An analysis of honeypot programs and the attack data collected

Chris Moore[1] and Ameer Al-Nemrat[2]

[1]University of St Mark & St John, Plymouth, England
cmoore@marjon.ac.uk
[2]University of East London, London, England
a.al-nemrat@uel.ac.uk

Abstract. Honeypots are computers specifically deployed to be a resource that is expected to be attacked or compromised. While the attacker is distracted with the decoy computer system we learn about the attacker and their methods of attack. From the information gained about the attacks we can then review and harden out security systems. Compared to an Intrusion Detection System (IDS) which may trigger false positives, we take the standpoint that nobody ought to be interacting with the decoy computer; therefore we regard all interactions to be of value and worth investigation. A sample of honeypots are evaluated and one selected to collect attacks. The captured attacks reveal the source IP address of the attacker and the service port under attack. Attacks where the exploit attempts to deploy a binary can capture the code, and automatically submit it for analysis to sandboxes such as VirusTotal.

Keywords: Honeypots · Security · Intrusion Detection

1 Introduction

As a security monitor, Spitzner [1] gives the definition “a honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource”. A honeypot is a tool that does not have any authorised use, so any interaction is deemed to be of malicious intent.

We are familiar with firewalls and Intrusion Detection and Prevention Systems (IDPS) as methods of network defence; however a honeypot offers a different approach. Kaur, Malhotra and Singh [2] argue that firewalls logging all traffic would collect an overwhelming amount of information that an administrator would find prohibitive to analyse; a honeypot however would only log attacks to a host, Joshi & Sardana [3] state that the small data sets are easier to manage and analyse.

Their work also argues that Intrusion Detection is not a definitive security solution, as IDPS have high occurrences of false positive alerts requiring extensive tuning of the IDPS. Similarly undetected intrusions (false negatives) also occur when malicious activity is not detected as a signature of the malware available to identify yet unknown or novel threats. From the viewpoint that a honeypot does not advertise any resources for regular use, any interaction with the honeypot is assumed to be an intrusion and worthy of further investigation. A honeypot can be used to check if intruders are rattling your door locks to test your home security, and the logging is analogous to the use of wet cement for detecting footprints.

2 Background

The concept of deceiving an attacker as a network security method has origins with the work documented of Cliff Stoll in 1986 [4] where he discovered an attacker had infiltrated the systems at Lawrence Berkeley Lab where Stoll was an astronomer. Rather than locking the attacker out of the system, Stoll decided to allow the attacker to stay on the system enabling Stoll to covertly learn more about the attacker and his techniques.

Where Stoll looked to track a specific user, in 1990, Bill Cheswick [5] intentionally built a system to be attacked. Cheswick created a 'Jail' that presented a contained environment to the attacker and appeared to include vulnerabilities that allowed him to study threats and how they were compromised. With his system in place, Cheswick's work describes how it was possible to monitor a user's techniques as they infiltrated system vulnerabilities and gained control.

The idea of emulating system vulnerabilities was developed in 1997 by Fred Cohen with his Deception Toolkit (DTK) [6], regarded as the first publically available honeypot that could be downloaded and deployed on one's own systems. As well as logging the attacker's interactions, the DTK was designed to deceive and psychologically confuse them. One method was actually to use port 365 as a deception port; suggesting attackers recognise this port was in use, and therefore avoid attacking that system, or utilised as a double-bluff and advertise this port on a production system.

The term honeypot was first used by Lance Spitzner in 1999 [7] in the series of papers for the Honeynet project, 'Know your Enemy' asserted that based on the knowledge you gain discovering what attackers are looking for and their tools, this knowledge can be used to secure and protect your systems. The work of the Honeynet project established awareness and value of honeypot systems. Spitzner distinguished different interaction types of honeypots, High-interaction and Low-interaction. High-

interaction honeypots offer a vulnerable real operating system allowing an attacker or worm to interact with the system, however all interactions are captured for analysis and can be used to detect zero-day attacks [8]. However high-interaction honeypots need a lot of monitoring and carry the risk that they can be compromised. Low-interaction honeypots simply look to emulate some of the services to be attacked. This is a simpler system to create, requiring less code to create the honeypot. However as a subset of services are emulated some responses or lack of them may help the attacker determine that a fake system is being attacked rather than a full system. This absence of a full operating system however reduces the risk that that honeypot system can be compromised. Spitzner also characterises different ways to deploy a honeypot, research and production. In a research environment; typically operated by anti-malware research and government organisations, methods of unauthorised access being employed by the hacker community can be gathered to gain knowledge of new attacks, leading to developments to defend against those attacks. Production honeypots are located alongside other servers on a production network to increase the security monitoring within the network. Generally production honeypots are low-interaction, accordingly capturing limited information compared to a research honeypot.

In the 2000s, network worms such as Code Red began to proliferate across the Internet, honeypots were an effective mechanism to capture the works and allow their attack to be analysed. By the mid-2000s the value of honeypots is becoming recognised, Provos & Holz [9] advocate virtual honeypots, allowing multiple honeypots to be deployed on a single system. This virtual strategy provides a more efficient method to deploy a collection of honeypot systems. If a virtual honeypot were compromised it could be restored efficiently.

Many different honeypot tools and services have been listed by the Honeynet project [10], the research aims to evaluate a range of honeypot solutions and to determine the most suitable candidate to perform attack monitoring.

3 Methodology

There are two themes to the collection of data about honeypots in this study, first is an evaluation of available honeypot systems to identify an appropriate product to perform the data collection for the second theme where we analyse the information collected by a suitable honeypot system.

3.1 Production Honeypot Products

Over the research period a variety of honeypot programs have been evaluated, some are Microsoft Windows based, while others run on a Linux distribution. Where a Linux distribution is used, the common base is Ubuntu 12.04. The Honeypot systems were given an appraisal on several criteria, such as the simplicity deploy, and the information from attacks reported.

BackOfficer Friendly. (BOF) is a simple Windows program dating from 1999, available from http://www.guardiansofjustice.com/diablo/Frames/Fileutil.htm. BOF was one of the first programs to give alerts when the system was probed for open ports. Basic replies to connections are returned, however it does provide a simple demonstration to determine how often a system is probed for intrusion.

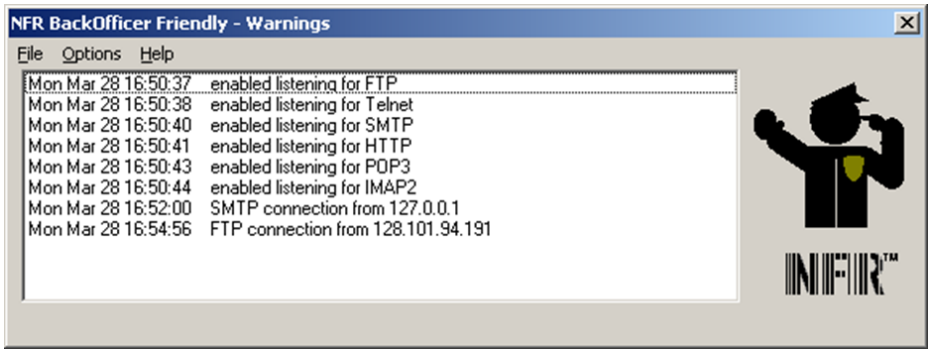

Fig. 1. Screen Display from BackOfficer Friendly

While the live output of BOF alerts are displayed (**Fig. 1**) this would require the operator to constantly monitor the display, more beneficial is the log output **Fig. 2** which can be used to review attacks made onto the system.

```
Mon May 18 20:13:17   Telnet connection from 115.75.209.123
Mon May 18 20:13:21   Telnet login attempted from 115.75.209.123: user: root, password: admin
Mon May 18 20:13:25   Telnet connection from 115.75.209.123
Mon May 18 20:13:29   Telnet login attempted from 115.75.209.123: user: root, password: Admin
Mon May 18 20:13:33   Telnet connection from 115.75.209.123
Mon May 18 20:14:09   HTTP request from 115.75.209.123: POST /cgi-bin/firmwarecfg
Mon May 18 23:48:41   SMTP connection from 118.165.71.122
Tue May 19 06:17:31   SMTP connection from 205.209.161.229
Tue May 19 08:04:51   HTTP bogus request from 79.110.133.215: OPTIONS / HTTP/1.1
Tue May 19 09:17:58   IMAP2 connection from e17.10.140.196
Tue May 19 09:22:18   HTTP bogus request from 189.16.81.166: OPTIONS / HTTP/1.1
Tue May 19 11:16:26   SMTP connection from 118.165.86.53
Tue May 19 13:57:27   SMTP connection from 61.247.233.92
Tue May 19 14:19:55   SMTP connection from 205.209.161.229
Tue May 19 15:41:31   HTTP bogus request from 79.26.13.74: OPTIONS / HTTP/1.1
Tue May 19 16:59:59   SMTP connection from 205.209.161.229
Tue May 19 20:11:01   HTTP request from 194.177.98.221: GET //appserv/main.php?appserv_root=http://ematrimoniale.go.ro/a.txt??
Tue May 19 20:11:29   HTTP request from 194.177.98.221: GET //appserv/main.php?appserv_root=http://ematrimoniale.go.ro/a.txt??
Tue May 19 22:16:39   HTTP bogus request from 95.e19.233.38: OPTIONS / HTTP/1.1
Tue May 19 22:16:53   HTTP bogus request from 219.167.137.187: OPTIONS / HTTP/1.1
Wed May 20 01:11:57   Telnet connection from 59.184.28.85
Wed May 20 01:12:00   Telnet login attempted from 59.184.28.85: user: root, password: admin
Wed May 20 01:12:03   Telnet connection from 59.184.28.85
Wed May 20 01:12:07   Telnet login attempted from 59.184.28.85: user: root, password: Admin
Wed May 20 01:12:10   Telnet connection from 59.184.28.85
Wed May 20 01:12:13   Telnet login attempted from 59.184.28.85: user: root, password: password
Wed May 20 01:12:16   Telnet connection from 59.184.28.85
Wed May 20 01:12:39   Telnet login attempted from 59.184.28.85: user: admin, password: admin
Wed May 20 01:12:42   HTTP request from 59.184.28.85: POST /cgi-bin/firmwarecfg
Wed May 20 04:47:20   SMTP connection from 205.209.161.236
Wed May 20 10:11:44   HTTP request from 87.208.35.87: GET /w00tw00t.at.ISC.SANS.Dfind
```

Fig. 2. Log obtained from BackOfficer Friendly

HoneyBOT. is a commercial Windows product; however an academic version is available for the evaluation period. Simple to deploy and initially detects at lot of background noise traffic, however the ports commonly seen on a network can be turned off, leaving just the unusual ports to be monitored for unusual activity. **Fig. 3** demonstrates the information captured by HoneyBOT.

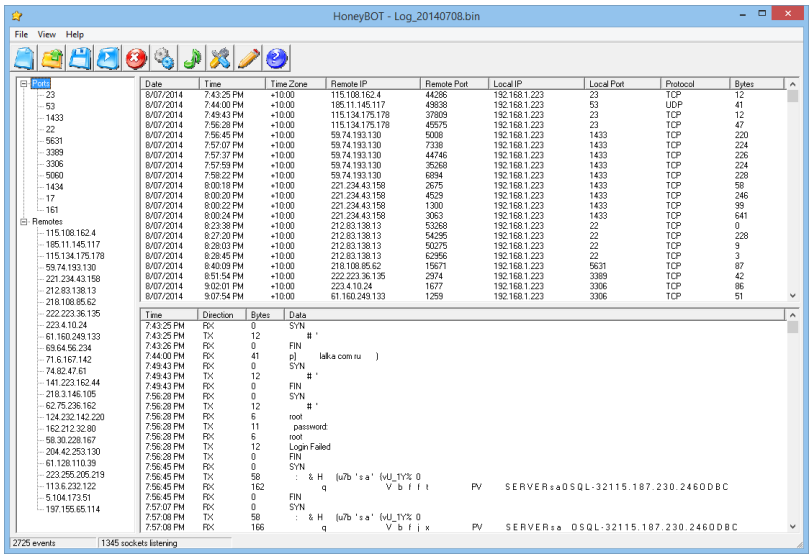

Fig. 3. Screen display from HoneyBOT

Nepenthes. Turning to Linux distributions for honeypots, Provos & Holz, [9] suggest Nepenthes as a system to install as a virtual honeypot . Installation is possible by downloading and compiling the source code, however deployment packages are available to acquire the software. Unlike the previous Windows example, there is no graphical output to monitor attacks made onto the honeypot. Attack data is collected in a series of log files, **Fig. 4** shows an example from *nepenthes.log* ; of significance are events where a TCP connection is accepted. These lines can be parsed at the command line and used to analyse the attack, better reports require the data to be exported into a spreadsheet for manipulation.

Nepenthes, however does more than just log the attempts to connect to the honeypot, it is able to imitate some of the operating system commands, and in the occurrence of an intrusion attempting to deploy malware, the executable code is captured. Captured code can be submitted to a sandbox service to analyse the malware.

```
[09102009 15:45:40 spam net handler] <in virtual int32_t nepenthes::TCPSocket::doRecv()>
[09102009 15:45:40 spam mgr event] <in virtual uint32_t nepenthes::EventManager::handleEvent
(nepenthes::Event*)>
[09102009 15:45:40 spam net handler] doRecv() 5
[09102009 15:45:50 debug net mgr] Socket TCP  (bind) 0.0.0.0:0 -> i1.i2.i3.i4:25
     DialogueFactory Watch Factory create Watch Dialogues could Accept a Connection
[09102009 15:45:50 spam net handler] <in virtual nepenthes::Socket*
nepenthes::TCPSocket::acceptConnection()>
[09102009 15:45:50 spam net handler] Socket TCP  (accept) e1.e2.e3.e4:53781 -> i1.i2.i3.i4:25
[09102009 15:45:50 spam net handler] Adding Dialogue Watch Factory
[09102009 15:45:50 spam mgr event] <in virtual uint32_t nepenthes::EventManager::handleEvent
(nepenthes::Event*)>
[09102009 15:45:50 debug net mgr] Accepted Connection Socket TCP  (accept) e1.e2.e3.e4:53781 ->
i1.i2.i3.i4:25
32 Sockets in list
[09102009 15:45:51 spam net handler] <in virtual int32_t nepenthes::TCPSocket::doRecv()>
[09102009 15:45:51 spam mgr event] <in virtual uint32_t nepenthes::EventManager::handleEvent
(nepenthes::Event*)>
[09102009 15:45:51 spam net handler] doRecv() 2
[09102009 15:45:51 spam net handler] <in virtual int32_t nepenthes::TCPSocket::doRecv()>
[09102009 15:45:51 spam mgr event] <in virtual uint32_t nepenthes::EventManager::handleEvent
(nepenthes::Event*)>
[09102009 15:45:51 spam net handler] doRecv() 1
```

Fig. 4. Output from Nepenthes.Log file

Dionaea. Released as the successor to Nepenthes, however the author's experience of the download, compilation and installation cycle has found this Dionaea to be less straightforward and ultimately less successful than the previously tested honeypot solutions. Work by Andy Smith [11] on automating this process made getting a Dionaea honeypot working much more efficiently. Once running, Dionaea collects attack information into logfiles and a SQLite database that requires further inspection to understand the attacks collected by the system.

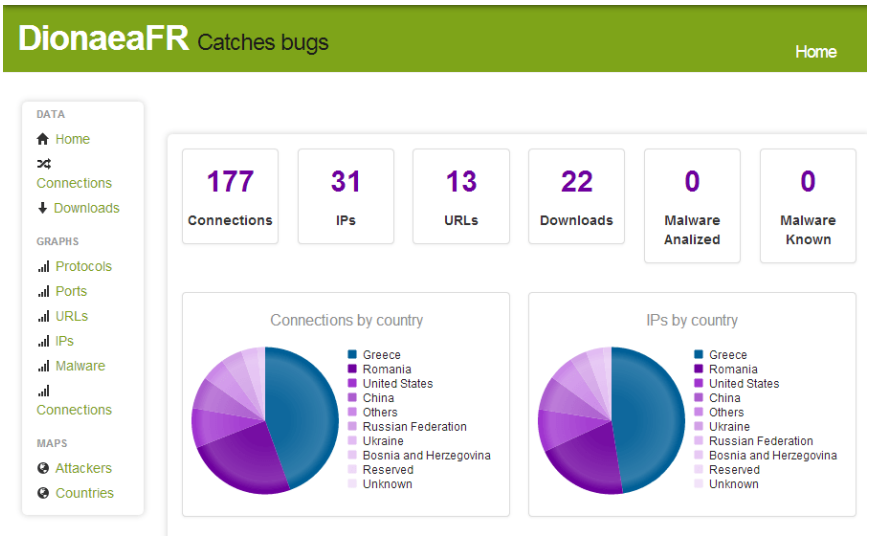

Fig. 5. DionaeaFR Graphical Interface

Python scripts have been developed to gain graphical representations of the data from the command line, however additional extensions to Dionaea have provided a graphical front end to enable an operator to get an overview of attacks on the system, **Fig. 5** illustrates some of information available. Recently a project from ThreatStream, has introduced Modern Honey Network [12] as a honeypot management system to assist in the deployment of honeypots. Dionaea is just one of several honeypots that MHN allows a user to deploy.

Kippo. Many intrusions look to take remote control of the attacked system via the Secure Shell command interface. Kippo is a medium interaction honeypot and concentrates on emulating the SSH interface to a higher level of sophistication than Dionaea offers. Kippo is included as part of the HoneyDrive virtual appliance, that also includes Kippo-Graph [13] scripts allow the visual display of the data collected by Kippo. For example **Fig. 6** shows the attempted username / password combinations are captured, and **Fig. 7** utilises a Geographical mapping of source IP address to locate the origin of an attack.

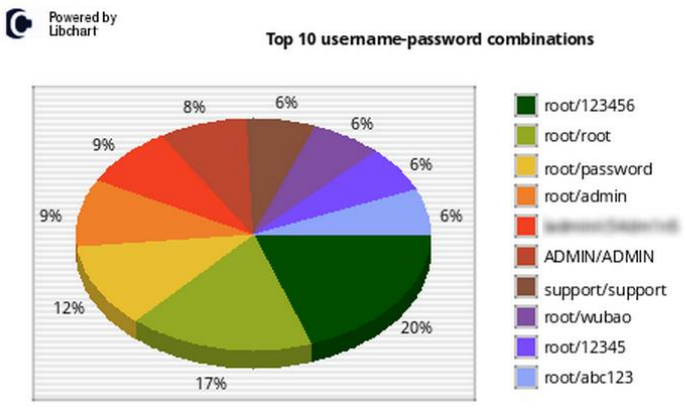

Fig. 6. Kippo-Graph visualisation of collected username / password combinations

Fig. 7. Kippo-Graph Geo IP representation of source address of attacks

During the trial, the majority of attacks are reported as originating from China and Japan.

Evaluation. The HoneyDrive distribution gives a rapid method to begin collecting attack data and easy to understand graphical display, however for this study, the MHN deployment of Dionaea was selected to collect attack data as this offered simpler access to the attack data.

3.2 Attack data collection

The honeypot experimental system was created on a virtual installation of Ubuntu 12.04 under VMware Workstation with a bridged network configuration allowing the virtual machine to be addressed on the network. After installation of Ubuntu, the Modern Honey Network –11-- platform was installed and configured following default settings. The honeypot system was then connected to a DMZ arm of the

firewall and configured with a public address. Once the MHN platform was running, it offers a choice of scripts to deploy under the MHN platform. A Dionaea honeypot [14] was selected and the command generated was executed on the Ubuntu command line.

This initiates the installation of a Dionaea honeypot. Once a honeypot is deployed, there is a wait to see when an attack is detected, although an interaction can be forced with an nmap [15] scan against the IP address of the MHN system. When connected to a public IP address the first attack can come in minutes after deployment.

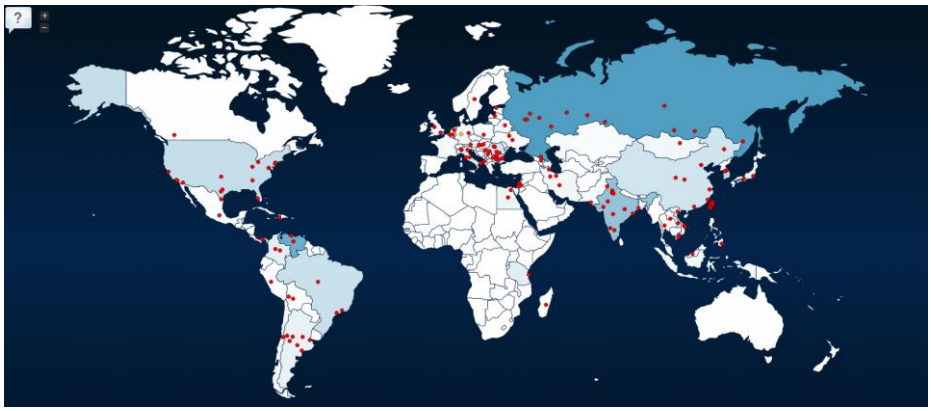

Fig. 8. Live map of attacks

MHN offers a map representation of live attacks, see **Fig. 8** for an example, an interesting graphical overview of the attacks being logged by the honeypot, however the database of the attack information can be extracted for deeper analysis. Additionally if you have a VirusTotal account, the Dionaea configuration can be updated with a suitable API Key and captured malware can be automatically directed to VirusTotal for analysis.

For the data analysis, the R statistical programing language [16] with the R-Studio [17] interface was installed. R is a free system allowing the user to write scripts to analyse the data. The data from MHN was exported via FTP to the R workstation for analysis. Scripts were written to remove unrequired information, and manipulate the timestamp fields by reformatting strings to date-time objects.

Among the fields available for analysis were: timestamp, source IP address and destination port. R scripts were developed to summarise the data and display bar charts.

4 Data Results

4.1 A. Source IP Address

Fig. 9 shows that over the period of the experiment, the IP Address 188.165.238.186 attacked the DMZ honeypot 13708 times. This is noteworthy as the next most frequent source IP addresses generated around 500 attacks. The count of attacks was almost 30 times greater than the average number of attacks for a typical address in the remaining 19 of the top 20.

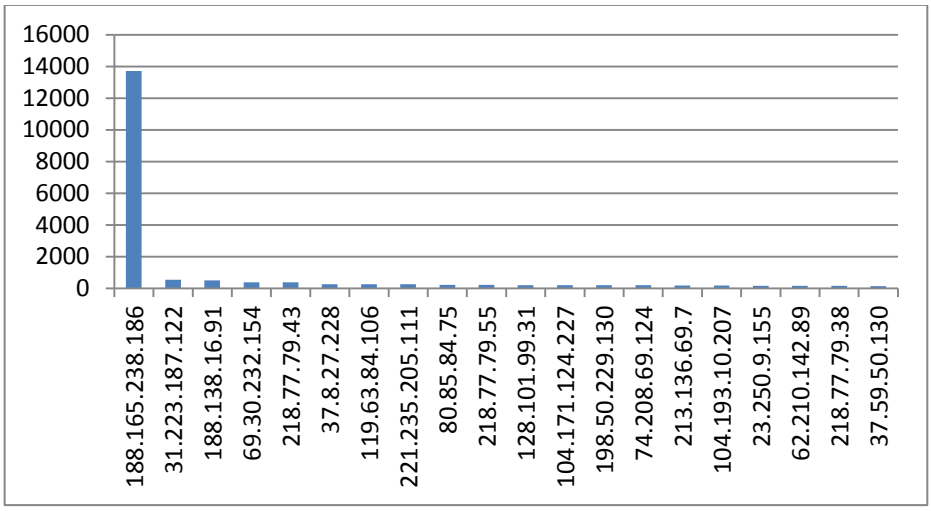

Fig. 9. Analysis of Source Address

4.2 Destination Port

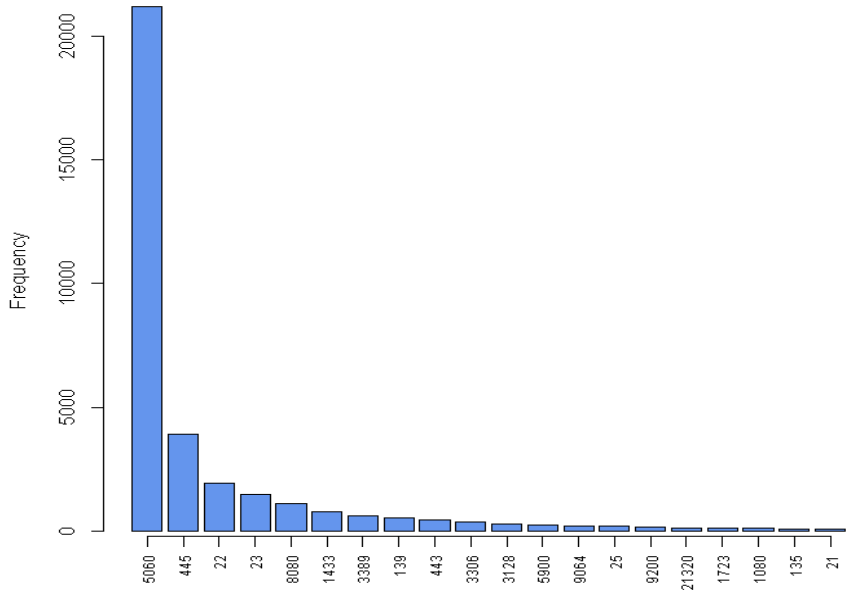

Fig. 10. Analysis of top 20 Destination Ports

Fig. 10 shows the destination port of attacks over the period of the experiment, notably Session Initiation Protocol (SIP) port 5060 attracted a relatively large number of attacks compared against other attacks looking for vulnerabilities in Microsoft Directory Services (445) SSH Remote Login Protocol (22) and Telnet (23).

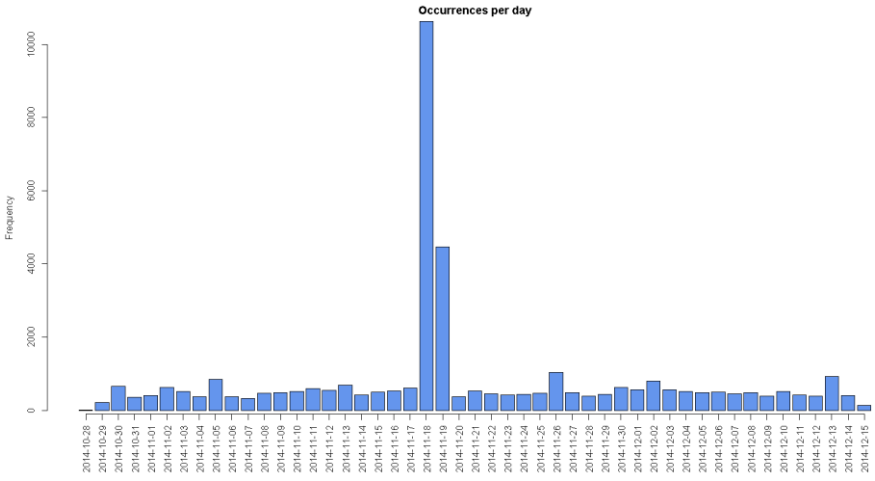

Fig. 11. Analysis data timestamp

4.3 Timestamp

Fig. 11 shows an average of around 500 attacks per day, however there is a distinct peak around 18th and 19th November 2014.

5 Data Analysis

The honeypot was connected to a public facing IP address and gathered a lot of unsolicited interest from remote devices. Regarding the number of attacks per day, we can establish a baseline of the number of expected attacks, and therefore be aware when this threshold is exceeded would statistically indicate a more persistent stream of attacks. For this honeypot experiment number of attacks per day (n=49) averaged 484 (σ157) in the experiment period 28th October to 15 December 2014. This implies that the mean of the population of attacks onto experimental honeypot, with 95% confidence, is in the interval of 484 ±(1.96 x 157), which is from 176 to 792. This would suggest a exceeding a threshold of around 800 attacks per day would indicate extraordinary activity and merit further investigation.

The graphs indicate where peaks in the data occur, when regarding the results of timestamp, source and destination together we can establish that between the 18th and 19th November 2014, IP address 188.165.238.186 generated over 14,000 attacks on port 5060. This suggests a sustained attack against Session Initiation Protocol (SIP), a protocol for handling telephone calls over an IP network. Our organisation currently does not use SIP, so this traffic would be denied at the firewall, however the awareness that SIP vulnerabilities are actively sought [18] would inform the network manager to harden any SIP services that may be installed in future.

6 Summary and Conclusion

There are many varieties of honeypot deployments, and we have established that an implementation of Dionaea allowed the collection of a rich set of attack data. Other well established honeypots from the Honeynet projects page need to be examined for comparison, during the period of the study extra honeypot features became available, such as Splunk in MHN and Kibana for Kippo.

The honeypot deployment only offers limited vision of attacks to its IP address rather than the whole sub-network. Nevertheless the honeypot indicated the number of attacks a public facing IP address attracts, and emphasises the need for security on these systems. Evidence specifying the source of threats and how they attempt to

infiltrate the system provide valuable information to the network manager, who should act on this information to harden their security.

This experiment suggests work to install R on the same machine as the MHN deployment, removing the delay in transferring the attack data to an analysis system therefore allowing timelier in depth analysis of the attack data.

Acknowledgements . Thanks to Ameer Al-Nemrat, University of East London for encouragement on developing this paper and support from the Computing and Media Services team at the University of St Mark & St John. Finally, thank you to the editors and peer reviewers for their time, expertise and guidance on this paper.

References

1. Spitzner, L.: Honeypots: Tracking Hackers. Addison-Wesley Educational Publishers Inc, Boston (2003)

2. Kaur, T., Malhotra, V., Singh, D.: Comparison of network security tools-Firewall, Intrusion Detection System and Honeypot. International Journal of Enhanced Research in Science Technology & Engineering, 200-204 (2014)

3. Joshi, R., Sardana, A.: Honeypots: A New Paradigm to Information Security. Science Publishers, Enfield, N.H. (2011)

4. Stoll, C.: The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage. Pocket Books, New York (2007)

5. Cheswick, B.: An Evening with Berferd. In Denning, D., Denning, P., eds. : Internet besieged. ACM Press/Addison-Wesley Publishing Co., New York (1998) 103-116

6. Cohen, F.: The Deception Toolkit Home Page and Mailing List. In: All.Net. (Accessed March 30, 2015) Available at: http://www.all.net/dtk/

7. Honeynet Project: Know Your Enemy: III. In: Honeynet Project. (Accessed March 30, 2015) Available at: http://old.honeynet.org/papers/enemy3/

8. Göbel, J., Dewald, A.: Client-Honeypots: Exploring Malicious Websites.

Oldenbourg Verlag, München (2011)

9. Provos, N., Holz, T.: Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Pearson Education (2007)

10. Honeynet Project: Projects. (Accessed June 2015) Available at: http://www.honeynet.org/project

11. Quick install of Dionaea on Ubuntu. In: Andy Smith's Blog. (Accessed May 2015) Available at: http://andrewmichaelsmith.com/2012/02/quick-install-of-dionaea-on-ubuntu/

12. Trost, J.: Modern Honey Network. In: ThreatStream. (Accessed May 2015) Available at: https://www.threatstream.com/blog/mhn-modern-honey-network

13. Kippo-Graph. In: BruteForce. (Accessed May 2015) Available at: https://bruteforce.gr/kippo-graph

14. Dionaea - Catches Bugs. In: Carnivore. (Accessed March 2015) Available at: http://dionaea.carnivore.it/

15. Insecure.org: Nmap Security Scanner. In: Nmap.org. (Accessed March 2015) Available at: http://nmap.org/

16. R Project: Getting Started. In: The R Project for Statistical Computing. (Accessed March 2015) Available at: http://www.r-project.org/

17. RStudio: Take control of your R code. In: RStudio. (Accessed March 2015) Available at: http://www.rstudio.com/products/rstudio/

18. Popeskic, : Attack on SIP protocol – VoIP Vulnerability. In: How does Internet work. (Accessed June 1, 2012) Available at: http://howdoesinternetwork.com/2012/voip-sip-attack