Correlation of Cyber Threat Intelligence Data Across Global Honeypots

Jay Thom
Computer Science Department
University of Nevada Reno
Reno, Nevada, USA
jthom@unr.edu

Yash Shah
Computer Science Department
University of Nevada Reno
Reno, Nevada, USA
yashs@nevada.unr.edu

Shamik Sengupta
Computer Science Department
University of Nevada Reno
Reno, Nevada, USA
ssengupta@unr.edu

2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) | 978-1-6654-1490-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCWC51732.2021.9376038

***Abstract*—Today's global network is filled with attackers both live and automated seeking to identify and compromise vulnerable devices, with initial scanning and attack activity occurring within minutes or even seconds of being connected to the Internet. To better understand these events, honeypots can be deployed to monitor and log activity by simulating actual Internet facing services such as SSH, Telnet, HTTP, or FTP, and malicious activity can be logged as attempts are made to compromise them. In this study six multi-service honeypots are deployed in locations around the globe to collect and catalog traffic over a period of several months between March and December, 2020. Analysis is performed on various characteristics including source and destination IP addresses and port numbers, usernames and passwords utilized, commands executed, and types of files downloaded. In addition, Cowrie log data is restructured to observe individual attacker sessions, study command sequences, and monitor tunneling activity. This data is then correlated across honeypots to compare attack and traffic patterns with the goal of learning more about the tactics being employed. By gathering data gathered from geographically separate zones over a long period of time a greater understanding can be developed regarding attacker intent and methodology, can aid in the development of effective approaches to identifying malicious behavior and attack sources, and can serve as a cyber-threat intelligence feed.**

***Index Terms*—honeypots, malicious traffic, botnet, Cowrie, ssh-attacks, cyber-threat intelligence**

I. Introduction

Scanning and brute-force attacks on Internet facing services such as SSH, Telnet, FTP, and HTTP, have become so common that within minutes or even seconds of connecting devices to the global network, attacks are being launched to compromise them. An unwary administrator might find their work is actively being compromised even as they are in the process of setting it up. What are they after, and what are they trying to accomplish? The answer is, quite a variety of things. Since the Mirai malware made headlines in 2016, the code behind the botnet has been released as open-source, and has been modified by various hackers seeking to build their own zombie-armies [14]. Evidence of this code and the sequence of actions indicating its installation are evident in data collected by SSH honeypots and account for a significant portion of scan and attack activity, along with a host of other botnet-related malware. There are also various random brute-force attempts in order to gain root access to the host device or to specific services, as well as non-malicious scanning activity.

978-0-7381-4394-1/21/$31.00 ©2021 IEEE

Honeypots are a useful tool for capturing such events by providing a realistic environment for attack, and then logging activity for later analysis. They have been extensively employed for such tasks as attack pattern comparisons, attack frequency analysis, attack origin analysis, root cause identification, and risk assessment [1]. For SSH attacks Cowrie is a popular medium interaction sandbox environment which provides a simulated file system and shell, and allows access with random credentials after a variable number of brute force attempts. Once inside, an attacker is deceived into believing it has accessed a real system, and is observed while carrying out whatever their intention is; changing or creating files, downloading software, or altering passwords or user accounts. This type of environment is especially effective with bots, as they are automated and generally less able to identify the environment as a honeypot than a live attacker would be.

Are there patterns in the activities seen in these environments, and are the attacks coordinated, or completely random? Can similarities or differences be identified across honeypots distributed globally? This study utilizes a series of docker containers running Cowrie to detect SSH attacks without compromising the host machine in an effort to answer these questions. In addition to Cowrie, the honeypots also utilize an Apache web server and an FTP server running in containerized environments, allowing for the collection of associated logs and tracking access attempts. Linux kernel logs from the host machines are also collected to monitor for compromise of the host device. Each honeypot is built on a Debian 10 virtual machine running in one of six Digital Ocean data centers located in geographically separate areas, including London, New York, Toronto, Amsterdam, Bangalore, and Singapore. Fake websites with domain names provided by Google Domains are utilized with names related to higher education in an attempt to attract specific kinds of traffic, although this feature has not been extensively developed thus far. Logs from these services are transferred using a cron tab and rsync on a nightly basis to a repository server where all logs are consolidated and analyzed. This study will look at data collected for the time period between March 2020 and December 2020, with the data being continuously amalgamated. Apache, FTP, and

Linux kernel logs are in their standard form (apache.json.log, ftp.json.log, kern.log), while Cowrie returns a .json or .log file containing a variety of tags, including event-ID, session-ID, source-IP, destination-IP, username and password (that were used to access the honeypot), source-port, destination-port, message (commands passed to the honeypot), file hashes, and a variety of other data points.

By collecting data from the honeypots over several months we observe data and patterns of activity, and attempt to draw correlations between honeypots and regions to learn more about attackers, their objectives, their methods, and intent. Standard .json log files are also restructured so that session-IDs rather than event-IDs are made the key value. This allows for the collection of command chains, and makes it easier to view attack patterns and analyze attacker behavior.

II. Related Works

Honeypots have been widely used for collecting and analyzing the activities of malicious actors. They provide an effective tool for observing attacker behavior, as it can be assumed no legitimate traffic should be exchanged with the honeypot services, and they have no production value [13] [20].

More than 67% of web servers and 71% of IoT devices connected to the Internet rely on Unix/Linux-based operating systems [24] [25]. In the work by Kambourakis et al [14], regular updates to firmware for these systems are often overlooked, leaving opportunities for malicious actors to develop methods for unauthorized access and remote manipulation. In addition, source code for many well-known and scalable exploits are publicly available, providing hackers with ample resources to bypass security measures and subvert vulnerable systems. Honeypots can be placed inside of a network as a distraction, drawing attackers away from valued resources, or as stand-alone services exposing vulnerable interfaces of compete services such as SSH, Telnet, HTTP, FTP, or SMTP. Services attempt to appear as legitimate to attackers, and log activity without implementing all of the service's logic and functionality, as shown by Bistarelli et al [20]. Kumar [6] and Kyriakou [8] et al demonstrate the advantages of deploying multiple honeypot tools and utilizing containers to produce a lightweight multi-service honeypot on a single virtual machine, server, or lightweight device (i.e. Raspberry Pi). In [3], [9], [12] examples of deployment and data collection from honeypots are detailed, and the basic functions of a botnet malware are examined based on scanning practices and the order of commands executed by an attacker once logged in.

Several open-source honeypot applications are available to emulate common services, provide limited functionality, and automatically log activity. Vetterl et al discuss applications such as Kojoney and Klippo [1] Other applications such as Dionaea, Whaler, and Cowrie provide access to services such as SMB, HTTP, FTP, TFTP, MSSQL, MySQL, SIP, SSH and the Docker API. Narwocki et al [7] explain how by exposing the common ports for these services attackers performing random scans of the Internet are often attracted to them within minutes, and then perform brute-force attacks using dictionaries of common usernames and passwords to gain access.

High value data can be collected, and detailed analysis is required to learn more about attack behavior. Fraunholz et al [11] discuss analysis based on timing behavior by correlating the overall number of attacks with the number of unique IP addresses seen, as well as the correlating between the overall number of attacks with the number of attacks per unique IP address. Vakilinia et al [2] discuss capturing commonly used passwords from brute force attacks and utilizing them as a feed for cyber threat intelligence. Fan et al [4] develop attack profiles by applying attack information to analyze malicious activity in order to unveil intruder motives. Fraunholz et al [15] discuss the application of machine learning techniques for classifying attacks on honeypots.

A major concern is the fingerprinting and identification of deployed honeypots by attackers. Vetterl et al [18] present a generic technique for fingerprinting honeypots at Internet scale with a single TCP packet. They conduct Internet-wide scans and are able to identify 7605 honeypot instances across nine separate implementations. They also discover most honeypot instances are not properly updated, making them even easier for attackers to identify. McCaughey et al [10] note many open-source software tools are available to help identify honeypot devices that have been on the network for extended periods of time by noting timing differences between honeypots and actual machines. Vetterl et al [17] discuss a project wherein they scan the Internet and discover thousands of honeypot devices. They also cover some of the legal issues involved in "logging into" honeypot machines, even for the purpose of identifying them. Cabral et al [9] discuss how Cowrie in its standard state can be easily identified by attackers using nmap, Shodan, and OS fingerprinting, and requires modification to be effective. Finally, Pitman et al [16] discuss their tool that can quantify the ability of a honeypots to fingerprint its environment, capture valid data, deceive an adversary, and monitor itself and its surroundings.

To better understand attack behavior and to develop a more complete understanding of how adversaries are utilizing services left exposed by weak or default login credentials we collect traffic on a global scale over an extended period of time, both to amass a large body of data for the development of Cyber Threat Intelligence (CTI) tools, and to identify patterns in behavior as attackers access and utilize services presented by honeypots located in geographically separated regions.

III. System Implementation

To collect data on a global scale the Digital Ocean developer cloud is utilized, allowing for the deployment of Low-cost virtual machines placed in various data centers around the world. In order to limit resources and reduce cost, specific honeypot services are run as containers on a Debian 10 virtual machine, and are exposed to the Internet on the standard ports. Log files from each of the services are collected and forwarded via rsync to a central repository server, allowing for the periodic deletion of local files to save space. Containerized

images of the running services sandbox malicious activity from the host machine. Rayson-Cowrie [21] is a version of the Cowrie honeypot running on Docker and maintained by Rayson Zhu. The Cowrie container captures log files for SSH and Telnet activity in both .log and .json format, and simulates a real file system allowing attackers to execute commands, create and download files, and forward traffic, but confines these activities to the honeypot. The official Docker image of httpd, the Apache HTTP server project [22] is run in a container to host a fake website and allow exposure to the Internet without risking the host machine. Apache log files of interactions with the web services are captured and stored. The Docker image stillard-pure-ftpd [23] obtained from Github is used to host an FTP server with a few sample files is exposed and logs collected. In addition to logs from sandboxed services, Linux kernel logs from the host machine are collected to track changes in the host and to help determine if it has been compromised.

A central repository server collects data from each honeypot, and processes log files to generate statistics about malicious traffic and to make comparisons in honeypot activity. For this work we will focus primarily on log files generated by Cowrie.

IV. Analysis and Insights from Cowrie Data

Cowrie generates daily logs in both json and log formats. Json logs are built around events with each action initiated with the honeypot generating an event ID which defines a command executed by an attacker. Within an event are numerous data points, including source IP, destination IP, source port, destination port, session ID, username, password, messages (which contain commands issued by the attacker), timestamp, and many others. In this section we cover details of the collected data.

A. Source IP Addresses and Port Numbers

When contact is made with the honeypot, the source IP address and port number of the machine initiating the contact is stored. From the series of commands (covered in more detail later), it appears the most common types of activity are illicit login attempts and requests to forward traffic to another device. In the first scenario, the machine initiating the contact (possibly infected by a bot) is randomly scanning the network, then performing brute force attacks on susceptible hosts using a dictionary of usernames and passwords, with the owner of the offending device remaining unaware of this activity. In the second scenario, attackers are logging in and passing traffic through the honeypot to hide their location. Attacker machines are often masked by one or more proxy, VPN, or VPS devices, so the source IP recorded may not be the actual identity of the attack origin. Recorded in Fig. 1 are the unique source IP addresses globally that accessed or attempted to access one of the honeypots. 170865 unique IP addresses were recorded in total, with an average of 28478 unique IP addresses seen at each honeypot. These addresses were globally diverse, and appear to be random. There were also 6527 addresses which were present in all six honeypots. Random addresses, which make up the bulk of the observed source IP addresses collected, would be expected as bots scan the Internet seeking new victims. However, we see many addresses targeting the same machines which would seem to indicate credentials are being shared across a network of devices, which are in turn accessing them for the purpose of recruitment or traffic forwarding.

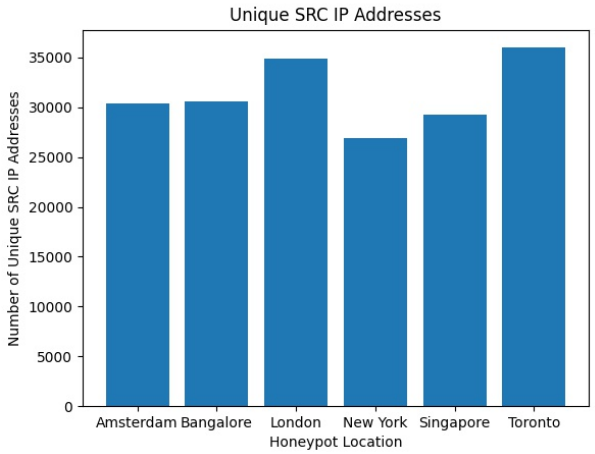

Fig. 1. Unique source IP addresses per honeypot location.

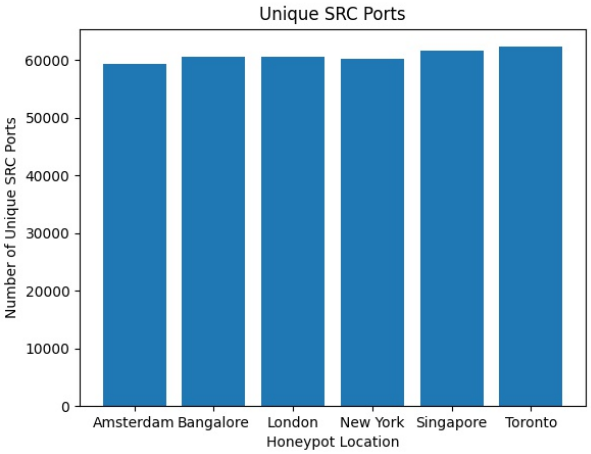

Fig. 2. Unique source port numbers per honeypot location.

While there are a variety of events occurring at each honeypot, a majority of the activity seen after a successful login are *session.connect*, *direct-tcp.forward* and *direct-tcp.data* requests. Many of the login attempts are being made for the purpose of utilizing compromised machines for routing traffic through SSH tunnels. By default, SSH sets the *AllowTcpForwarding* flag to *yes*, enabling others to use the victim machine as a SOCKS proxy to route any type of traffic generated by any protocol or program. To prevent this, the forwarding flag should be set to *no*. In addition, a limit on the number of login attempts allowed should be set to prevent brute force attacks. These forwarding requests are logged by Cowrie, but

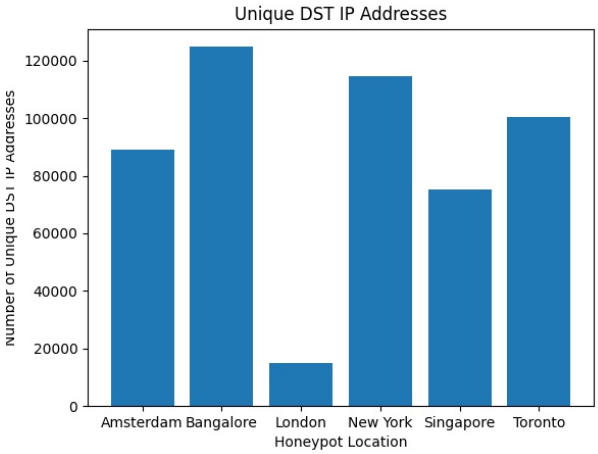

Fig. 3. Unique destination IP addresses per honeypot location.

are not actually forwarded. Requested destination IP addresses appear to be randomly distributed between those sent directly to targeted devices and with messages routed through a known proxy device to further mask an attacker's actual location. Source port numbers on the order of 60k were present in each honeypot as seen in Fig. 2 as bots were utilizing random port numbers to initiate contact.

B. Destination IP Addresses and Port Numbers

In order to better understand this forwarding behavior, an analysis is performed mapping attackers to their targets, and targets to attackers. Table II gives a numeric example of these mappings from the London honeypot. The left side of the table lists the top 20 attackers in terms of the number of targets each attacker can be mapped to, while the left side of the table lists the top 20 targets in terms of the number of attackers each target can be mapped to. It can be seen here there are far more targets per attacker than vise-versa, suggesting that this IP address has identified the honeypot as available for tunneling activity and is running through a list of targets using this host as a proxy. At the same time, we see these targets are being contacted by what would appear a coordinated network of attackers, some targets being accessed through all six honeypot locations. This suggests information about the honeypot and its compromised credentials are being shared across members of a botnet.

The honeypot in London saw far fewer unique destination IPs than the other honeypots (12746), while Bangalore saw the most (100252). Totals for all honeypots can be seen in Fig. 3 This is probably an indication of the types of attacks being carried out, possibly recruitment versus tunneling activity, although it is unclear why the London honeypot was being utilized differently over this time period. As for destination ports, there were 1038 port numbers targeted from Bangalore and 718 from Amsterdam, while the other honeypots ranged between 29-57, see Fig. 4. It is likely these two locations were engaged in scanning activity, while the others were focused on tunneling data to selected targets. London, Toronto, and New York were targeting mainly common services such as web, telnet, smtp, ssh, etc. Bangalore and Amsterdam seemed to target these, as well as many non-common port numbers associated with specific services (i.e. bo2k or Ghidra) that could have been identified by nmap scans.

We look specifically at tunneled traffic, examining the number of attackers and targets present in each honeypot, and then looking for their presence across all honeypots. Fig. 5 shows that there are no attackers that are found in every location (although most are found in more than one honeypot), while Fig. 6 shows there are 1045 targeted IP addresses that all six honeypots have in common. It would appear that while there are high value targets being sought by more than one attacking entity or botnet, no individual attacking IP is seen in every honeypot.

C. Daily Events

As mentioned previously, Cowrie .json logs are built around events, with multiple events often contained within a single session. A unique session ID is created when a connection is initially established, and the session is terminated when the connection is eventually closed. Fig. 7 shows the number of unique events per day across all honeypots, while Fig. 8 shows the total of all unique sessions per day across honeypots. Interestingly, there are several spikes in both events and sessions, indicating increased activity across different honeypots on the same days even though they are located in geographically separated regions. An attempt was made to correlate these peak days with other events such as news items or known attacks, but with no convincing results. There were several spikes in April 2020, likely due to the beginning of the Covid-19 pandemic and a surge in the number of people working on machines no longer protected by workplace security, as well as quarantined individuals being online at home more than usual.

D. Cowrie Sessions

To bring event IDs into perspective, Cowrie sessions can be used to aggregate a series of events under a single ses-

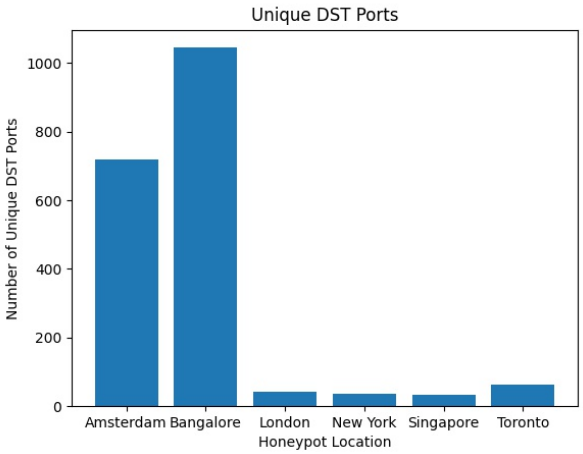

Fig. 4. Unique destination port numbers per honeypot location.

TABLE I
SERIES OF COMMANDS PER SESSION AND FREQUENCY OF OCCURRENCE.

	London	Amsterdam	Toronto	New York	Singapore	Bangalore
FADBEFADBE	40.3%	67.7%	67.8%	28.7%	53.4%	66.7%
FAGEFAGE	36.1%	21.5%	19.3%	36.2%	33.5%	15.6%
FAEFAE	10.5%	2.5%	0.41%	22.3%	5.5%	10.4%
FAGGGEFAGGGE	4.1%	0.1%	2.6%	5.3%	1.6%	3.2%
FEFE	4.5%	0.7%	2.8%	3.4%	3.0%	1.6%
FADBCEFADBCE	0.1%	0.5%	0.8%	0.1%	0.2%	0.3%

TABLE II
MAPPING OF ATTACKERS TO TARGETS AND TARGETS TO ATTACKERS.

Attackers to Targets	*Targets to Attackers*
5.182.39.88:114839	***google.com***:872
5.182.39.61:50318	***ya.ru***:871
5.182.39.62:25441	**208.95.112.1**:609
5.182.39.64:16741	***216.239.32.21***:492
5.188.62.11:13714	***216.239.36.21***:443
5.182.39.6:13049	***216.239.38.21***:374
45.227.255.163:7023	***216.239.34.21***:338
88.214.26.90:5120	***v4.ident.me***:270
5.182.39.185:3081	***video-weaver.arn03.hls.ttvnw.net***:138
45.227.255.205:762	***ipinfo.io***:118
5.182.39.96:443	***www.instagram.com***:113
5.188.86.172:233	***www.youtube.com***:102
88.214.26.93:166	***ip.bablosoft.com***:101
193.105.134.45:119	***video-weaver.waw01.hls.ttvnw.net***:101
103.114.104.68:60	***104.16.119.50***:101
51.158.111.157:53	***104.16.120.50***:101
45.155.205.87:44	***speedtest.tele2.net***:96
79.173.88.244:36	***87.250.250.242***:93
14.177.178.248:30	***m.youtube.com***:90
14.186.28.128:29	***check2.zennolab.com***:89

sion, giving a better view of command patterns and attacker behavior. By restructuring the Cowrie log as a dictionary with the session ID as a key rather than the event ID, all events containing the same session ID can be consolidated, and an order of events can be captured. There were 16 possible event IDs logged by Cowrie indicating actions such as a new connection, the success of a login attempt, a file download, a message being forwarded, etc. To make these aggregated lists of events easier to analyze, we assign a letter value *A-P* to each command, then build a string based on the sequence of commands executed during each distinct session. Table III lists the most common of these *A-G*, the others were omitted for space. The top six command sequences are listed in table I, along with the frequency of their occurrence compared to all identified sequences.

TABLE III
COMMAND LEGEND.

A	***cowrie.client.version***
B	***cowrie.direct-tcp.request***
C	***cowrie.direct-tcp.data***
D	***cowrie.login.success***
E	***cowrie.session.closed***
F	***cowrie.session.connect***
G	***cowrie.login.failed***

The most common sequence involves a connection (cowrie.session.connect), followed by the cowrie version supplied as part of the ssh handshake (*cowrie.client.version*), an indication of successful login (*cowrie.client.success*), then a request for a direct-tcp connection (*cowrie.direct-tcp.reques*) allowing the attacker to pass data through the honeypot to another destination. Finally, the connection is closed (cowrie.session.close). For the honeypot in London, of the

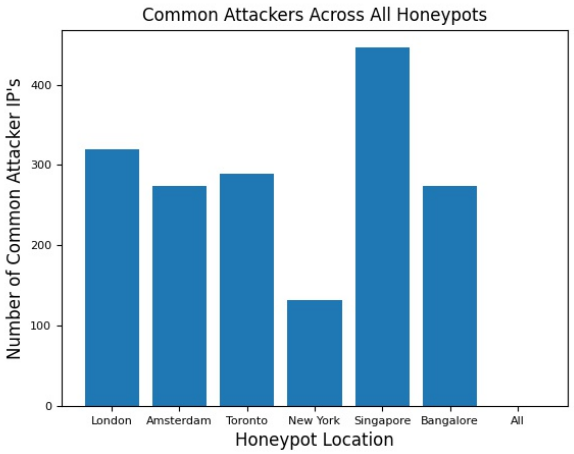

Fig. 5. Common attacker IPs across honeypots.

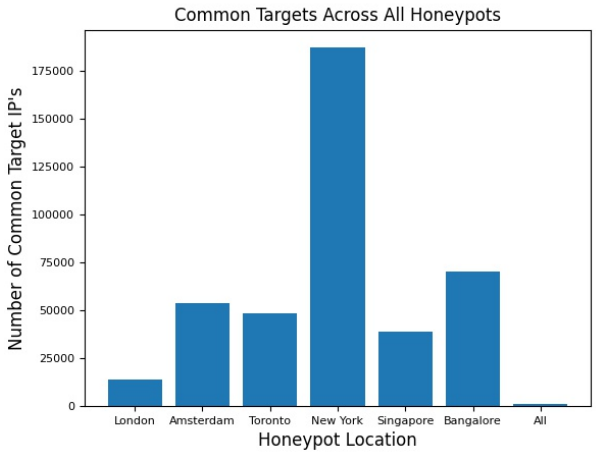

Fig. 6. Common target IPs across honeypots.

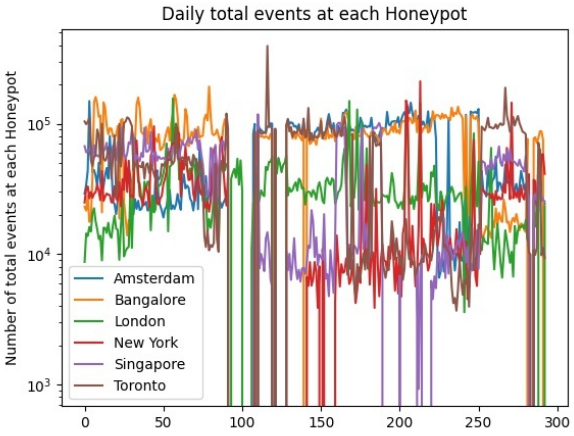

Fig. 7. Daily events per honeypot.

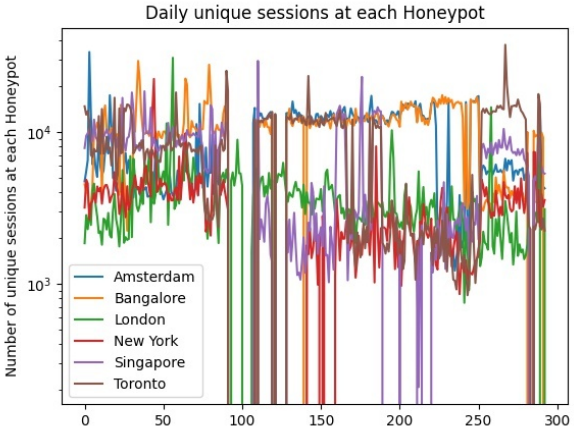

Fig. 8. Daily sessions per honeypot.

874782 sessions on April 12, 2020, about 40.3 percent, or 352537 of these followed this pattern. For the London honeypot there were 4459 unique command patterns captured in total. Most were of the common type indicated here, which were quite specific and relatively short. However, many individual unique command patterns found that were very long and exhibited repetitive patterns (some entailing hundreds of individual commands). We believe these likely contain repeated patterns that could be associated with an individual bot. This analysis will make an interesting future work as it could help to identify behaviors indicating malicious activity and aid in attacker identification.

E. Usenames and Passwords

Each login attempt, whether successful or not, captures the credentials that were given. A majority of these credentials attempt privileged access (either 'root', 'admin', or some other known default credential such as 'ubnt' for a Ubiquiti device), but many do not and instead utilize random usernames and a wide variety of passwords. As discussed in [2], this data is very useful as a Cyber Threat Intelligence (CTI) feed, allowing for compromised usernames and passwords to be black-listed from a system. Fig. 9 shows the number of unique usernames found in each honeypot over the duration of this study, while Fig. 10 shows all unique passwords used to gain access. Cowrie randomly accepts any login credentials after a variable number of attempts (set at three for our application), in an effort to "fool" attackers into believing they have correctly guessed login credentials. The most commonly used usernames and passwords are listed in table IV.

F. Malicious Downloads

Some Cowrie event contains a key *messages* that detail a command being executed in the honeypot. If a search is done for the string *wget* one can find attempts by an attacker to retrieve files from a remote host and download it to the honeypot. These are typically shell scripts that are then made executable using a *chmod* command either in the same message, or in a subsequent message, and are then run in the compromised machine. A list of these compromised files is gathered and could be used as a CTI feed to identify known malicious filenames. The IP addresses or URLs indicated in these download commands can be considered highly malicious, either as command and control devices, or more likely as file storage devices used for downloading malicious software. Again, these are often routed through a VPN or VPS. To help pinpoint likely sources for these downloads, a list of known VPN and data center addresses, and an api (getipintel.net) are used, and a list of possible actual download IP addresses is compiled. 1564 unique IP addresses are identified that do not appear in the known VPN or data center list. As a future work, it would be interesting to retrieve these shell scripts and analyze them forensically. The top 20 malicious file names, all of which appear in each of the six honeypots is given in table IV.

V. Conclusion

To better understand attack patterns and behavior, honeypots can be deployed to monitor and log activity by simulating actual Internet facing services. By examining traffic patterns, downloads, and traffic forwarding across a series of geographically separate devices we attempt to better understand

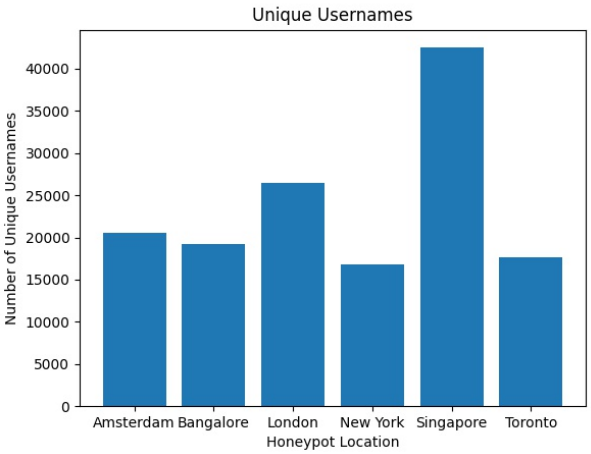

Fig. 9. Unique usernames per honeypot.

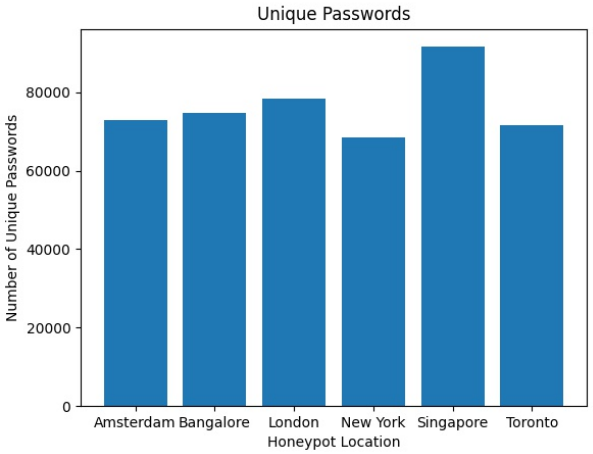

Fig. 10. Unique passwords per honeypot.

TABLE IV
MOST USED USERNAMES, PASSWORDS AND DOWNLOAD FILENAMES.

Usernames	*Passwords*	*Malicious Filenames*
root	*guest*	*bins.sh*
guest	*admin*	*GhOul.sh*
admin	*root*	*yoyobins.sh*
test	*test*	*SnOopy.sh*
user	*123456*	*axisbins.sh*
ubnt	*1234*	*EkSgbins.sh*
0101	*user*	*8UsA.sh*
nproc	*password*	*Pemex.sh*
22	*ubnt*	*Sakura.sh*
support	*0101*	*sh*
oracle	*123*	*skid.sh*
postgres	*nproc*	*UwUsh*
ubuntu		*zeros6x.sh*
usario	*support*	*mavscock.sh*
	matrix	*KigaNet.x86*
git	*12345*	*infn.x86*
Administrator	*usario*	*ISIS.sh*
pi	*123456789*	*installer.sh*
1234	*12345678*	*Gummy.sh*
ftpuser	*1*	*gtop.sh*

malicious activity and work to identify patterns that can be useful in identifying malicious traffic in actual servers or IoT devices. In addition, collected data such as source IP addresses, download filenames, and login credentials can be useful as a threat intelligence feed to protect digital assets. As a future work more analysis on command series patterns could be conducted to possibly predict attack behavior and to identify threats in real time on production servers and IoT assets.

ACKNOWLEDGMENT

This research is supported by NSF Award #1739032.

REFERENCES

[1] Lingenfelter, Bryson, Iman Vakilinia, and Shamik Sengupta. "Analyzing variation among IoT botnets using medium interaction honeypots." 2020 10th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2020.

[2] Vakilinia, Iman, Sui Cheung, and Shamik Sengupta. "Sharing susceptible passwords as cyber threat intelligence feed." MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM). IEEE, 2018.

[3] Memari, Nogol, Shaiful Jahari Hashim, and Khairulmizam Samsudin. "Network probe patterns against a honeynet in Malaysia". Vol. 8. Kajang: Science and Technology Research Institute for Defence, 2015.

[4] Fan, Wenjun, Zhihui Du, David Fernández, and Victor A. Villagra. "Enabling an anatomic view to investigate honeypot systems: A survey." IEEE Systems Journal 12, no. 4 (2017): 3906-3919.

[5] Luo, Y., Zhang, Z.,Esaki, H., Ochiai, H. "Classification of TCP 445 attacks and global snapshot with honeypot analysis", IEEE, 2019, doi:10.1109/AITC.2019.8921162.

[6] Kumar, Sanjeev, B. Janet, and R. Eswari. "Multi platform honeypot for generation of cyber threat intelligence." 2019 IEEE 9th International Conference on Advanced Computing (IACC). IEEE, 2019.

[7] Nawrocki, M., Wählisch, M., Schmidt, T.C., Keil, C. and Schönfelder, J., 2016. "A survey on honeypot software and data analysis". arXiv preprint arXiv:1608.06249.

[8] Kyriakou, Andronikos, and Nicolas Sklavos. "Container-based honeypot deployment for the analysis of malicious activity." 2018 Global Information Infrastructure and Networking Symposium (GIIS). IEEE, 2018.

[9] Cabral, Warren, Craig Valli, Leslie Sikos, and Samuel Wakeling. "Review and analysis of Cowrie artefacts and their potential to be used deceptively." 2019 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 166-171. IEEE, 2019.

[10] McCaughey, Ryan J. "Deception using an SSH honeypot". Naval Postgraduate School Monterey United States, 2017.

[11] Fraunholz, Daniel, Daniel Krohmer, Simon Duque Anton, and Hans Dieter Schotten. "Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot." 2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security), pp. 1-7. IEEE, 2017.

[12] Zhang, Zhiqing, Hiroshi Esaki, and Hideya Ochiai. "Unveiling malicious activities in LAN with honeypot." 2019 4th International Conference on Information Technology (InCIT). IEEE, 2019.

[13] Koniaris, Ioannis, Georgios Papadimitriou, and Petros Nicopolitidis. "Analysis and visualization of SSH attacks using honeypots." Eurocon 2013. IEEE, 2013.

[14] Kambourakis, Georgios, Constantinos Kolias, and Angelos Stavrou. "The mirai botnet and the iot zombie armies." MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM). IEEE, 2017.

[15] Fraunholz, Daniel, Daniel Krohmer, Simon Duque Anton, and Hans Dieter Schotten. "YAAS-On the attribution of honeypot data." IJCSA 2, no. 1 (2017): 31-48.

[16] Pittman, Jason M., Kyle Hoffpauir, and Nathan Markle. "Primer–a tool for testing honeypot measures of effectiveness." arXiv preprint arXiv:2011.00582. 2020.

[17] Vetterl, Alexander, Richard Clayton, and Ian Walden. "Counting outdated honeypots: legal and useful." 2019 IEEE Security and Privacy Workshops (SPW). IEEE, 2019.

[18] Vetterl, Alexander, and Richard Clayton. "Bitter harvest: systematically fingerprinting low-and medium-interaction honeypots at internet scale." 12th USENIX Workshop on Offensive Technologies (WOOT 18). 2018.

[19] Bontchev, Vesselin, Veneta-Yosifova, "Analysis of the global attack landscape using data from a telnet honeypot." Information and Security: An International Journal 43 (2019): 264-282.

[20] Bistarelli, Stefano, Emanuele Bosimini, and Francesco Santini. "A report on the security of home connections with IoT and Docker honeypots." ITASEC. 2020.

[21] "Rayson Cowrie." [Online] Docker Hub, 26 Nov. 2020, Available: https://hub.docker.com/r/rayson/cowrie/.

[22] "HTTPD, the Apache http server project." [Online] Docker Hub, 26 Nov. 2020, Available: https://hub.docker.com/_/httpd

[23] "Stillard Pure FTPD." [Online] Github, 26 Nov. 2020, Available: https://github.com/stilliard/docker-pure-ftpd

[24] "Linux Took Over the Web. Now, It's Taking Over the World" [Online] Wired Magazine, Aug. 2016, Available: https://www.wired.com/2016/08/linux-took-web-now-taking-world

[25] "Eclipse 2018 survey: The IoT landscape, what it empirically looks like" [Online] Canonical Ubuntu Blog, Apr. 2018, Available: https://ubuntu.com/blog/eclipse-2018-survey-the-iot-landscape-what-it-empirically-looks-like

[26] "VPNs and Data Center IPs" [Online] Github, 14 Dec. 2020, Available: https://github.com/ejrv/VPNs