Building A Global Network of Sensors to Understand Internet Security: Creating a Modern Honey Network (MHN)

In partial fulfillment of the requirements of
W251 Scaling Up

Jill Zhang
Leslie Teo
Todd Young

Table of Content

Introduction

The goal of our final project is to better understand internet security. This project was motivated by the observation that our Virtual Servers created for class was subjected to numerous failed login attempts.

Our idea was to create a network of "servers" across the globe and then to capture unauthorized attempts at access ("attacks"). We would collect this data and analyze it.

As we studied this problem we discovered that several security-related open source projects provide possible solutions. In particular, we discovered a rich set of "honey pots" which are decoy machines/infrastructure that is designed and deployed to be attached. The Modern

Honey Network (MHN) (https://github.com/threatstream/mhn) provides a simple way to connect, manage, and display information on attacks.

In this project implemented four MHNs with each server connected to between [5] to 20 “honey pots”. This enabled us to collection upwards of 3,000,000 attacks over the last month.[1] In addition to the challenges associated with setting up the network, we decided to supplement both streaming and historical analysis of the data with methods used in class (e.g. Spark, Spark Streaming, ELK, and Splunk) to both learn from and to compare these methods.

This paper is structured with the following three sections:

1. How was the MHN set up? What is the structure of our MHNs? What are its the components, and how each part work with each other?
2. How did we process the data from our MHNs?
3. What did we learn and what are further extensions to this project?

Our Modern Honey Network

We used software provided by the Modern Honeypot Network to facilitate our work. We deployed honeypots of various types and on various cloud-based computing platforms. There are two major types of devices in a honeypot network: Servers and Sensors.

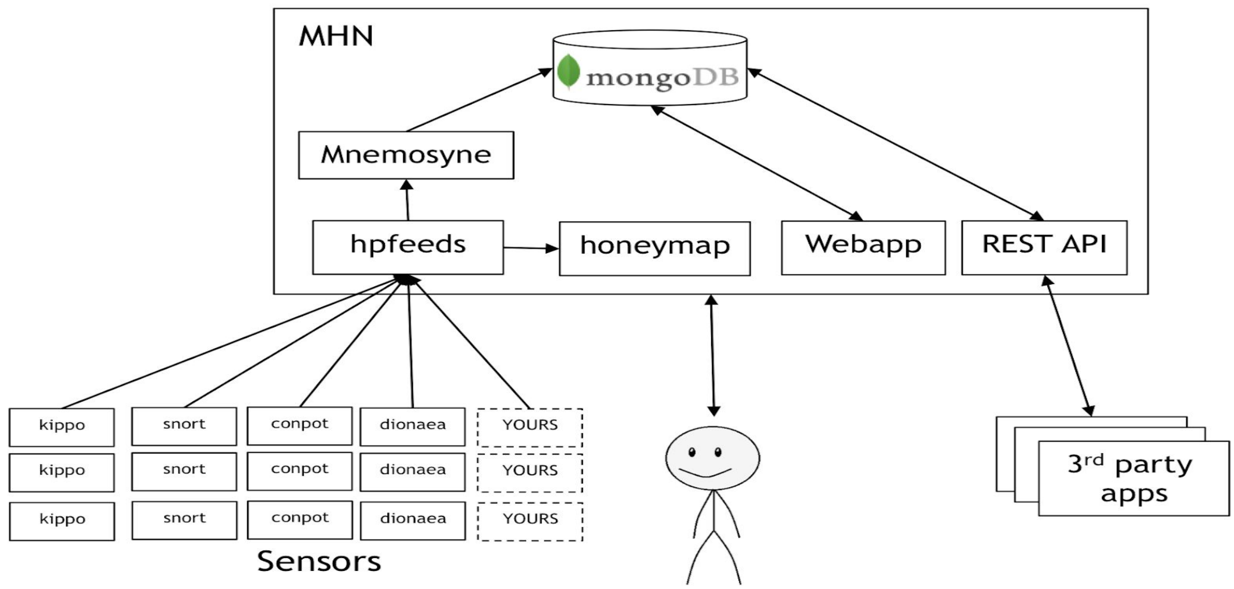
Server

The MNN server is where the Modern Honey Network software is installed. We installed our server on virtual machines with at least 16GB in memory and 2 CPUs, running Ubuntu 16.04 or 14.04.[2] MHN consists of the main framework of the honeypot server, including Mnemosyne, hpfeeds, and the geolocation (Honeymap) services (see Figure 1).

Figure 1. The MHN Server

[1] Our MHN servers/network were not set up at the time. The first we set up around mid-July, most were set up in early August.

[2] We learned the hard way that other linux distributions may not be fully compatible with MHN. As we had to install 40 over virtual machines we used Ansible to carry out routine set up. We found this much easier to use that Salt.



Hpfeeds is the publishing/subscriber system that exchange attacking feeds that collected from the sensors. Mnemosyne provides immutable persistence for hpfeeds, and normalizes the data to enable sensor agnostic analysis and expose the normalized data through a RESTful API. The honeymap transforms the source ip information to geo location that shows on the web server UI.

For this project, we set up 4 server as below. The native honey map UI is available at http://server-ip:3000, while point to the server-ip then logging in will provide access the the MHN server including some basic data.

1. 184.173.18.156
2. 184.173.47.227
3. 119.81.53.116 DEMO MACHINE THAT IS STILL RUNNING [login: lt22202@gmail.com, W251.Project]
4. 161.202.173.204

Honey Pots

Honey pots are the virtual servers that are set up to be exposed to potential attackers on purpose. We set up 45 honey pots across the world and with different kinds of cloud service providers (Table 1).[3] Our servers ran Ubuntu, but did not require as much CPU or memory as the server. We hoped that the distribution of honey pots would allow us to observe attacks globally; we also wondered if there would different honey pots in different regions would experience different attacks.

[3] For reasons of cost, most of our honey pots were created on Softlayer and Amazon Cloud.

Table 1. List of VS with Sensors

Location	IP	Provider		Location	IP	Provider
Singapore	119.81.53.118	SL		Singapore	52.221.252.112	AWS
Singapore	119.81.53.115	SL		Mumbai	13.126.194.202	AWS
Amsterdam	169.50.175.2	SL		N Virginia	54.210.124.97	AWS
Dallas	169.48.188.43	SL		Canada	35.182.57.108	AWS
San Jose	169.45.93.11	SL		London	52.56.247.95	AWS
Toronto	158.85.107.118	SL		Sao Paolo	52.67.197.185	AWS
London	46.101.46.189	Digital		Hong Kong	47.52.26.7	AliCloud
Frankfurt	165.227.134.114	Digital		Houston	184.173.18.158	SL
Bangalore	139.59.4.205	Digital		Houston	184.173.18.155	SL
San Fran	165.227.9.187	Digital		Houston	184.173.18. 232	SL
Singapore	119.81.53.114	SL		N Virgina	172.31.55.4	AWS
Amsterdam	169.50.175.7	SL		Mexico	169.57.0.146	SL
Dallas	75.126.206.186	SL		Singapore	54.169.90.171	AWS
San Jose	169.45.93.4	SL		Sdyney	52.64.99.227	AWS
Toronto	158.85.107.117	SL		Tokyo	54.65.91.144	AWS
Chennai	169.38.101.39	SL		Sao Paolo	18.231.63.13	AWS
Frankfurt	158.177.78.102	SL		N Virginia	54.87.202.64	AWS
Hong Kong	119.81.151.205	SL		Ohio	18.220.145.228	AWS
London	158.176.126.27	SL		N California	54.183.136.214	AWS
Canada	35.182.142.179	AWS		London	52.56.197.48	AWS
Ireland	52.30.252.2	AWS		Oregon	52.35.117.218	AWS
Frankfurt	35.158.209.37	AWS				

Sensor Types

We followed the general recommendation to include 4 types of sensors on each honey pot:

Dionaea: Dionaea captures malware sent to exploit common service (e.g. smb, http) vulnerabilities. It is meant to be a nepenthes successor, embedding python as scripting language, and uses libemu to detect shellcodes.

Kippo: Kippo is a medium interaction SSH honeypot designed to log brute force attacks and, most importantly, the entire shell interaction performed by the attacker.

P0f: P0f is a tool that utilizes an array of sophisticated, purely passive traffic fingerprinting mechanisms to identify the players behind any incidental TCP/IP communications (often as little as a single normal SYN) without interfering in any way.

Snort: It is an open source intrusion prevention system capable of real-time traffic analysis and packet logging

p0f and snort provide very useful contextual data for threat intelligence such as the attacking host's operating system, its uptime, its network connection type, and possibly some characteristics of the attack payload such as vulnerability exploited or malware family.

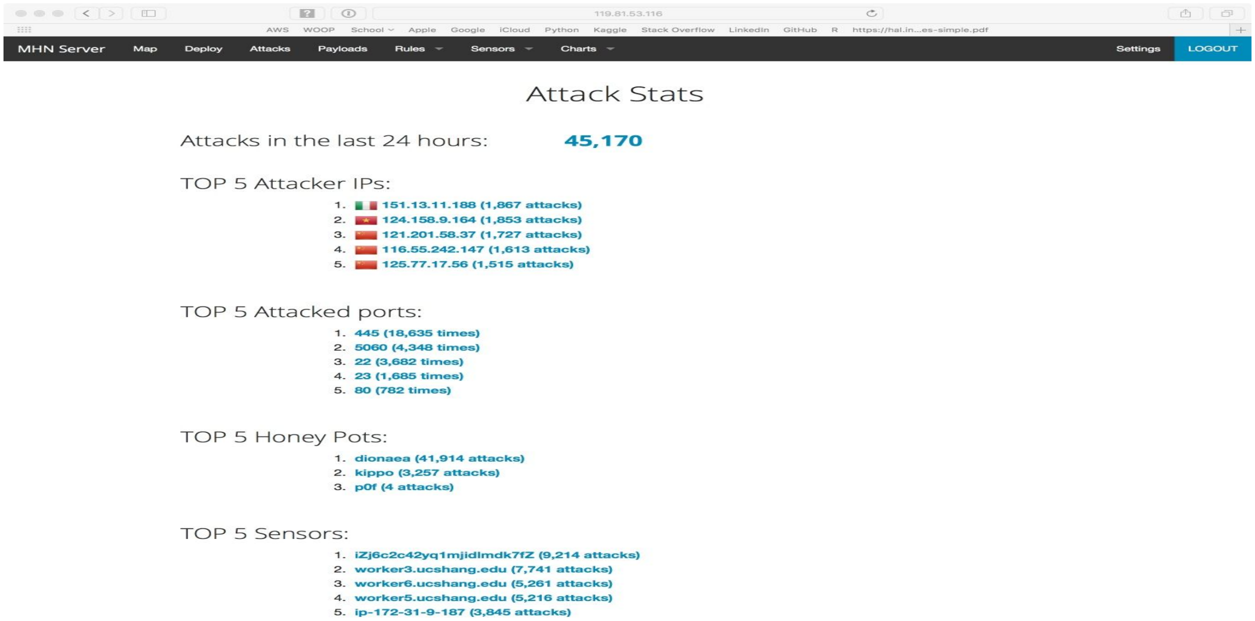
Honeypots such as Dionaea, co-deployed with p0f and snort can be a very effective sensor for catching these sorts of network activity and MHN will ensure that the data collected from these sensors will be stored and integrated with a SIEM such as ArcSight, Splunk, or even ELK.

Basic Information

Once our MHN was set up and sensors deployed, MHN provides some basic real time information on attacks.

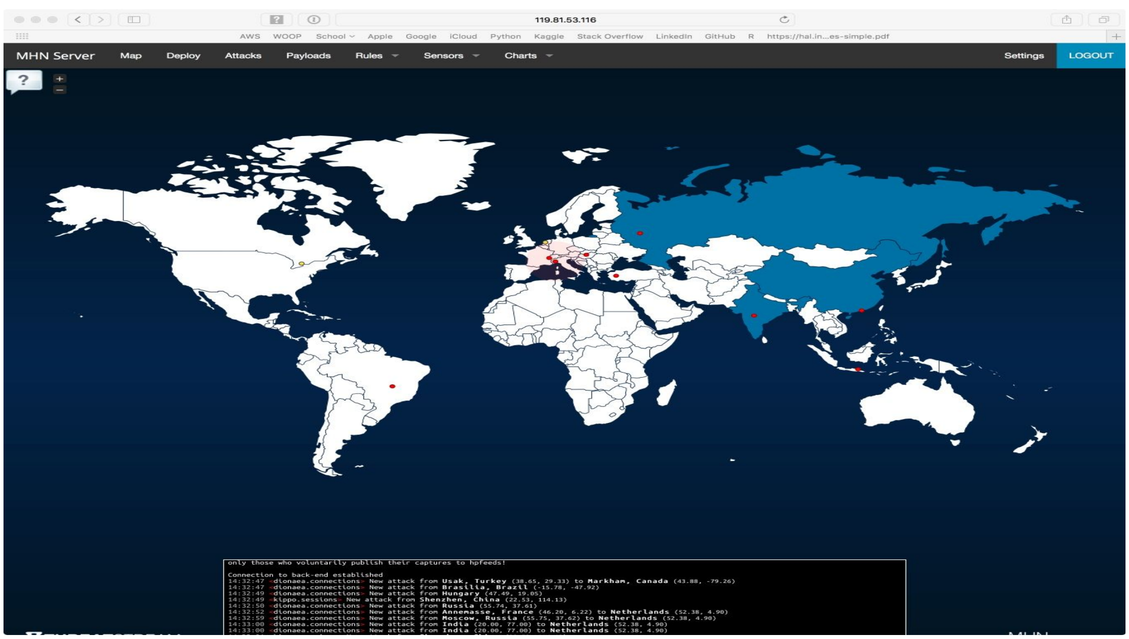
Figure 2 shows basic attack stats available such as the number of attacks over the last 24 hours, the top 5 attacking IPs, their country of origin, and ports attacked.

Figure 2. Basic Stats



The MHN Server also provides a streaming map of attacks (Figure 3). See for example http://119.81.53.116:3000.

Figure 3. Map (Available at http://serverid:3000)



Extending Data Exploration

The information provided the MNH server was interesting but it's presentation was inflexible and somewhat limited. For example, windows were defined in 24 hour blocks. Much of the collected attack data were presented or analyzed.

We therefore decided to extend the data collection and analysis along four main threads:

1) Streaming the data via Spark to enable us to be more flexible in analysis (e.g. windows)
2) Streaming the data via ELK
3) Streaming the data via Splunk
4) Processing the complete historical data

Each of these is discussed in detail in next.

Spark Streaming (Real Time)

The Streaming application is implemented through Mongodb conditional queries. The socketStream.py file is written to query the mongodb database on the server every 10 seconds (current.time - 10 second to now). A socket instance is created in the script and send the query data to TCP localhost and port 6000. To get more information about the attacks, we used the geoip library and get the country, city, region and dma information about the source ip.

The detailed steps are as below:

1. Define the Geoip function. Fetch the geo location information from the IP address. The following information is returned.
    a. Country
    b. City
    c. Region
    d. DMA
2. Define the mongdb query. The script query the mongbd database every 10 seconds and send the data to the tcp stream service
3. The spark streaming receive the data from the tcp streaming and split the input stream to different field.
4. The streaming data was aggregated by country, city, destination port, destination ip and Honey port type

How to access the MHN streaming services:

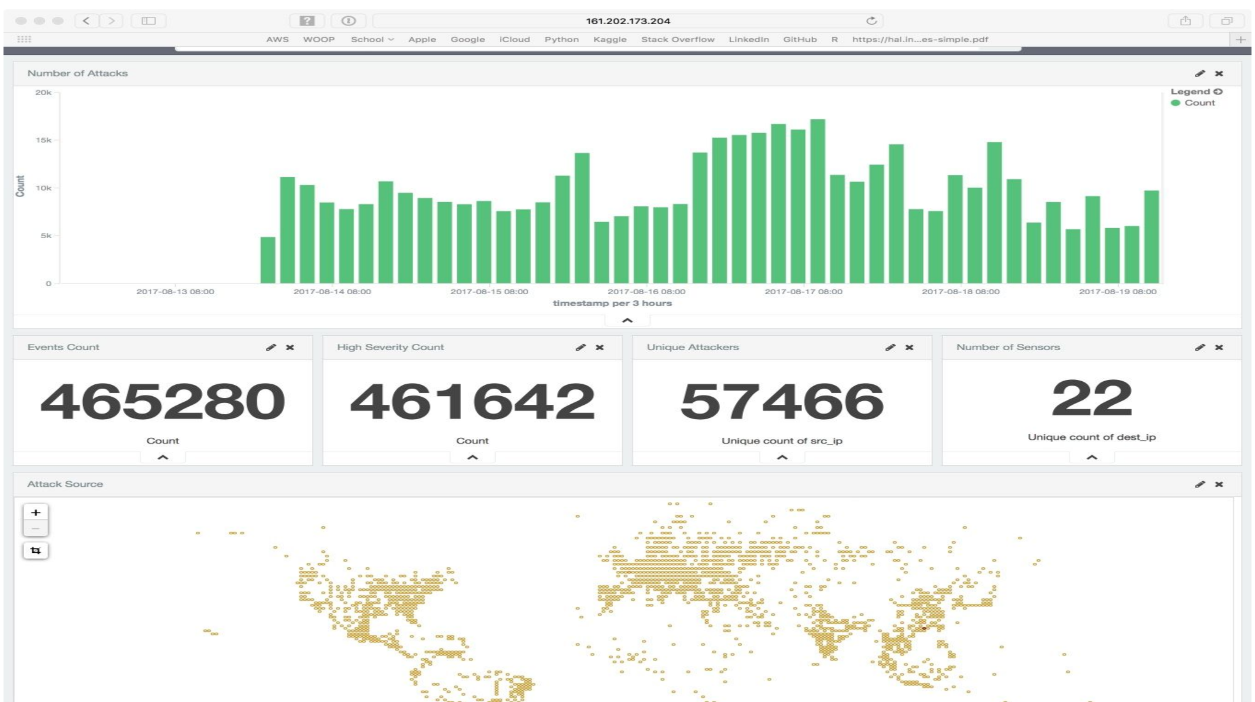
1. On the /root folder, run the streaming script as `python socketStream2.py`
2. Run `$SPARK_HOME/bin/spark-submit --class "honeypot" $(find target -iname "*.jar")`

Streaming with ELK

The MHN server includes options to integrate with a logstash, elasticsearch, and kibana stack. Essentially we would modify the hpfeeds to produce files which would be piped to logstash, processed in elasticsearch, and then visualized in kibana.[4]

The working product (http://161.202.173.204:5601/) is much more flexible compared to native implementation. Aside from a deep dive into the data, varies charts/metrics can be produced and put together in dashboards (Figure 3).

Figure 3. A Kibana Dashboard for MHN



---

[4] MHN works with Elasticsearch 1.4, Kibana and Logstash 4. Much time was spent trying to integrate these software and dealing with conflicts.

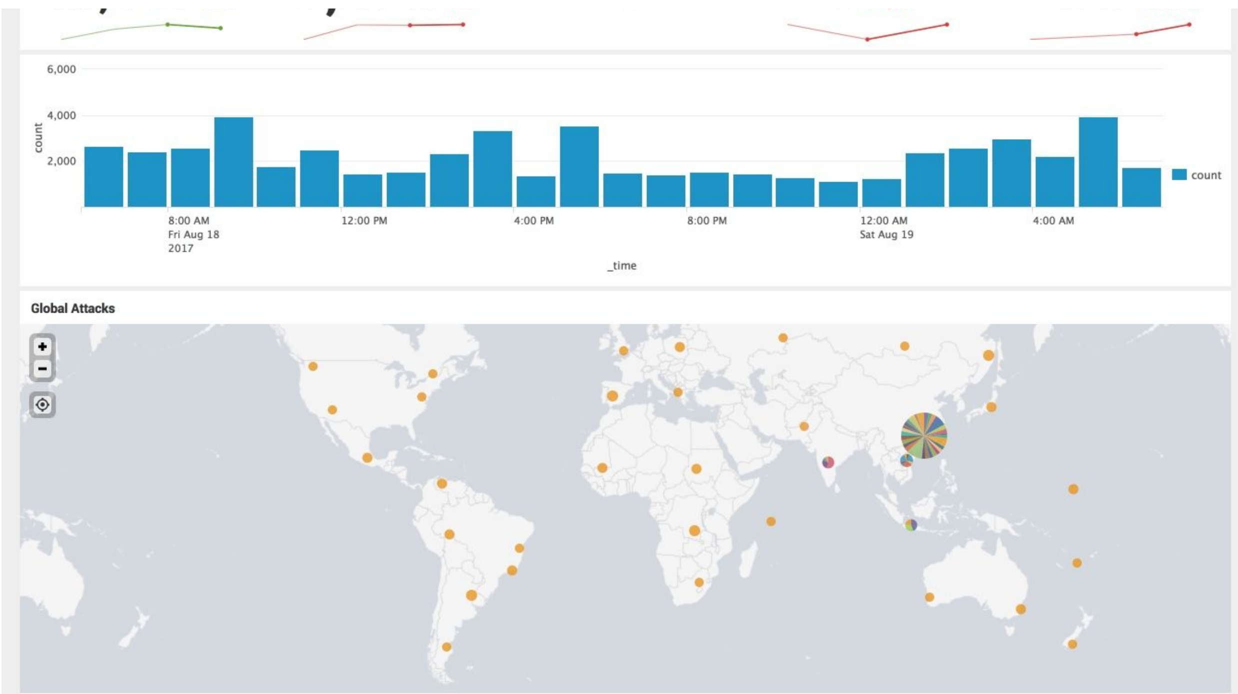
Streaming with Splunk

Splunk is a commercial software that is used for search, monitoring, and analyzing machine generated data. Splunk has a module for MHN so we installed the free version of Splunk so try it out.

A working version is here
(http://119.81.53.116:8000/en-US/app/mhn-splunk/overview?form.field1.earliest=-24h%40h&form.field1.latest=now)
With admin, W251.Project

Like Kibana, more data can be analyzed and presented in dashboards (Figure 4).

Figure 4. Splunk for MHN



Batch Processing of Historical Data in Spark/Hadoop

On approximately August 10, we stored the contents of the MHN Server MongoDB Session database for batch processing. At that time, there were approximately 1.2 million rows of attack data.

The data was processed as follows:

1. The data was copied into HDFS so that the data could be processed on a 4-node, 16-core Spark Cluster.

2. A scala job was submitted to the cluster to clean and flatten the json data. The output from MongoDB was not a true JSON format, and the JSON fields were nested into three layers. The output data has one JSON line per attack, and the fields are all under the top level.

A shell script /final/submit_spark_job.sh simplifies submission of main.scala to the cluster.

The output is written to hdfs as a single json file.

3. We wanted to geocode the data based on IP addresses, so we used a pygeoip package from MaxMind http://dev.maxmind.com/geoip/

The script hp_geo.py reads the json file from step 2 above (from HDFS) and adds the following columns to the data:

- country
- country code
- city
- region
- dma (subregion)

The data was converted into a DataFrame to take advantage of SQL-like querying capabilities, and we generated Top 20 lists for various fields and field combinations for the data including:

- Top Attacking Countries
- Within China/Russia, Top Attacking Cities
- Top Attacking IP addresses
- Top Ports Attacked
- Top Protocols used in Attacks

Access:

At http://192.155.209.117:8080 you can find the webGUI for the spark cluster.

The password for server access is: [provided separately]

File Structure:

/root/final/
- submit_spark_job.sh
- main.scala
- hp_geo.py

hdfs:/user/root/
- session.json
- honey4_out.json/part-00000-cde36a09-a4c0-4730-87d6-60def9df8239.json

The output summary statistics for the batch-processed data are placed in the appendix historical analysis section.

Statistics on Attacks

We analyzed some of our historical data to see if there was any evidence that some providers were more vulnerable than others. For this test we compared AWS and SoftLayer machines, during a specific period of 2 weeks. We found no evidence that there were differences on average (p-value = 0.581).

Hypothesis:

Null Hypothesis-There is no difference between the average attacks between Softlayer sensor and AWS sensor

Alternative Hypothesis-There is difference between the average attacks between Softlayer sensor and AWS sensor.

Method:

Compute the number of attacks for 5 sensors in Softlayer and AWS from July 13th 2017 to August 14th 2017.

Result:

```
-------------------------------------------------------------------------------
Comparing SL and AWS Vulnerability
-------------------------------------------------------------------------------

Total number of records in database is 2375133.
Total number of attacks for period under study (2 weeks) is 925406.
Total number of attacks on 5 SL machines is 398850.
Average number of attacks per machine is 79770.00.
Total number of attacks on 5 AWS machines is 284065.
Average number of attacks per machine is 56813.00.
2 Tailed T-Test:
The t-statistic is 0.575 and the p-value is 0.581
If we assume unequal variances than the t-statistic is 0.575 and the p-value is
0.584.
```

Conclusion

1. We found that our honey pots and sensors were subjected to a significant number of attacks (2.7 million within one month)
2. Most source IPs originated in China. Other top countries include USA, Russia and France.
3. Frequent cities were Nanjing, Shenzhen, Beijing and Guangzhou.
4. Ports attacked 445, 5060 and 22.
5. Where logins were attempted, most popular login ids were
6. Password was 12345, 1234, default, 1234567 and other simple combination of numbers and strings.
7. No statistical evidence that there was a difference between AWS and SL (however, for AWS one has to turn on global IP access.)

Annex: Technical Notes

Code for T-tests

```
# Program to do comparisons in number of attacks

# IDs for SL VS
id_ibm = [
"3003b20c-67ae-11e7-9428-0654b098fe83",
"4968a73e-67ae-11e7-9428-0654b098fe83",
"b04a1fee-67b0-11e7-9428-0654b098fe83",
"cf0febe8-67b0-11e7-9428-0654b098fe83",
"4fbc59f6-67b2-11e7-9428-0654b098fe83"
]

# IDs for AWS VS
id_aws = [
"dc8ac93c-7813-11e7-9428-0654b098fe83",
"08a40324-782a-11e7-9428-0654b098fe83",
"c276c020-782a-11e7-9428-0654b098fe83",
"5a3cc8a0-782b-11e7-9428-0654b098fe83",
"3c186e46-782c-11e7-9428-0654b098fe83"
]


# Import libraries
import pymongo
from pymongo import MongoClient
import pprint
import datetime
import numpy as np
import scipy.stats as stats
import math

# Define Mongo DB access
client = MongoClient()
db = client.mnemosyne
collection = db.hpfeed

# Define json template for hpfeed
hpfeed = {
      "_id" : "59673e953167c67ef223f2a8",
```

```
        "ident" : "4968a73e-67ae-11e7-9428-0654b098fe83",
        "timestamp" : "2017-07-13T09:34:12.999Z",
        "normalized" : "true",
        "payload" : {
              "local_host" : "169.50.175.2",
              "connection_type" : "reject",
              "connection_protocol" : "pcap",
              "remote_port" : 60111,
              "local_port" : 23,
              "remote_hostname" : "",
              "connection_transport" : "tcp",
              "remote_host" : "177.40.229.135"
         },
        "channel" : "dionaea.connections"
}
```


```
print "-"*78
print "Comparing SL and AWS Vulnerability"
print "-"*78

# Total number of records in database
hpfeeds = db.hpfeed
print "\nTotal number of records in database is %d." %hpfeeds.count()

# Define time period to be studied

start = datetime.datetime(2017,7,31,12,0,0)
end = datetime.datetime(2017,8,14,12,0,0) # Two weeks later

num1 = hpfeeds.find({"timestamp": {"$lt": end, "$gte": start}}).count()
print "Total number of attacks for period under study (2 weeks) is %d. " %num1

# Count number of attacks and run test

ibm = np.arange(5)
for i in range(1,6):
  ibm[i-1] = hpfeeds.find({
      "timestamp": {"$lt": end, "$gte": start},
      "ident": id_ibm[i-1]}).count()
print "Total number of attacks on 5 SL machines is %d."  %ibm.sum()
print "Average number of attacks per machine is %.2f." %ibm.mean()

aws = np.arange(5)
for i in range(1,6):
  aws[i-1] = hpfeeds.find({
      "timestamp": {"$lt": end, "$gte": start},
      "ident": id_aws[i-1]}).count()
```

```
print "Total number of attacks on 5 AWS machines is %d." %aws.sum()
print "Average number of attacks per machine is %.2f." %aws.mean()
print "2 Tailed T-Test:"

# Two sample t-test
two_sample = stats.ttest_ind(ibm,aws)

print "The t-statistic is %.3f and the p-value is %.3f" %two_sample

two_sample_diff_var = stats.ttest_ind(ibm, aws, equal_var=False)

print "If we assume unequal variances than the t-statistic is %.3f and the
p-value is %.3f." % two_sample_diff_var
```

Historical Analysis Result

```
The Top Attacking Countries are:

+------------+------+
|country_code| count|
+------------+------+
|          CN|558494|
|          US|290705|
|          RU|257634|
|          FR|154363|
|          LT|124593|
|          CA|108683|
|          IN| 88305|
|          ID| 79673|
|          VN| 79106|
|          BR| 75634|
|          TW| 68057|
|          UA| 66996|
|          DE| 62465|
|          ES| 51228|
|          TR| 44586|
|          GB| 43441|
|          AM| 39780|
|          HK| 37777|
|          VE| 37762|
|          CZ| 35020|
+------------+------+
```

```
Within China, Attacks are Coming
From These Cities:
+---------+-----+
|     city|count|
+---------+-----+
| Shenzhen|73786|
|  Nanjing|57721|
|  Beijing|54233|
|Zhengzhou|43186|
|Guangzhou|36345|
| Hangzhou|35315|
|    Jinan|25997|
| Nanchang|20352|
| Shanghai|17806|
| Shenyang|16548|
|     null|16136|
|    Hebei|14775|
|    Wuhan| 9673|
|   Fuzhou| 9417|
|  Chengdu| 9065|
|     Xian| 7709|
|  Kunming| 6785|
|   Jinhua| 6550|
|    Hefei| 6494|
| Shaoxing| 5865|
+---------+-----+
 Within Russia, Attacks are Coming
From These Cities
```

city	count
Moscow	76697
null	28112
Chelyabinsk	18767
Saint Petersburg	16936
Vladivostok	15214
Dimitrovgrad	8865
Yekaterinburg	4164
Pes	3208
Krasnodar	2615
Sakmara	2601
Ufa	2342
Voronezh	2046
Saratov	2041
Orenburg	1858
Ulyanovsk	1777
Cheboksary	1684
Novosibirsk	1565
Kazan	1506
Omsk	1451
Volgograd	1446

These Ports Are Being Attacked The Most

destination_port	count
445	964342
5060	297269
22	178122
80	178088
23	134045
3306	52540
1433	34772
10000	24095
3389	16229
139	9436
2323	9085
8080	7215
6881	6536
5358	6312
443	4663
25194	3548
1	2843
14415	2752
8545	2693
21	2317

The Greatest Number of Attacks is from These IP Addresses

source_ip	count
93.115.28.176	88232
192.95.29.173	64756
91.76.180.81	48917
192.230.86.226	43382
145.239.121.91	41893
51.254.167.59	40890
178.160.133.227	37645
93.115.26.10	29998
189.38.115.13	29919
145.239.118.160	27862
217.182.102.235	25039
125.46.45.110	23077
37.187.198.104	20647
209.9.53.48	20367
116.31.116.17	19370
37.187.76.119	18993
178.157.81.139	16682
185.57.30.187	15045
127.0.0.1	13614
116.31.116.14	12563

These Protocols Are Being Used Most in Attacks

protocol	count
smbd	953603
pcap	742083
TftpServerHandler	514387
SipSession	169552
SipCall	123436
ssh	120248
mysqld	49836
mssqld	32510
httpd	26084
TCP	7268
UDP	4649
RtpUdpStream	2520
ftpd	2205
microsoft-ds	628
epmapper	475
ftpdatalisten	192
emulation	19
mirrorc	9
mirrord	9
ICMP	5

Streaming analysis result

```
Aug 19th 2017 16:44 to 16:45
Attacks info in the last 10 seconds:
Country Info
####################################
###################
1 attack come from India
City Info
####################################
###################
1 attack come from Mumbai
port Info
####################################
###################
1 attack at 445
Honeypot info
####################################
###################
1 attacks are dionaea
Destination IP info
####################################
###################
1 attacks are 184.173.18.155
Attacks info in the last 10 seconds:
Country Info
####################################
###################
2 attack come from India
City Info
####################################
###################
2 attack come from Mumbai
port Info
####################################
###################
2 attack at 445
Honeypot info
####################################
###################
2 attacks are dionaea
Destination IP info
####################################
###################
2 attacks are 184.173.18.158
Attacks info in the last 10 seconds:
Country Info
####################################
###################
1 attack come from India
City Info
####################################
###################
1 attack come from Mumbai
port Info
####################################
###################
1 attack at 445
Honeypot info
####################################
###################
1 attacks are dionaea
Destination IP info
####################################
###################
1 attacks are 184.173.18.158
Attacks info in the last 10 seconds:
Country Info
####################################
###################
2 attack come from United States
City Info
####################################
###################
2 attack come from None
port Info
####################################
###################
2 attack at 445
Honeypot info
####################################
###################
2 attacks are dionaea
Destination IP info
####################################
###################
2 attacks are 184.173.18.158
Attacks info in the last 10 seconds:
Country Info
####################################
###################
1 attack come from Taiwan
```

```
City Info
####################################
###################
1 attack come from None
port Info
####################################
###################
1 attack at 445
Honeypot info
####################################
###################
1 attacks are dionaea
Destination IP info
####################################
###################
1 attacks are 184.173.18.158
Attacks info in the last 10 seconds:
Country Info
####################################
###################
1 attack come from India
City Info
####################################
###################
1 attack come from Gandhinagar
port Info
####################################
###################
1 attack at 445
Honeypot info
####################################
###################
1 attacks are dionaea
Destination IP info
####################################
###################
1 attacks are 184.173.18.155
Attacks info in the last 10 seconds:
Country Info
####################################
###################
1 attack come from Bangladesh
City Info
####################################
###################
1 attack come from Dhaka
port Info
####################################
###################
1 attack at 445
Honeypot info
####################################
###################
1 attacks are dionaea
```

References

Here are some references on the Modern Honey Pot, official/semi-official:

* The Github page [GitHub - threatstream/mhn: Modern Honey Network](https://github.com/threatstream/mhn)
* Threatstream info on MHN (I think Anomali has taken over from Threatstream) [mhn](https://threatstream.github.io/mhn/)
* Anomali pages on MHN [Modern Honey Net | Manage & Deploy Honeypot Sensors](https://www.anomali.com/platform/modern-honey-net)
* Some use cases and tips on what to deploy [Modern Honey Network & Honeypot Use Cases](https://www.anomali.com/platform/modern-honey-net/mhn-usage-cases)
* Webinar [4 Ways to Get the Most Out of the Modern Honey Network | Anomali](https://www.anomali.com/blog/4-ways-to-get-the-most-out-of-the-modern-honey-network)

Other resources:

* https://medium.com/@theroxyd/honeypot-farming-setup-mhn-f07d241fcac6
* https://jerrygamblin.com/2017/05/29/build-your-own-honeypot-network-in-under-an-hour/
* https://zeltser.com/modern-honey-network-experiments/
* https://www.blackmoreops.com/2016/05/06/setup-honeypot-in-kali-linux/

Troubleshooting:

* [MHN Troubleshooting Guide · threatstream/mhn Wiki · GitHub](https://github.com/threatstream/mhn/wiki/MHN-Troubleshooting-Guide)