**SSH Honeypot**

Project Report

**Introduction:**

In the modern world of computing, we exchange lot of data over cyber space. Much of this data is private and needs to be protected from the hands of malicious attackers. To safeguard our cyber space, our security policies and prevention techniques need to adapt overtime. In order to do this, we need to gain a better understanding of attackers.

**Objective:**

The objective of this project is to gain intelligence on cyber threats. To do this we will implement an SSH honeypot from scratch and deploy it in a publicly accessible server to be attacked on the Internet. We will log the incoming attack traffic and visualize it in two ways. First, we will provide a threat map of where the attack traffic is coming from and secondly, we will visualize the top 100 tried passwords. This will provide information on the geographic locations used to perform the attacks as well as the password variations that are used.

**Types of Honeypots:**

In computing, honeypots are security mechanism set to detect, deflect, or counteract attempts in some manner at unauthorized use of information system. Broadly, they can be classified into two categories, production and research honeypots. This project falls under the research honeypot category. Honeypots can also provide high and low interaction. There are many well-known SSH Honeypots available on internet such as Kippo. You can use the standard version, or you can use Cowrie which is variation based on Kippo. Kippo and Cowrie both offer high interaction capabilities. For the purposes of the project, we will develop and deploy a low interaction honeypot as this is sufficient to collect password and location data.

**SSH Protocol**

The SSH protocol is a method for secure remote login from one computer to another. It uses encryption to provide secure services over unsecured networks. The service traditionally runs on port 22. We will use the Paramiko python library to implement the SSH protocol.

**Method:**

Attackers can scan networks and check if there are any open ports or services running using tools such as nmap. If attackers identify that a service is running on an open port, they can try to gain access to the system. They could attempt brute force attacks or check for known vulnerabilities on the running services.

The SSH honeypot that we deploy will run a dummy SSH service on port 22. This will appear to be a legitimate service. Since this is low-interaction level honeypot, it is designed to reject all login attempts. However, it will collect metadata about attacker who attempt to login.

Hardware used for this project is Raspberry pi 3. We installed Raspbian Desktop version OS on 8 GB of memory card which we used throughout this process. The SSH honeypot was tested first on a local network using virtual machines and then deployed into a public network. The below image shows the network topology.

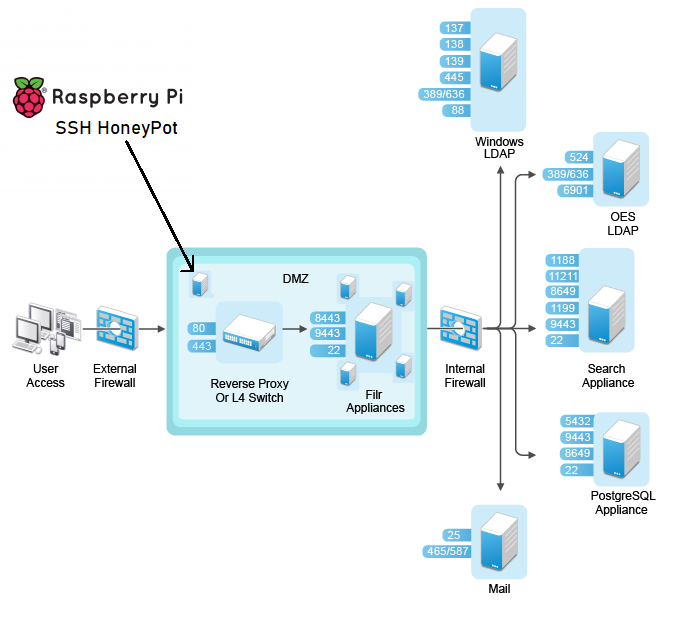


Figure showing the network topology

The raspberrypi is present just after the external screening firewall and is a part of the DMZ. It has its own IP (140.226.235.13) which is owned by the University of Colorado. Being in front of the internal firewall, our honeypot received all the malicious traffic that mostly gets dropped by it. This helped in collecting real attacks as the connection is almost unfiltered.

If you’re using any other hardware, then you have to install python interpreter since this project is developed in python and its libraries. Install software dependencies like pip, git and paramiko for making this project available on your local machine. Paramiko is python implementation of SSHv2 protocol providing both client and server functionality. Paramiko uses Python C low-level extension for Cryptography implementation. Since we know that SSH is secured protocol and need cryptographical algorithm for encrypted message exchange.

This project will create SSH server and bind it on port 22 on host hardware. To setup you have to follow certain steps to keep server running on port 22. First make sure there no other service is running on port 22. There are several methods depending on your host operating system. First start server and make sure it runs all the time when system is up. You can setup cronjob if you are using linux environment or in windows system you can use task scheduler to make this server running. Directory keys contains private key which will be used for ssh service in paramiko module. We have made this server to allow limited number to simultaneous logins, so it uses threading concepts of python library to achieve mutual exclusion.

Server can log all coming request into file. Current attempts can also be seen live on server command line. There is basic error code printed on server command line as well if any exception is triggered. Simultaneously all entries are fetched for analysis. We have live threat map which will map this request into geographical location. All requests are parsed for analysis into pandas dataframe. We can generate many reports, for example top 20 most common tried passwords and username combination. Top countries which have used this dummy SSH service to attack. How many requests are made from same location? At what intensity these requests are made and from behaviors we can understand are those bots or actual human attempting those.

When we created this server first time and tried to run locally, we face several issues. There were already other services running on same port on which we were trying to deploy SSH server, but we couldn’t stop them since some of those service had root privilege. When we deployed project into actual environment, we couldn’t configure it to be public facing device which is direct access point from outside network. Since getting static ip was challenge we used university secured network with all required permission from Office of Information Technology to deploy this service on specific port.

Following proper steps allowed us to deploy project successfully and in first run it was up and running for 3 consecutive days. We collected almost 15 thousand records in that first run and we shut the service to analyze data that we have collected. More about results is described in next section.

**Results:**

We found 11054 intrusion attempts in just 3 days!

Here are top 100 passwords we sorted in the descending order of the frequency out of **7966** unique passwords.

|  |  |
| --- | --- |
| **Password** | **Counts** |
| admin | 36 |
| root | 27 |
| 123456 | 22 |
| password | 21 |
| 1234 | 21 |
| 12345 | 16 |
| raspberry | 15 |
| ubnt | 15 |
| 123 | 14 |
| 1q2w3e | 13 |
| 123qwe | 13 |
| guest | 13 |
| 1qaz@wsx | 13 |
| 1234567890 | 12 |
| 1qaz2wsx | 12 |
| 12345678 | 12 |
| abc@123 | 12 |
| user | 12 |
| 1234567 | 12 |
| !@ | 12 |
| test | 11 |
| 1 | 11 |
| support | 11 |
| 1234qwer | 10 |
| 123qwe!@# | 10 |
| default | 10 |
| 1qaz@WSX3edc | 9 |
| 123qweasdzxc | 9 |
| 123qweasd | 9 |
| 12qwaszx | 9 |
| changeme | 9 |
| 123456789 | 9 |
| abc123 | 9 |
| 111111 | 9 |
| 1qazxsw2 | 9 |
| 1qazXSW@ | 9 |
| passw0rd | 8 |
| 1QAZ2wsx | 8 |
| a123456 | 8 |
| 123123 | 8 |
| Admin@123 | 8 |
| 12345qwert | 8 |
| 123abc | 8 |
| 1qaz!QAZ | 8 |
| Abcd1234 | 8 |
| smoker666 | 7 |
| 123qwe!@#QWE | 7 |
| 1q2w3e4r5t | 7 |
| 123.com | 7 |
| 1q2w3e4r | 7 |
| Aa123456 | 7 |
| sales | 7 |
| 123qaz!@# | 7 |
| 8ik | 7 |
| 123456qwerty | 7 |
| admin@123 | 7 |
| pass | 7 |
| 7 | 7 |
| oracle | 7 |
| 1qaz2wsx3edc | 7 |
| administrator | 7 |
| toor | 7 |
| operator | 6 |
| apache | 6 |
| 123@qwe | 6 |
| abc-123 | 6 |
| git | 6 |
| ubuntu | 6 |
| 123456!@# | 6 |
| 2wsx3edc | 6 |
| asdfghjkl | 6 |
| welcome | 6 |
| ABCabc123 | 6 |
| !QAZ@WSX | 6 |
| 54321 | 6 |
| alpine | 6 |
| 11111 | 6 |
| avonline | 6 |
| 1qaz1qaz | 6 |
| 654321 | 6 |
| abcd1234 | 6 |
| temppwd | 6 |
| 123@123 | 6 |
| admin123456 | 6 |
| admin123 | 6 |
| abcde12345 | 6 |
| 123123123 | 6 |
| 1234!@#$ | 6 |
| 121212 | 6 |
| abc | 5 |
| 1a2s3d4f | 5 |
| pa55w0rd | 5 |
| raspberryraspberry993311 | 5 |
| 1q2w3e4r5t6y7u | 5 |
| qwer1234 | 5 |
| caonima | 5 |
| 123321 | 5 |
| 11223344 | 5 |
| zxcvbnm | 5 |
| 11112222 | 5 |

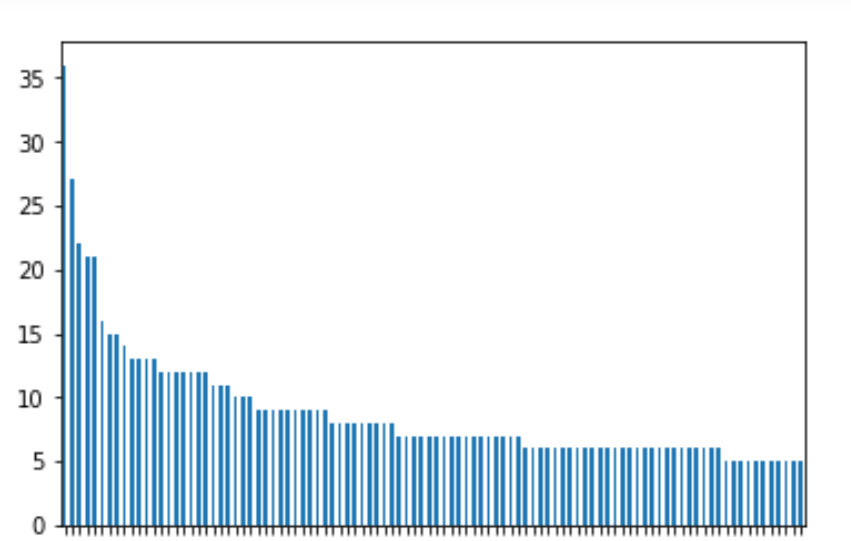


Figure showing the plot of top 100 passwords

**User ID analysis:**

A total of 147 unique user id popped up.

Here are top 20 of them:

|  |  |
| --- | --- |
| **User** | **Count** |
| root | 10687 |
| admin | 57 |
| pi | 19 |
| support | 14 |
| user | 14 |
| ubnt | 13 |
| guest | 10 |
| hfghfgh | 9 |
| butter | 8 |
| test | 7 |
| bin | 6 |
| ubuntu | 6 |
| debian | 5 |
| oracle | 5 |
| daemon | 5 |
| fake | 4 |
| sales | 4 |
| SP35 | 3 |
| dasusr1 | 3 |
| osmc | 3 |

**IP analysis:**

A total of 88 different ‘IP’ addresses showed up in our data.

Here are the top 20 of them:

|  |  |
| --- | --- |
| **IP** | **Count** |
| 140.143.250.114 | 4033 |
| 62.231.7.221 | 1256 |
| 218.92.0.201 | 731 |
| 218.92.0.205 | 524 |
| 218.92.0.211 | 364 |
| 103.44.144.53 | 183 |
| 103.59.208.101 | 107 |
| 103.125.191.106 | 94 |
| 218.92.0.208 | 87 |
| 3.8.32.113 | 77 |
| 179.242.183.104 | 65 |
| 177.197.36.120 | 52 |
| 106.52.151.34 | 43 |
| 123.51.152.54 | 27 |
| 61.143.61.86 | 19 |
| 104.244.72.98 | 12 |
| 107.189.11.160 | 11 |
| 194.180.224.100 | 9 |
| 78.195.108.27 | 8 |
| 178.198.209.138 | 8 |



Figure showing the plot of the attacks from different locations

**Discussion:**

From above results we understood that there are always high number of requests on our gateway which we hardly know. Attacker always tries to exploit vulnerabilities which are preventable in some cases if we follow proper precautions. This project helped understand many aspects of those attacks. It took the attackers just 50 mins to discover this new machine on the web. Furthermore, we followed those data for reverse analysis and check what we can find. Are they masquerading also to cover their tracks? What are the famous and preferred techniques which they tried on our system? Were any of them successful? Many of questions were answered but since we implemented this project on low interaction level, we had limited data to analyze. We can implement original like file system and let attacker play with it. By such manner we could have observe more behavioral patterns which are in trend currently.

**Conclusion:**

There are many ways we can secure our cyber space. Creating honeypot and analyzing patterns from that could be more cost effective in some case or for research purpose. Results for SSH Honeypot were really fascinating for us and they helped us in many ways to create secure cyber space. Try this project, deploy at your network gateway and check how fascinating these results are. This is project is made with open source libraries so you can refer documentation with source libraries for modification and develop extended version of it.

**Future Work:**

With all the codebase and infrastructure, we used in this project, a lot of different future opportunities opens. As we followed an approach of modular programming, we made sure that it will be easy to extend for any future contributor. Here are some of the things that can be added to fire it up.

1. The Data gathered as a result can be used to learn attack patterns, by guessing the way an attacker tries different passwords.
2. The threat visualization tool is in its nebular form. Requires a scaled map for edge cases. Also, it can be tinkered with to apply more colors and a dashboard to display the hitlist locations. A geoip tool can be added to fetch the city/country. The code can be found at <https://github.com/Xkdash/OpenLTM> and can be used as an api for other projects too.
3. The physical setup can be improved by providing an advanced hardware for handling more connections simultaneously and capture more traffic without getting overwhelmed.

**Reference:**

1. What is shell: <https://en.wikipedia.org/wiki/Shell_(computing)>
2. What is ssh: <https://en.wikipedia.org/wiki/Secure_Shell>
3. What is honeypot: <https://en.wikipedia.org/wiki/Honeypot_(computing)>
4. Different levels of SSH Honeypot implementation: <https://www.infragistics.com/community/blogs/b/torrey-betts/posts/what-i-learned-after-using-an-ssh-honeypot-for-7-days>
5. Kippo Honeypot: <https://en.wikipedia.org/wiki/Kippo>
6. Cowrie Honeypot: <https://medium.com/threatpunter/how-to-setup-cowrie-an-ssh-honeypot-535a68832e4c>
7. Paramiko Honeypot: <http://www.paramiko.org/installing.html>