to make the installation files.

```
$ sudo make
```

To install the program, we issue the following command.

```
$ sudo make install
```

After MPCIH has finished installing, we use the following command to add the directory of binary files associated with MPICH, so that we can access them globally.

```
$ export PATH=$PATH:/home/rpimpi/mpich3-
install/bin
```

Now we can test whether the MPI installation was successful or encountered any problems.

```
$ mpiexec -f machinefile -n <number> hostname
```

But before we issue the above command we first need to create a machine file that stores the IP address of the slave nodes.

```
$ nano machinefile
```

We add the IP address of the current raspberry file into the machine. (the IP address is found by issuing the command ifconfig in a terminal window.)

```
192.168.1.100 (add your raspberry pi IP address)
```

After adding the IP address of the main node, we can now issue the following command to test the installation of MPICH:

```
$ mpiexec -f machinefile –n 1 hostname
```

A successful installation and executing of the previous command will output the following: *raspberrypi*

B. *Pyrit Installation*

Pyrit is an open source software, that uses a combination of Aircrack-ng and John the Ripper. We choose Pyrit over Aircrack-ng because of its MPI support. With Pyrit supporting the Open MPI framework, we can utilize all the resources of the Bramble cluster. To being installing Pyrit, we first must install the necessary dependencies:

```
$ sudo apt-get install libpcap-dev
$ sudo apt-get install scapy
$ sudo apt-get install python-dev
```

Use git to clone the repository of Pyrit:

```
$ git clone https://github.com/JPaulMora/Pyrit.git
```

Change to the downloaded repository directory:

```
$ cd Pyrit
```

Install Pyrit using the python automated installation script.

```
$ sudo ./setup.py build install
```

To verify that Pyrit has been installed, we issue the following command.

```
$ which Pyrit
```

In the next segment, we will install Dispy, Psutil, and Nmap onto the cluster so that we can run python scripts across the cluster and use Nmap to locate the IP addresses of the cluster nodes,

C. *Dispy / Nmap / Psutil Installation*

Dispy is a distributed Python implementation that is used to write code and execute them across a cluster. Version 4.7.1 was chosen because newer versions of the software have been proven to be incompatible with Raspbian. To begin the installation of Dispy, we simply enter the following command in the terminal:

```
$ sudo pip3 install dispy==4.7.1
```

Nmap is used to discover the IP address of the other Raspberry Pi slave nodes that form the cluster. To install nmap we issue the following command:

```
$ sudo apt-get install nmap
```

Once Nmap has finished installing, we use the following command to test whether nmap was installed correctly:

```
$ nmap -sP 192.168.0.*
```

Once the Bramble cluster has been constructed and configured, we will use Psutil to monitor the memory and CPU levels of each of the slave clusters and display it on the main node. The installation of Psutil is only necessary on the slave's node, but since we are going to replicate the master node SD card to have a consistent directory structure and working instances of software, we will install Psutil on the master node so that we can save time when setting up the remaining cluster. To install Psutil with the following command:

```
$ sudo pip3 install psutil
```

In this next segment, we will install John the Ripper to crack the encrypted archive password and the WPA/2 password. John the Ripper also supports the MPI framework which will allow for parallel computation.

D. John the Ripper (MPI) Installation

John the Ripper is a free, open source software developed by Openwall. It has become the go-to software for cracking various types of passwords using attacks such as dictionary, brute-force and rainbow tables. We also choose John the Ripper because of its support of MPI. With the support for MPI, computations from John the Ripper can utilize the resources of our Bramble cluster. Before we install JtR we first install the dependencies:

```
$ sudo apt-get install libssl-dev
```

Next, we will build John the Ripper from its latest source code. To clone the latest version of John the Ripper we use git to clone the official repository.

```
$ git clone
https://github.com/magnumripper/JohnTheRipper.git
```

Navigate to the repository folder that we just downloaded by entering the command in the terminal window.

```
$ cd JohnTheRipper
```

To configure JtR with MPI support we issue the following commands to configure, make and install John the Ripper.

```
$ ./configure --enable-mpi && make -s clean && make
-s
```

E. Replicating master node SD card.

Now that the master nodes SD card has been configured, we now need to create an image of the SD card so that we can replicate it to the slave nodes. To create an image of the SD card we use "Win32 Disk Imager". With Win32 Disk Imager downloaded and installed, we shut down the Raspberry Pi and eject its SD Card. Place the ejected SD Card into a USB SD Card reader and insert it into a computer. Going back to the Win32 Disk Imager window, enter a location and file name in the "Image File" text field, such as *C:\backup\masternodeimage.img* to set a location and file name to save the image of the master node. In the upper right corner of the program, in the Device segment, select the USB drive letter associated with the SD Card and then click the "Read" button to commence the image creation process. After the backup process has completed, safely eject the USB SD Card Reader and replace the SD Card with a blank SD card and reinsert the USB SD Card Reader into the computer. With the Win32 Disk Imager opened, click on the folder icon and select the image file that you have previously created, select the drive letter of the USB card reader under the "Device" segment and click "Write" to replicate the saved image to the new SD. Repeat this step for each Raspberry Pi slave nodes.

F. Slave Node Set-up

After cloning the master node to other SD Cards, insert a cloned SD Card into a Raspberry Pi and attach a mouse, keyboard, and monitor. Boot up the Raspberry Pi and open a terminal window. We are going to create a static IP address for each of the slave Raspberry Pis. With the terminal window open, enter the following command to open the network interface configuration file.

```
$sudo nano /etc/dhcpcd.conf
```

With the dhcpcd.conf open scroll down and look for where it states:

```
# Custom static IP address for eth0.
```

And uncomment the fields:

```
interface eth0
static ip_address=192.168.0.100/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```

Change the "static ip_address" to a sequence of numbers that you have chosen for your cluster network structure. For example, 192.168.0.100, 192.168.0.101, 192.168.0.102, etc. The number that you have chosen should reflect the position of the slave Raspberry Pi in the network cluster. For instance, if the master Raspberry Pi has a static IP Address of 192.168.0.100, then the first slave Raspberry Pi would be designated as 192.168.0.101 and so forth. Repeat this step for each addition slave Raspberry Pi's and optionally complete this set on the master Raspberry Pi for simplicity reasons. After assigning a static IP Address to each Raspberry Pi, make sure to enable SSH on each of the Raspberry Pi so that you can remote login to each Pi from the master node. Click on the raspberry icon on the taskbar and select the Preferences menu, then select Raspberry Pi Configuration. With the Raspberry Pi Configuration window opened, select the Interfaces tab and

click on enable for SSH if it is not enabled. In the next segment, we will create a new SSH key and copy it to each one of the slave nodes so that we can remote log in to each node without having to enter a password.

G. Setting up SSH access to slave Pis.

With the Raspberry Pi's setup and connected using a network switch or router, powered up, and assigned a static IP address we will run the following command to confirm that each pi shows up on the network successfully.

```
$sudo nmap -sP 192.168.0.*
```

Note that your IP address mask will be different, replace the IP address above with the network mask that you have chosen. The output will display network devices (Raspberry Pi's) found on the network, take caution when using nmap, because if you run the command above on a system that is connected to the internet, your ISP account could be flagged for malicious activity because nmap is used to map all IP addresses on a network. Now that we have mapped the cluster nodes, we will now generate an SSH key so that we can copy them to each of the subsequent nodes for a much easier means of communicating and access the cluster nodes. We start by entering the following command into a terminal window on the master node:

```
$ssh-keygen
```

Press Enter when asked for a location to save the key and then press Enter again twice when asked for a passphrase to leave it empty. Now that the RSA key has been created, issue the following command to copy the newly generated key for each node of the cluster.

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub <remote ip>
```

Replace <remote ip> with the IP address of the cluster nodes that were found using nmap. E.g. $ ssh-copy-id -i ~/.ssh/id_rsa.pub 192.168.0.101.
Confirm that the keys have been successfully copied, by issuing the command in a terminal window:

```
$ssh ip_address
```

Replace ip_address with one of the IP address of the nodes that you have copied the SSH key to. E.g. $ssh 192.168.0.101
If you have successfully copied the SSH key, you should now be able to SSH into any of the nodes without a password. In the next segment, we will create a directory that will be used to house the zip files that will be accessible by all nodes of the cluster.

H. NFS File Server Setup

On the main node of the cluster, we first must install the NFS software, which will be used to mount and share a directory of the main node.

```
$ sudo apt-get install nfs-server
```

Enter the following command to create a folder directory, which is where we will store shared files.

```
$ sudo mkdir /var/mpishare
```

Now, enter the following command to create shared permissions for the mounted folder.

```
$ sudo chown -R pi:pi /var/mpishare
```

After installing the NFS server and creating the shared directory, we need to add mounting points to the NFS configuration file. Edit the configuration file by entering the following command into a terminal window.

```
$ sudo nano /etc/exports
```

Add the following lines.

```
/var/mpishare <ip address>*(rw, sync)
```

For each of the nodes that are inside of the cluster, add another line from above with the IP address of that node to the configuration file and save it. To apply the changes made to the configuration file, you will run the command below to update the NFS service. Next, we will set up the mounting of the shared directory of the main node to the slave nodes. On a slave node, we will create a directory which we will use to mount the main nodes directory. Enter the following command to create a location for our mounting point.

```
$ sudo mkdir /var/mpishare
```

Change the permission of the NFS mount point to pi for easier user access with the following command.

```
$ sudo chown -R pi:pi /var/mpishare
```

To test mounting of the directory, issue the following command in a terminal window.

```
$ sudo mount <main node IP address>:/var/mpishare
/var/mpishare
```

To confirm that the directory has been successfully mounted, on the main node move or create any file to the location /var/mpishare and then return to the slave node and type in the following

```
$ ls /var/mpishare
```

The output should list the files that you've placed into the MPI shared folder on the master node. In the next section, we will outline the flow of the experiment and introduce the controlled variables.

III. Experimental Results

The experiment consists of cracking an encrypted archive with a set password between five and eight characters and a Wi-Fi WPA/2 handshake consisting of sniffed packets from a mobile device to a wireless network. We will be cracking a zip file with a specified password and a Wi-Fi handshake that consists of the same or similar password utilizing the WPA/2 protocol on the Bramble cluster; consisting of a single node, two nodes, four nodes, a laptop and a desktop. The purpose of using a desktop is to establish a baseline for comparing the results from the bramble cluster and laptop. The goal of this project is to build a portable penetration testing rig that is headless and power efficient. Figure 2 illustrates the process involved in ascertaining the information from a wireless network, password guidelines for the encrypted archive, followed by initialization of the cracking process, analysis of results and lastly the repetition of the process for each scenario. We expect to see similar performance for the four-node cluster in comparison to the laptop when it comes to cracking simple passwords for an encrypted zip file.

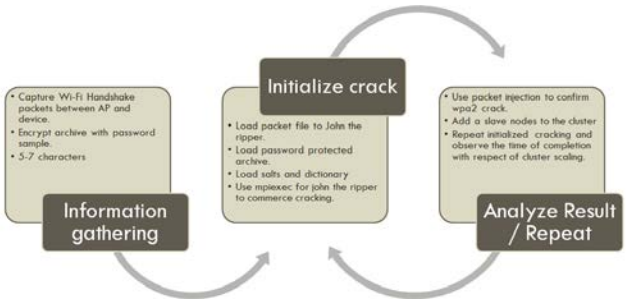

Figure 2 – Flow chart for penetration testing.

The results that were obtained from the cracking process of the encrypted archive are presented in Table 1 and the results for the Wi-Fi handshake cracking are presented in Table 2. The results are organized by the time it took for each scenario to complete cracking the password, followed by successful guess per second (g/s), password tested per seconds (p/s), crypts (password hashes) computed per second and lastly the crypts tested per second (C/s). The desktop and laptop that were used in the experiment only utilized its central processing unit instead of its discrete graphics card. The CPU for the desktop was an i7-6700 and the laptop utilized an i5-6300U. In the next section, we will compare the scenarios for a single node, two nodes, laptop and a desktop with a four-node cluster, for the encrypted archive cracking process and the Wi-Fi handshake cracking process. In the next section, we will compare the scenarios for the one node, two nodes, laptop and desktop to the four-node cluster.

Table 1 - Results from JtR for encrypted_arc.zip

Scenario	Time (m:s)	g/s	p/s	C/s	c/s
Desktop i7	1:59	0.0008337	30677K	30677K	30677K
Laptop i5	5:04	0.003279	12067K	12067K	12067K
1-Node	15:57	0.001043	979510	979510	979510
2-Node	6:32	0.002548	1343K	1343K	1343K
4-Node	3:31	0.004738	1471K	1471K	1471K

g/s = successful guesses per second
p/s = password tested per second
C/s = crypts (password hashes) computed per second
c/s = crypts tested per second

Table 2 - Results from JtR for WPA/2 handshake

Scenario	Time (h:m:s)	g/s	p/s	C/s	c/s
Desktop i7	3:44	0.004460	4961	4961	4961
Laptop i5	15:22	0.001083	1205	1205	1205
2-Node	1:21:06	0.000205	28.54	28.54	28.54
4-Node	32:20	0.000515	35.74	35.74	35.74

g/s = successful guesses per second
p/s = password tested per second
C/s = crypts (password hashes) computed per second
c/s = crypts tested per second

The equations that we used to calculate the differences between Scenario A (1-Node, 2-Node, Laptop, Desktop) to Scenario B (4-Node) is presented in Equation 1, where delta V indicates the voltage difference. Therefore in Table 3 and 4, a negative number indicates superior performance of Scenario A when compared to Scenario B. The power consumption of a single raspberry pi under full load from John the Ripper was around 11.8 Wh, which was considerably less than a tradition laptop or desktop. Overall the power consumption of the 4-node cluster that we built only consumed less than 100 Wh of power, even with the use of the Wi-Fi sniffer.

$$Percentage\ Difference = \frac{\Delta V}{\frac{\varepsilon V}{2}} * 100 \qquad Eq.1$$

The results that were obtained for cracking the password for the encrypted archive in Table 3 was consistent with our expectations. The Raspberry Pi computer system is widely known as a low-performance computer so observing how it stacked up to a desktop was not much of a surprise. However, comparing a single Raspberry Pi to the 2-node and 4-node cluster was interesting as it demonstrated stable performance scaling. The resulting difference between the single Raspberry Pi and two node cluster showed that the two-node cluster completed the operation in half the time of a single unit (which

indicates a scaling factor of one-half). With the addition of an additional node, we find a performance increase of one-fourth.

Table 3 - Comparison of encrypted_arc.zip on 1-Node, 2-Node, Laptop and Desktop to the 4-Node Cluster

Scenario A	Time (m:s)	Scenario B	Time (m:s)	% Diff
1-Node	15:57	4-Node	3:31	127.459%
2-Node	6:32			59.6026%
Laptop	5:04			35.6589%
Desktop	1:59			-56.1934%

A negative number indicates greater performance for Scenario A over Scenario B.

The data in Table 4 indicates that the 4-Node cluster was two times slower to crack the Wi-Fi password when compared to an i5 Laptop. The Raspberry Pi's power performance scales better in a Wi-Fi cracking scenario than in the password encrypted archive. Performance between a single node and a 4-node was much greater than a single node and a 2-node cluster. The duration difference between the Raspberry Pi Cluster and the tradition computers were not a feasible solution for a portable penetration testing rig due to the wide gap in the time it took to crack the passwords.

Table 4 - Comparison of Wi-Fi Handshake on 1-Node, 2-Node, Laptop and Desktop to the 4-Node Cluster

Scenario A	Time (h:m:s)	Scenario B	Time (m:s)	% Diff
1-Node	2:45:31	4-Node	32:20	134.631%
2-Node	1:21:06			85.983%
Laptop	15:22			-71.1391%
Desktop	3:44			-158.595%

A negative number indicates greater performance for Scenario A over Scenario B.

IV. Conclusion

We successfully accomplished our goals of cracking a zip file with a specified password and cracking a Wi-Fi WPA/2 handshake consisting of sniffed packets from a mobile device to a wireless network on two nodes, 4 nodes, a desktop computer and a laptop utilizing a Bramble cluster. Our results indicate that it is feasible to construct a portable penetration testing rig that is both headless and power efficient. We found that a 4-node cluster was 35% faster in cracking the encrypted archive password when compared to the i5 Laptop. Our results reveal that while using a Raspberry Pi cluster as a portable penetration testing rig was achievable, given the time required to creack WPA/WPA2 passwords it would be more efficient to create a cluster utilizing traditional computers. Additionally, we believe further testing can be done on future bramble clusters to compare the efficiency and speed required to break more complex passwords, as well as to study the effects of additional pis on the cluster.

Acknowledgment

The authors gratefully acknowledge support of the National Science Foundation grant Cloud Computing – Data, Networking, Innovation (CC-DNI), area 4, 15-535, also known as "SecureCloud".

Reference

[1] https://www.privacyrights.org/databreaches?title=&taxonomy_vocabulary_11_tid%5B%5D=2434&=Search+Data+Breaches (last accessed Jan 2018)

[2] R. Meulen and C. Petty, "Gartner forecasts worldwide security spending to reach $98 Billion in 2018", https://www.gartner.com/newsroom/id/3836563 (last accessed Jan 2018)

[3] G. Weidman, *Penetration Testing: A Hand-On Introduction to Hacking*, California: No Starch Press, 2014.

[4] Core Security guide to Penetration Testing, https://www.coresecurity.com/content/penetration-testing (last accessed Jan 2018)

[5] MPICH specification, http://www.mpich.org/about/overview /(last accessed Jan 2018)

[6] John the Ripper specification, http://www.openwall.com/john/ (last accessed Jan 2018)

[7] Aircrack-ng specification, http://aircrack-ng.org/ (last accessed Jan 2018)