The Effects of Network Size

on Attacker Behavior

HACS200

Group 2A Humamana

Sumit Nawathe, Amrit Magesh, Anthony Huynh, Anthony Ma

0. Executive Summary

Advanced persistent threats are becoming increasingly common and dangerous forms of complex cyber attacks on organizations and governments (Rege). These attacks often feature malicious actors navigating through and establishing a presence in a computer network ("Lateral Movement"). Our goal is to analyze what properties of a network affect attacker behavior related to lateral movement by answering the question: how does the size of a network affect the scale/intensity of attacks it faces? Our null hypothesis is that network size has no effect on attacker behavior, and we have 3 alternative hypotheses that argue that network size can affect different measurements of attack intensity: time spent in the network, number of containers in the network visited, and the number of commands entered. To test these hypotheses, we designed an experiment with honeypot networks with different numbers of connected containers that can be reached by SSH. We collected data on attacker authentication/logins, commands entered, and containers accessed, primarily utilizing the ACES MITM server and container auth.log files. For each network, we also implemented an SSH banner stating the network size to guarantee experimental intervention for all data points. Unfortunately, we did not find any instances of attackers moving laterally through our honeypots. Regarding our other metrics, we found that honeypot network size did not affect attacker time in the honeypots, but did affect the number/frequency of commands entered. Each of these analyses faced complications, including different methods of splitting the data as well as inconsistent pairwise tests. Thus, while we conclude that knowledge of network size can be seen as affecting attacker behavior depending on perspective, more research is needed to identify the direction/scale of trends as well as generalize the experiment to identify other significant network features and subsequent changing components of attacker behavior.

1. Background Research

Similar work has studied the impact of honeypot network architecture on attacker decisions and behavior. The research most similar to our experiment is an analysis by Katakwar et al., which examines the impact of network size on attacker decisions. Their paper treats network attacks as a theoretical decision game with various parameters including size; their experiment consists of having participants play a game in which they are shown a network of web servers, half of which are honeypots, through a simplified web interface and are able to select whether to attack specific machines. The authors found that the proportions of honeypots probed among the network increased with network size, and the proportion of no-attack actions was larger for smaller networks. The primary limitation of this study is that the laboratory experiment is not fully indicative of real-world behavior; while the characteristics of search/attack and hidden attacker knowledge were preserved in the experiment, use of arbitrary humans and a simplified interface/procedure limits what conclusions can be drawn.

In contrast, our research was conducted with real attackers exploring our honeypot network through SSH connections. As seen in Section 3, our experiment relied on attackers using SSH vulnerable keys to move laterally between honeypots in the network; this is a known and common method of lateral movement, as described in the paper by Cao et al. This paper presents an automated system for identifying SSH brute-force attacks; the authors found that many attacks attempted to use SSH keys, likely obtained through "direct compromise of file systems, either by using ransomware or through reverse-engineering." They further recommended that hashing the "known_host" file and using passphrases to protect private SSH keys would reduce attackers' lateral movement, indicating that this is a popular and often vulnerable way to perform lateral movement, which we take advantage of in Section 3.

In our honeypot architecture, we have restricted ourselves to only SSH, deciding against involving honeypots built on web or database interactions in our network. In the paper by Bar et al., the authors trained a Markov model on a dataset collected through attacks on a variety of honeypots of different types, vulnerabilities, and services. The authors found that attack propagation -- sequential attacks between honeypots -- mostly occurs between honeypots of the same type, forming communities of attacks. Thus, only using SSH honeypots in our research, while possibly introducing bias, may lead to a larger dataset of attacker movement between honeypots.

Our alternative hypotheses argue that increasing network size will affect the number and intensity of attacks, and intuitively, we expect an increasing relationship with network size. This notion is supported at the largest scales by the paper by Bagchi and Tang, which attempts to find a mathematical model for attack frequency relative to the size of the Internet from its earliest days, treating the Internet as a network. The authors found that "the number of incident attacks can be better predicted by considering an exponential network size effect," and believe this result to be generalizable to single organizations with large networks. Our experiment acted on a much smaller scale, but hoped to find a similar trend.

2. Research Question and Hypothesis

Our overarching research question is: how does network size affect the scale/intensity of attacks it faces? We define a network as a group of machines that can be accessed by one another through a series of SSH connections and at least one of which can be accessed from outside the network (externally). An attack is defined as a malicious actor successfully connecting to (logging into) at least one machine in the network.

Our independent variable consists of one factor: size refers to the number of machines in the network. We define our dependent variable, scale/intensity of attack, through three separate metrics: (1) Time is the number of seconds connected to the network. (2) Lateral movement is the total number of machines in the network visited. (3) Activity is the number of commands (by line) entered during their time in the system.

We expect that attackers will react to information about their target network, and that network size may affect their attack's scale and goals. As such, our null hypothesis (H0) is that the size of a network has no effect on the scale/intensity of attacks.

We present three alternative hypotheses related to our three dependent variable metrics. HA1: Network size affects the amount of time that attackers spend in the network system. HA2: Network size affects the extent of lateral movement of attackers. HA3: Network size affects the level of activity (number of commands entered) of attackers.

3. Experiment Design

To collect data on how network size affects these metrics of attacker behavior, we designed an experiment utilizing honeypot networks of varying sizes. For every honeypot network that we created, we began by picking a network size n, randomly selected from among 3, 6, 9, and 12 network containers. Our honeypot consisted of $n + 1$ containers: n honey-filled containers, and one container acting as a "router", the only externally-accessible entrypoint for the attacker into our network. Each container had an OpenSSH server running, and attackers could only connect to the router and network containers through SSH. The router container was configured to contain the public keys of all network containers in its /root/.ssh/known_hosts file, as well as an SSH alias to each network container as root (figure B.E2). Thus, by accessing these files on the router, an attacker can determine the size of the network and easily traverse laterally

(pivot) from the router to other machines in the honeypot network. In addition, on the router container, we added an SSH banner that contains the size of the honeypot network, which guarantees experimental intervention once an attacker successfully enters the honeypot, even if the attacker did not traverse laterally or observe the SSH files (*script C1*).

For ease of data collection, each honeypot network had a corresponding instance of the ACES MITM server. This server facilitates connections to the router container while providing extensive logging on authentication and attacker behavior/commands. (A diagram of the network structure with MITM is in *figure E1*.) The server was configured to allow auto-access to an attacker after two login attempts. Additionally, we also tailed each individual containers' auth.log file, which tracks SSH authentication attempts and timestamps. These two data sources were logged to files alongside the honeypot network initialization records for the duration of the honeypot's lifetime.

Each of the network containers (excluding the router) was populated with honey, randomly selected from either PII, financial data, or medical data. Our honey was generated using multiple fake data generation libraries (Synthea; Faker; FakeStockData). For each network container instantiation, a pre-generated honey file was randomly selected, with equal probabilities among types of honey, and loaded into the container. Due to the great number of containers and combined size of honey loaded simultaneously across all honeypots, we mounted a 100GB drive for use by LXC to provide sufficient disk space for all containers, as well as dramatically increased filesystem inotify resources.

We made extensive use of firewall rules and network address translation to restrict possible connections between containers. The external IP address is mapped to the ACES MITM server, which then forwards connections to the router container. Each network container was

only accessible from the router container, while all internet traffic to all containers in the network was directed through the external IP address. The honeypot initially is open to all potential attackers. When an attacker first successfully entered the honeypot router (after MITM auto-access), we add a firewall rule to restrict access to that honeypot to only that attacker's IP address for the remainder of the honeypot's lifetime, in order to ensure no interference between attacker data points and to make data processing feasible. After this, we provided the attacker with root privileges for all containers and copied the SSH known_hosts and alias files to their home directory. Then, we start a 30 minute timer, after which the attacker is removed and the honeypot is recycled (a new network with randomly chosen size is created for the same external IP address) (*script C1*). The rationale for this recycle time is that we anticipated attackers would use mostly non-interactive mode, so this ceiling would likely not affect attackers' time in the honeypot. Moreover, we anticipated most attackers not traversing laterally through the network, so we wanted as many unique attacks as possible to have a higher chance of observing pivoting.

On recycle, we ran simple preprocessing scripts on the collected ACES MITM log and containers' auth.log files to extract command lines that the attacker entered as well as how long attackers spent in each container. The raw log files and preprocessed output were all packaged into a .tar.gz file labeled by an epoch timestamp; the compressed archive is then stored in a folder containing all honeypot data points for honeypots of the same network size (4 folders total) (*script C4*). The recycle process then continues to instantiate a new honeypot with a randomly chosen size for the same external IP address, following all of the procedures previously outlined (*script C1*). At all times, we had 5 honeypot networks with separate external IP addresses running simultaneously, each of which had a separate recycle process and independent randomization. Automated incremental backups of our total dataset were taken daily

to a separate virtual machine using a crontab script, which provided a separate, secured workspace for data processing.

4. Experiment Design Changes

In our original research question, we aimed to analyze how the size and shape of a network affects the scale/intensity of an attack. For the independent variables, we defined size as the number of containers and shape as the structural layout of the network nodes, which determine paths attackers can laterally move through. For the dependent variables, we defined scale/intensity through time (the number of seconds an attacker is connected to a network), lateral traversal (number of containers connected to within the network), activity (number of commands entered by line), and penetration (level of file system access).

Our first major design change was removing shape as an independent variable. Before implementation, we realized that creating different network topologies, such as trees or cyclical graphs, along with managing randomized SSH connections would become a very difficult process to complete before honeypot deployment. Moreover, incorporating both size and shape requires us to examine the joint, marginal, and conditional distributions of our dependent variables, which would require very complex data analysis. Our second design change was to drop file system access as a dependent variable. We determined that automating generation and detection of levels of file access (hidden files, permissions, encrypted files) would be extremely tedious to implement and was not as closely related to our question's theme of attacker lateral movement as the other metrics.

As our experiment progressed, we made a few design modifications to improve our data collection. In our original vision for the experiment, we planned to limit our data points to attackers who had traversed laterally to at least one network container, which would guarantee

that they were aware of the network's size. However, after our honeypot deployment, we quickly found that almost no attackers had pivoted to an internal container. Thus, on October 20th, we added the SSH banner that included network size on login to guarantee experimental intervention at the point of attacker connection. Subsequently, we removed all data collected prior to this modification. Additionally, we realized that in our original implementation the attacker must visit the root directory after login to access SSH aliases and known_hosts files. To simplify this process, we altered our honeypot setup to copy the .ssh folder to the attacker's home directory in the router container upon login to make it easier for an attacker to identify other containers in the network and potentially move laterally. This was implemented on October 21st, and once again all prior data points were removed. Finally, we discovered an implementation issue that allowed multiple attackers to enter the honeypots simultaneously, which was contrary to our experiment paradigm and made data analysis impossible. We resolved this on November 1st, and dropped all prior data.

5. Data Collection

Our data collection primarily focused on attack features that directly related to our dependent variables: authentication attempts, attacker login/exit timestamps, and commands entered while having access to the honeypot in SSH interactive or noninteractive mode. To collect these, we primarily utilized the ACES MITM Server logs, as it contains all of this information at a granular level. With timestamps in milliseconds for each log entry, we had data on attacker access attempts, IP addresses, and session streams, which could be easily processed to extract key features. In addition, we also utilize each container's auth.log file (by continuously tailing it from the honeypot host machine), which tracks all SSH login attempts. This allows us to definitively answer whether an attacker attempted to ssh to a network machine from the router.

In total, we had about 4000 attacked honeypots (records from around 4000 honeypot recycles), evenly distributed across honeypot sizes. (The distribution of recycles and sessions are shown in *figures B.A1-2*.) The majority of the data during our experiment's lifetime is usable; the data discarded was (1) sessions before the SSH banner was installed as we were not guaranteed experimental intervention, and (2) sessions before we fixed our firewall rules to only allow one attacker at a time, as previous sessions would not follow our original experiment's paradigm. All sessions since November 1st are used in our analysis.

To adequately handle this scale of data, we developed a multi-stage data processing pipeline. When each honeypot was recycled (end of its lifetime), we had access to its MITM log file and each of its container's auth.log files (tailed). We ran some preliminary preprocessing on the raw files to measure (1) the amount of time the attacker spent connected to the honeypot and each container (*script C3*), and (2) all of the commands entered in SSH interactive or non -interactive mode. The raw and simple preprocessed files are then zipped into a timestamp -labeled file, which is placed into a folder for honeypots of the same size network for analysis.

For second-stage data-processing, we had two different pipelines. Due to volume, we believed it would be prudent to calculate basic metrics as our packaged data arrived, rather than attempt to perform complex calculation all-at-once, which would have overwhelmed our system. Therefore, we created a system that automatically tracked which new files were added when taking our regular backups and calculated the following metrics for each recycle session: sum of time across all containers, sum of all commands (including separation by semicolons), unique commands, time spent in the router, time spent in network machines, and number of interactive commands. This gave us access to easily digestible metrics in a csv file which could be formatted/outputted to run through a Python statistics pipeline. However, as we discovered edge

cases and thought of more metrics to add, we realized that it was difficult to modify this continuously-running pipeline. Thus, we created a second parallel system in Ruby which was dynamically configurable to calculate a single metric on every datapoint (packaged honeypot session) all-at-once (*script C2*). While this was resource-intensive and was unable to access any of our previously-calculated metrics due to storage complications, it did allow us to easily develop processing code and run one-off calculations on our entire dataset to help us find interesting features for further analysis.

The output of both systems of preprocessing was formatted in a way that was human-readable (for spot checking data points) and could be easily parsed in a Python statistical pipeline. We primarily used Jupyter notebooks with the pandas and scipy packages to aggregate the data, and Minitab to perform hypothesis testing. Some simple late-stage processing was also performed in Python on the calculated metrics due to ease-of-use.

When calculating attacker time in the honeypots, we initially planned to use the container auth.log files to find when each attacker enters and leaves. However, these log files only record times to the second, and as the majority of sessions were fast non-interactive mode commands, no time would register. Thus, the container auth.log files could detect the presence of an attacker (suitable for the lateral traversal dependent variable), but not measure time passing. In contrast, the ACES MITM records attacker access and exit timestamp in milliseconds, but only to the network as a whole (specifically the router container). The MITM log can be used to find how long the attacker was connected to the honeypot, but not the time spent in any of the individual network containers. We flagged those data points using the container auth.log and manually computed the times by cross-checking when lateral traversal commands were entered with the specific container's log in those few cases (of which we found none, so this was not necessary.)

The attacker's time for each session was the difference between the time the attacker was marked as entering and leaving in the MITM logs; the attacker's time connected to that honeypot during the honeypot's lifetime is the sum of all the session times during that lifetime.

Attacker commands are easily found in the MITM logs, and can be extracted by looking for the correct tags. Our original research question focused on the number of lines entered, but as most sessions are non-interactive, this would be highly correlated with the number of attack sessions during a honeypot's lifetime. Additionally, when doing case analysis, we found great variance in how much attackers accomplished in even a single noninteractive line. Thus, in addition to analyzing just the number of lines, we also broke apart lines into separate commands (separated by semicolons) and counted each component separately.

6. Data Analysis

Unfortunately, in our dataset of attackers we had no instances of lateral movement from the router to the other network containers. No attackers interacted with the ~/.ssh directory contents (known_hosts, aliases), and all of the sessions were in SSH noninteractive mode. While we are able to analyze the effects of our experimental intervention, any analysis will lack some groundedness relative to our meta-question of what affects attackers' lateral movement.

For our first alternative hypothesis, we began by analyzing the total amount of time the attacker spent in the honeypot during each honeypot recycle, cumulative over all sessions that the attacker had during the 30 minute honeypot recycle window. This quantity attempts to measure intensity over a given time period: attackers that spend more time in the honeypot during a set window could be viewed as engaging more with the honeypot and thus performing a more intense attack. Our data is shown in *figure B.B1*; clearly, it is extremely skewed, with most attackers spending close to 0 seconds, but several large outliers spending thousands of seconds

per recycle (up to a max of 1800 seconds). Due to this heavy skew, we chose to run a Kruskal-Wallis test to examine whether there is a difference between the total-attack-time distributions for the different-size honeypots. From the test in *figure B.B2*, we obtain a p-value of 0.11, so we are unable to reject the null hypothesis that honeypot size affects total attack time (over the 30 minute interval) at the 10% significance level. As an additional analysis, we also examined the distributions of attack times by session for the different-size honeypot, where we counted each individual session as a distinct datapoint rather than summing them over recycle periods. We do this because measuring by recycle introduces a time cap component (measuring attacker time as a proportion of total time given), while session durations are absolute. This dataset is presented in *figure B.B3*; as the distribution is very concentrated near 0 seconds, we also present the same data with a log scale in *figure B.B4*. We perform the same Kruskal-Wallis test on this session duration dataset, as shown in *figure B.B5*, and obtain a p-value less than 10^-3, which allows us to reject the null hypothesis that there is no difference between the distributions of session durations for different-size honeypots at the 10% significance level. As a follow-up analysis, we perform a Mann-Whitney U test for each pair of honeypot sizes to examine whether each pair has the same distribution, as presented in *figures B.B6-11*. We obtain significant p-values (at the 10% level) for tests between sizes 3/6, 3/9, 6/12, and 9/12; for each of these pairs, we can reject the null hypothesis that the two sizes have the same session-time distribution (at the 10% level). This suggests that sizes 3/12 and 6/9 have similar distributions, which is counter-intuitive as we would expect honeypots of similar sizes to have similar distributions. Furthermore, from the medians found in performing the tests, there is no monotone relationship between size and attack duration. Thus, we cannot make a strong statement regarding a trend, only a weak statement that there is a difference in distributions.

To test our third alternative hypothesis, we first calculated the number of commands (measured by line) entered by the attacker during each recycle period, cumulative over all sessions during that recycle. Similar to the time analysis, this attempts to measure the intensity of attacks, as attackers who enter more commands during the 30-minute recycle window can be seen as engaging more with the honeypot (more commands per minute). Our data is shown in *figure B.C1*; once again, it is very skewed, with most attackers entering a few command lines per recycle period, while several outliers enter dozens of lines. We perform a Kruskal-Wallis test to examine whether there is a difference in the distributions of the number of lines of commands for the different-size honeypots, as shown in *figure B.C2*, and obtain a p-value of 0.048. Thus, we can reject the null hypothesis that there is no relationship between network size and lines of commands entered during the honeypot's lifetime at the 10% significance level. To examine this further, we perform a Mann-Whitney U test on each pair of honeypot sizes, as shown in *figures B.C3-8*, and obtain significant p-values (at the 10% level) for size pairs 3/9, 6/12, and 9/12 (for each pair, we can reject the null hypothesis that honeypots of those sizes had the same distribution of number of commands per recycle). This does not yield a consistent grouping of similar distributions, so we are unable to draw strong conclusions. Further, from the distribution medians, we do not find any monotone relationship between network size and number of commands. During our data analysis, we noticed that some attackers used single noninteractive command lines to perform multiple commands at once by use of semicolons in Bash. Thus, as an additional analysis, we counted the number of commands entered during each recycle session, splitting lines on semicolons (for example, a single line "ls; pwd" would count as 2 commands). This data is presented in *figure B.C9*, and the corresponding Kruskal-Wallis test in *figure B.C10* produces a p-value of 0.085, which allows us to reject the null hypothesis that network size has

no effect on number of commands entered, when measured by counting lines split on semicolons, at the 10% significance level. Once again, we perform pairwise Mann-Whitney U tests, as shown in *figures B.C11-16*, and obtain significant p-values (at the 10% level) for honeypot size pairs 6/12 and 9/12 (allowing us to reject the null hypothesis of no difference between these pairs of distributions). Once again, this does not yield a consistent grouping of similar distributions, making interpretation limited.

In our exploratory data analysis, we attempted to examine the number of commands entered (by line and when split on semicolons) for each session rather than for over all recycle periods, but this did not prove fruitful. For measuring commands by-line, as almost all of our data is from SSH non-interactive mode, almost all of the sessions would register as consisting of one command line. When splitting lines on semicolons, while there was more variation, once again almost all of the sessions (vast majority non-interactive) consisted of a single command. The frequency distribution for the number of sessions having different numbers of commands (split on semicolons) for each size honeypot is given in *figure B.C17* for reference, but little further analysis could be performed with interpretable results.

As another additional analysis, we looked further at the content of command lines entered for each size honeypot. For each size of honeypot, we looked at every distinct command line entered and counted the number of times it appeared over all corresponding honeypot recycles/sessions. We found that there were several very popular commands, and several commands that were run either very few times or were esoteric (not relatable to any other popular command). We present all command lines for which we found at least 5 instances for each honeypot size in *figure B.D1*. To test the null hypothesis that the relative frequencies of these “popular” command lines are the same for each size honeypot, we perform a chi-square test

for association between these frequency distributions, shown in the same figure. We obtain a p-value of 0.004, which allows us to reject this null hypothesis and conclude that there is a distinction in what popular commands are run in each size honeypot (at the 10% level). As a follow-up, we run chi-square tests of association for each pair of honeypot sizes, as shown in *figures B.D2-7*, and find significant p-values (at the 10% level) for size pairs 3/6, 3/9 and 9/12. Once again, this does not yield a consistent grouping of similar distributions, so we cannot draw significant conclusions. From the results of the overall chi-square test, there does not seem to be a monotone trend between relative frequencies of any pair of commands over all honeypot sizes.

Finally, as we have no instances of attackers moving laterally to other honeypot network containers, we are unable to respond to our second alternative hypothesis relating to differences in tendency to move laterally for different size networks.

7. Conclusion

From our data analysis, we obtained varying results with limited interpretability regarding our hypotheses. We were unable to reject the null hypothesis relative to our first alternative hypothesis that the time attackers spend in the honeypot during its lifetime is affected by network size. We would have been able to reject the null hypothesis if we had instead measured time by session and not by recycle period, but the resulting groups of similar distributions do not reveal a consistent trend between network size and session duration (in particular, no monotone trend). These results suggest that attackers may respond to knowing network size in a way that alters session duration, but that this is not sufficiently significant or frequent enough to affect the overall proportion of time spent in the network. Thus, we conclude that knowledge of network size does not significantly appeal to or dissuade attackers to spend time in a machine/network at the medium-time scale (not short/session-time scale).

Regarding our analysis of attacker commands, we were able to reject the null hypothesis in favor of our stated alternative that network size does impact the number of commands entered (but not regarding any monotone trend). We obtained similar results when using different methods of counting the number of commands. However, in no case were we able to obtain consistent groupings of honeypot sizes with similar distributions. Thus, we are unable to draw strong trend-oriented conclusions; we can only say that knowing network size likely influences attack intensity by affecting how many commands attackers enter during their time connected to the system. From our alternative analysis, we also found that relative frequency distributions of types of commands differ for different size honeypots. Combined, the differences in content and quantity of commands may suggest that attackers' destructive/informative actions using commands may differ for honeypots of different sizes.

We cannot make any statements regarding attacker lateral movement as we did not find any instances of attackers moving laterally. It seems that attackers are not interested in moving to machines beyond the entrypoint, but the previous analyses suggest that merely knowing network properties may have an impact on attacker behavior. To gain a better understanding of the trends induced in attacker behavior by network size, our future work may expand our methodology to test network sizes at larger orders of magnitude (dozens or hundreds), which may elicit more tangible differences. Our future work may also examine the effect of network topology or other factors, such as rate limiting, on attacker behavior. Additionally, the features of attack time and number of commands that we relied on partially depend on exclusively using SSH honeypots; thus, future work may involve similar experiments measuring how network features impact attacker behavior when different services are involved, such as web services or databases.

Appendix A

Over the course of this HACS project, we learned numerous lessons. Perhaps the very first thing we realized was to be more generous in our timeline. We were overly ambitious in the design period of the project since we had to cut down on our initial research question, which was too complex to implement in the scope of this project's development period and analyze data for. Further, we learned that we should plan to have more buffer days for roadblocks, especially as we were nearing deployment and there were minor issues we were continuously debugging in our recycle and data collection scripts as well as issues we were not aware of (file system resource limits, disk space). As a team, we learned the importance of meeting regularly. Having a consistent form of communication in our group chat allowed us to schedule meetings frequently, which let us all be on the same page and document important points discussed. Lastly, we learned how critical it is to be flexible and open-minded. We ran into numerous roadblocks with file system resources, disk space, and scripts during development, and we learned to respond quickly and adapt to the issues. Moreover, we understand that attacker behavior can often be undesirable, so suggestions from instructors on potential alternatives for our data collection pipeline helped us produce the most optimal statistical analyses.

Speaking of instructors, we were very appreciative of the constant support and communication from the TAs and professors, who always came around to ask for updates and were immediate to respond to questions/comments on Slack. However, we would have appreciated more direction at the beginning of the project as we had no idea of the scale of this project and thus had to bring it down a lot. There was also some technical knowledge outside of what was taught in HACS101 that was vital for this project (unexpected roadblocks with the MITM server, utilizing github, process control) that some of us did not prepare for or have as

much experience in, which facilitated a lot of learning on the side during implementation. One area that we appreciate for having a lecture was statistics; however, we would have appreciated a better understanding of the specific hypothesis tests given to groups.

We were not surprised by the attacks as both TAs and ACES alumni had pointed out that the data was mostly limited to bots, with very few interactive sessions. We were very surprised the first weekend after we had deployed since we had an extensive amount of attacker connections. We learned most connections would be non-interactively run, which modified our data collection. An analysis of the command distribution supports that many sessions were similar to one another, with the same groups of attackers repeatedly running the same commands. At the beginning, we were very skeptical that any data would yield an attacker traversal, so we were surprised to find out that there were a few attackers who had pivoted in some of the data we had to unfortunately discard.

Appendix B

Figure A1:

RECYCLE_AND_SESSION_COUNTS.CSV

Chi-Square Goodness-of-Fit Test for Observed Counts in Variable: Number of Recycles

Using category names in Honeypot Size

Observed and Expected Counts

Category	Observed	Test Proportion	Expected	Contribution to Chi-Square
3	1134	0.25	1109.25	0.55223
6	1135	0.25	1109.25	0.59776
9	1100	0.25	1109.25	0.07714
12	1068	0.25	1109.25	1.53398

Chi-Square Test

N	DF	Chi-Sq	P-Value
4437	3	2.76110	0.430

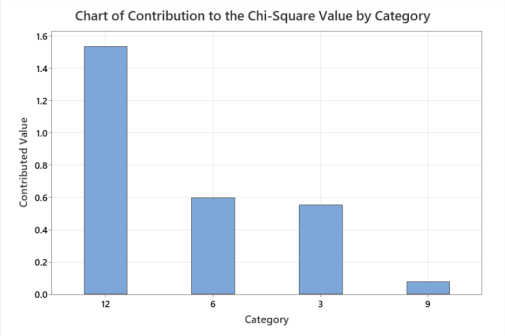

Figure A2:

RECYCLE_AND_SESSION_COUNTS.CSV

Chi-Square Goodness-of-Fit Test for Observed Counts in Variable: Number of Sessions

Using category names in Honeypot Size

Observed and Expected Counts

Category	Observed	Test Proportion	Expected	Contribution to Chi-Square
3	6022	0.25	6006.5	0.039998
6	6066	0.25	6006.5	0.589403
9	5947	0.25	6006.5	0.589403
12	5991	0.25	6006.5	0.039998

Chi-Square Test

N	DF	Chi-Sq	P-Value
24026	3	1.25880	0.739

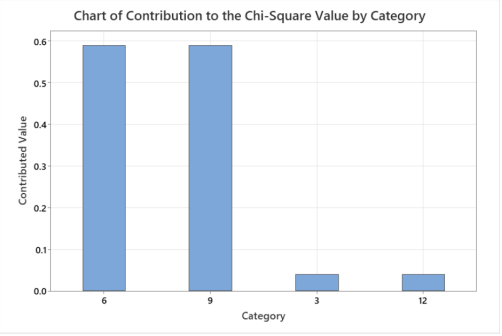

Figure B1:

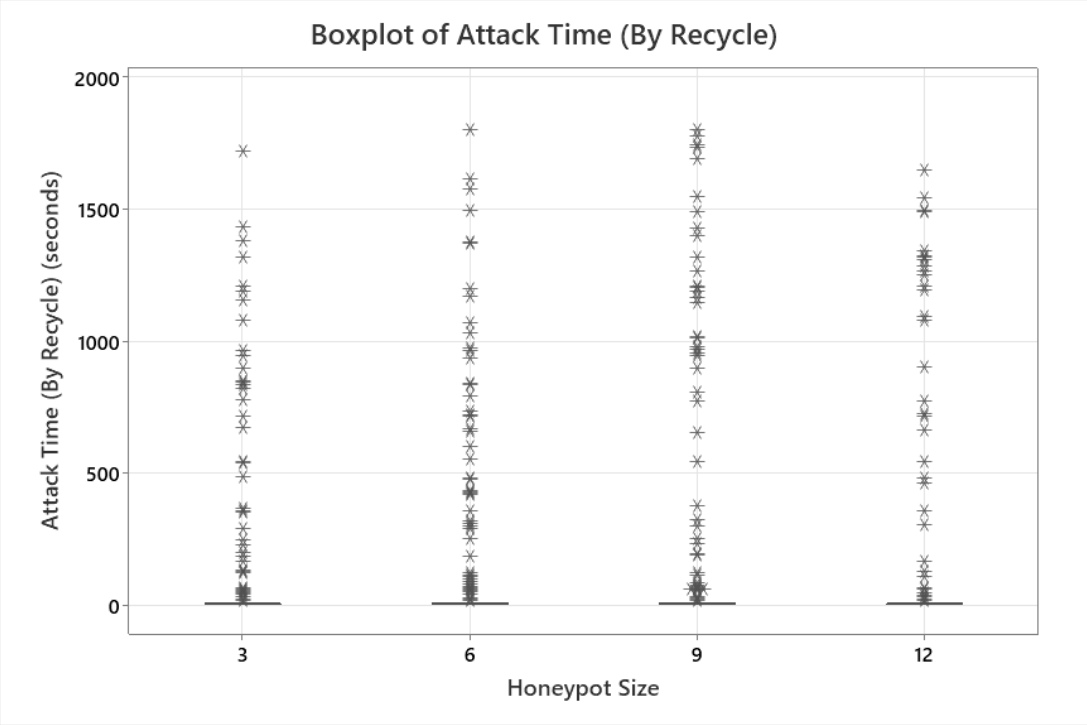

Figure B2:

ATTACK_TIME_BY_RECYCLE.CSV

Kruskal-Wallis Test: Attack Time (By Recycle) versus Honeypot Size

Descriptive Statistics

Honeypot Size	N	Median	Mean Rank	Z-Value
3	1134	4.08400	2187.3	-0.97
6	1135	4.14100	2182.4	-1.12
9	1100	4.17450	2210.8	-0.25
12	1068	4.45150	2300.1	2.37
Overall	4437		2219.0	

Test

Null hypothesis H_0: All medians are equal
Alternative hypothesis H_1: At least one median is different

Method	DF	H-Value	P-Value
Not adjusted for ties	3	5.95	0.114
Adjusted for ties	3	5.95	0.114

Figure B3:

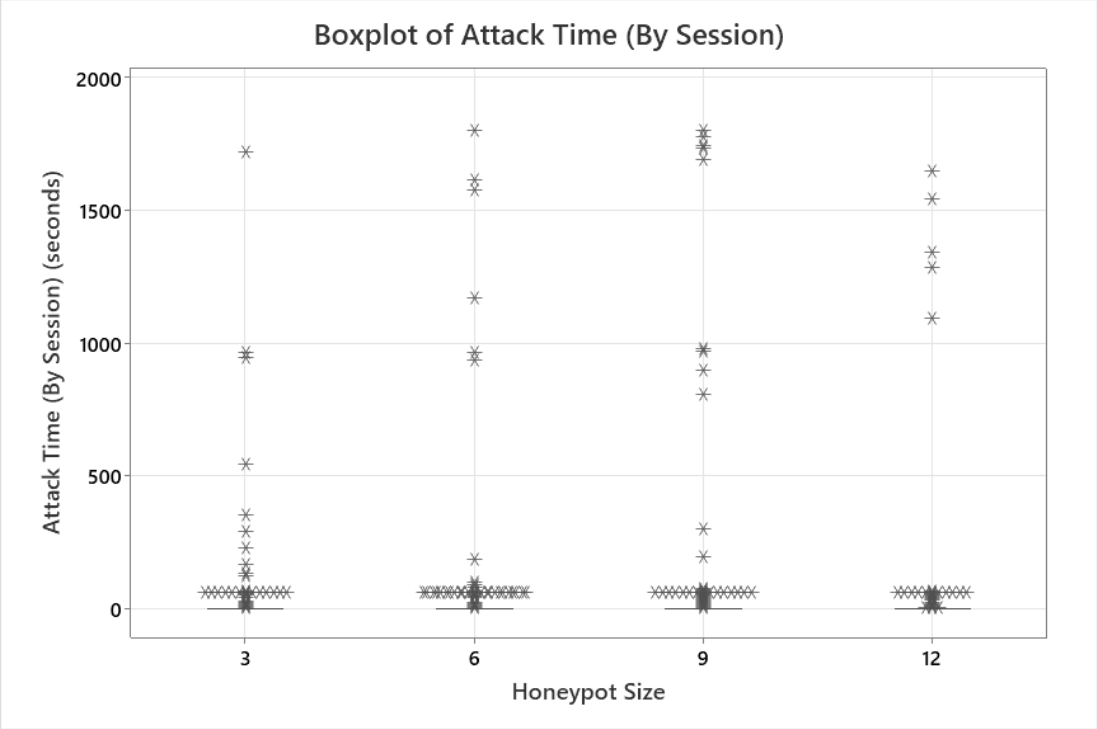

Figure B4:

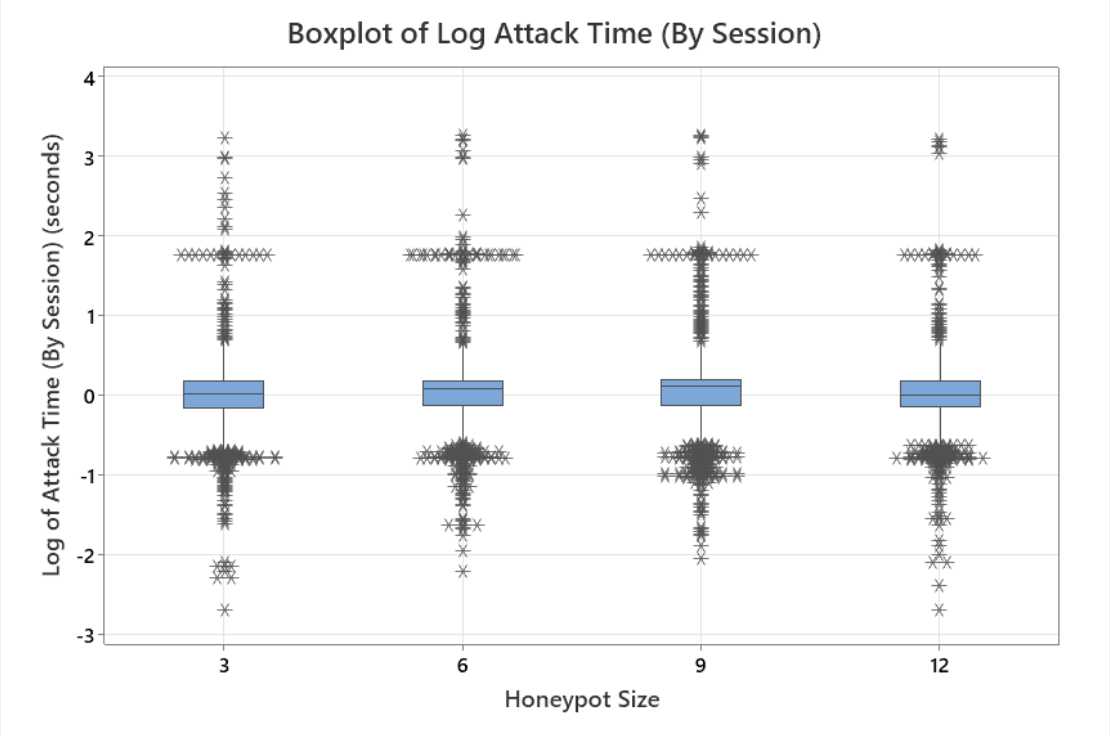

Figure B5:

ATTACK_TIME_BY_SESSION.CSV

Kruskal-Wallis Test: Attack Time (By Session) versus Honeypot Size

Descriptive Statistics

Honeypot Size	N	Median	Mean Rank	Z-Value
3	5541	1.05500	10751.2	-4.14
6	5582	1.19300	11221.8	2.21
9	5464	1.27500	11398.9	4.54
12	5529	1.00600	10865.2	-2.60
Overall	22116		11058.5	

Test

Null hypothesis H_0: All medians are equal
Alternative hypothesis H_1: At least one median is different

Method	DF	H-Value	P-Value
Not adjusted for ties	3	37.10	0.000
Adjusted for ties	3	37.10	0.000

Figure B6:

ATTACK_TIME_BY_SESSION_MWU.CSV

Mann-Whitney: Session Attack Time Size 3, Session Attack Time Size 6

Method

η_1: median of Session Attack Time Size 3
η_2: median of Session Attack Time Size 6
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Session Attack Time Size 3	5541	1.05500
Session Attack Time Size 6	5582	1.19300

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0279999	(-0.0469999, -0.0129998)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	30153402.50	0.000
Adjusted for ties	30153402.50	0.000

Figure B7:

ATTACK_TIME_BY_SESSION_MWU.CSV

Mann-Whitney: Session Attack Time Size 3, Session Attack Time Size 9

Method

η_1: median of Session Attack Time Size 3
η_2: median of Session Attack Time Size 9
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Session Attack Time Size 3	5541	1.05500
Session Attack Time Size 9	5464	1.27500

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0430000	(-0.0659997, -0.0239999)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	29612956.50	0.000
Adjusted for ties	29612956.50	0.000

Figure B8:

ATTACK_TIME_BY_SESSION_MWU.CSV

Mann-Whitney: Session Attack Time Size 3, Session Attack Time Size 12

Method

η_1: median of Session Attack Time Size 3
η_2: median of Session Attack Time Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Session Attack Time Size 3	5541	1.05500
Session Attack Time Size 12	5529	1.00600

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0059998	(-0.0199997, 0.0060000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	30514066.50	0.347
Adjusted for ties	30514066.50	0.347

Figure B9:

ATTACK_TIME_BY_SESSION_MWU.CSV

Mann-Whitney: Session Attack Time Size 6, Session Attack Time Size 9

Method

η_1: median of Session Attack Time Size 6
η_2: median of Session Attack Time Size 9
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Session Attack Time Size 6	5582	1.19300
Session Attack Time Size 9	5464	1.27500

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0100000	(-0.0250001, 0.0029998)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	30574882.00	0.125
Adjusted for ties	30574882.00	0.125

Figure B10:

ATTACK_TIME_BY_SESSION_MWU.CSV

Mann-Whitney: Session Attack Time Size 6, Session Attack Time Size 12

Method

η_1: median of Session Attack Time Size 6
η_2: median of Session Attack Time Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Session Attack Time Size 6	5582	1.19300
Session Attack Time Size 12	5529	1.00600

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
0.0200000	(0.0060000, 0.0369999)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	31516920.50	0.003
Adjusted for ties	31516920.50	0.003

Figure B11:

ATTACK_TIME_BY_SESSION_MWU.CSV

Mann-Whitney: Session Attack Time Size 9, Session Attack Time Size 12

Method

η_1: median of Session Attack Time Size 9
η_2: median of Session Attack Time Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Session Attack Time Size 9	5464	1.27500
Session Attack Time Size 12	5529	1.00600

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
0.0330002	(0.0170000, 0.0540001)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	30759244.50	0.000
Adjusted for ties	30759244.50	0.000

Figure C1:

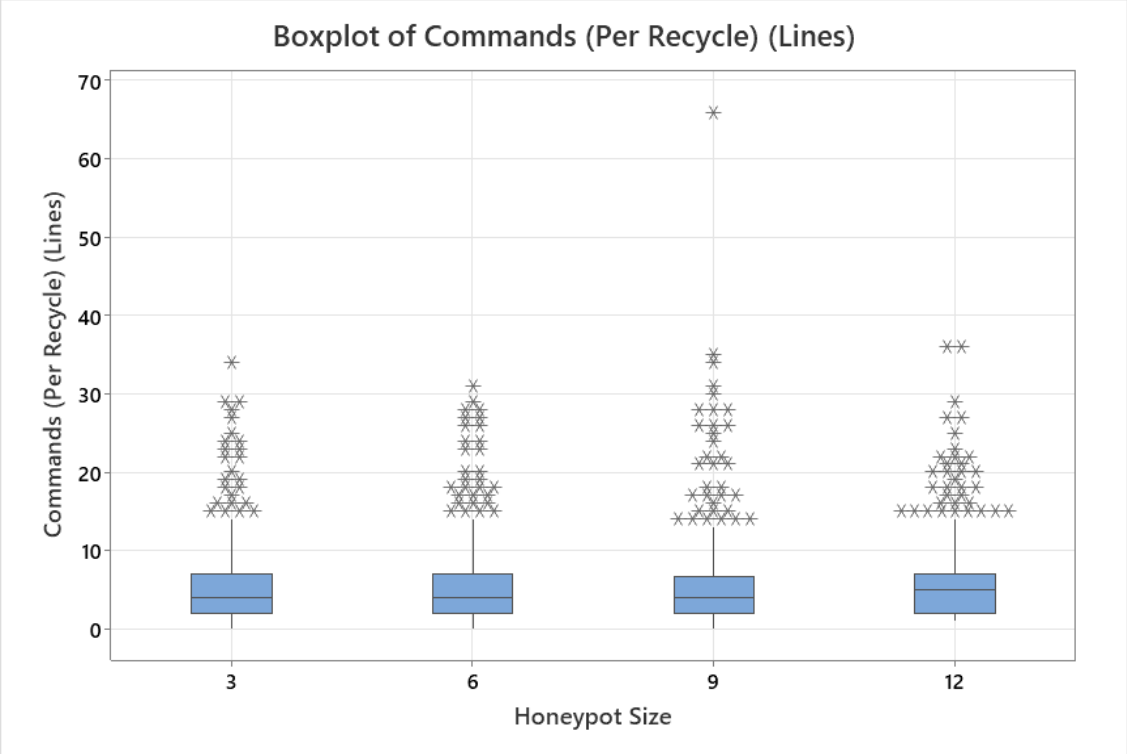

Figure C2:

COMMANDS_RECYCLE_LINE.CSV

Kruskal-Wallis Test: Commands (Per Recycle) (Lines) versus Honeypot Size

Descriptive Statistics

Honeypot Size	N	Median	Mean Rank	Z-Value
3	1134	4	2247.3	0.86
6	1135	4	2179.4	-1.21
9	1100	4	2157.9	-1.82
12	1068	5	2293.9	2.19
Overall	4437		2219.0	

Test

Null hypothesis H_0: All medians are equal
Alternative hypothesis H_1: At least one median is different

Method	DF	H-Value	P-Value
Not adjusted for ties	3	7.79	0.051
Adjusted for ties	3	7.93	0.048

Figure C3:

COMMANDS_RECYCLE_LINE_MWU.CSV

Mann-Whitney: Commands (Recycle/Line) Size 3, Commands (Recycle/Line) Size 6

Method

η_1: median of Commands (Recycle/Line) Size 3
η_2: median of Commands (Recycle/Line) Size 6
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands (Recycle/Line) Size 3	1134	4
Commands (Recycle/Line) Size 6	1135	4

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0000000	(0.0000000, -0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1306933.50	0.203
Adjusted for ties	1306933.50	0.200

Figure C4:

COMMANDS_RECYCLE_LINE_MWU.CSV

Mann-Whitney: Commands (Recycle/Line) Size 3, Commands (Recycle/Line) Size 9

Method

η_1: median of Commands (Recycle/Line) Size 3
η_2: median of Commands (Recycle/Line) Size 9
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands (Recycle/Line) Size 3	1134	4
Commands (Recycle/Line) Size 9	1100	4

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
0.0000000	(0.0000000, 0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1292887.50	0.093
Adjusted for ties	1292887.50	0.090

Figure C5:

COMMANDS_RECYCLE_LINE_MWU.CSV

Mann-Whitney: Commands (Recycle/Line) Size 3, Commands (Recycle/Line) Size 12

Method

η_1: median of Commands (Recycle/Line) Size 3
η_2: median of Commands (Recycle/Line) Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands (Recycle/Line) Size 3	1134	4
Commands (Recycle/Line) Size 12	1068	5

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
0.0000000	(-0.0000000, -0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1235675.50	0.368
Adjusted for ties	1235675.50	0.364

Figure C6:

COMMANDS_RECYCLE_LINE_MWU.CSV

Mann-Whitney: Commands (Recycle/Line) Size 6, Commands (Recycle/Line) Size 9

Method

η_1: median of Commands (Recycle/Line) Size 6
η_2: median of Commands (Recycle/Line) Size 9
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands (Recycle/Line) Size 6	1135	4
Commands (Recycle/Line) Size 9	1100	4

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
0.0000000	(-0.0000000, 0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1274414.00	0.719
Adjusted for ties	1274414.00	0.717

Figure C7:

COMMANDS_RECYCLE_LINE_MWU.CSV

Mann-Whitney: Commands (Recycle/Line) Size 6, Commands (Recycle/Line) Size 12

Method

η_1: median of Commands (Recycle/Line) Size 6
η_2: median of Commands (Recycle/Line) Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands (Recycle/Line) Size 6	1135	4
Commands (Recycle/Line) Size 12	1068	5

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
0.0000000	(0.0000000, -0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1220212.00	0.041
Adjusted for ties	1220212.00	0.039

Figure C8:

COMMANDS_RECYCLE_LINE_MWU.CSV

Mann-Whitney: Commands (Recycle/Line) Size 9, Commands (Recycle/Line) Size 12

Method

η_1: median of Commands (Recycle/Line) Size 9
η_2: median of Commands (Recycle/Line) Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands (Recycle/Line) Size 9	1100	4
Commands (Recycle/Line) Size 12	1068	5

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0000000	(0.0000000, -0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1156906.50	0.013
Adjusted for ties	1156906.50	0.013

Figure C9:

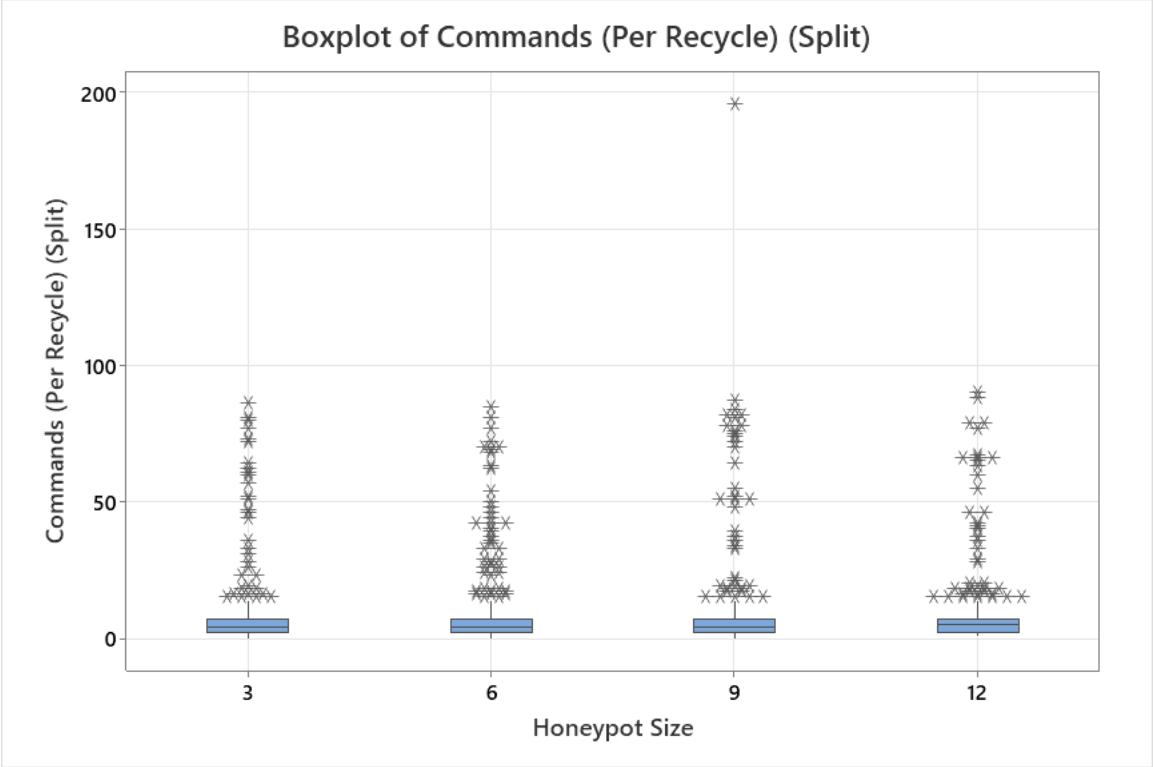

Figure C10:

COMMANDS_RECYCLE_SPLIT.CSV

Kruskal-Wallis Test: Commands (Per Recycle) (Split) versus Honeypot Size

Descriptive Statistics

Honeypot Size	N	Median	Mean Rank	Z-Value
3	1134	4	2238.6	0.60
6	1135	4	2184.0	-1.07
9	1100	4	2164.5	-1.63
12	1068	5	2291.5	2.12
Overall	4437		2219.0	

Test

Null hypothesis H_0: All medians are equal
Alternative hypothesis H_1: At least one median is different

Method	DF	H-Value	P-Value
Not adjusted for ties	3	6.52	0.089
Adjusted for ties	3	6.63	0.085

Figure C11:

COMMANDS_RECYCLE_SPLIT_MWU.CSV

Mann-Whitney: Commands(Recycle/Split) Size 3, Commands(Recycle/Split) Size 6

Method

η_1: median of Commands(Recycle/Split) Size 3
η_2: median of Commands(Recycle/Split) Size 6
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands(Recycle/Split) Size 3	1134	4
Commands(Recycle/Split) Size 6	1135	4

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
0.0000000	(0.0000000, -0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1303090.00	0.305
Adjusted for ties	1303090.00	0.301

Figure C12:

COMMANDS_RECYCLE_SPLIT_MWU.CSV

Mann-Whitney: Commands(Recycle/Split) Size 3, Commands(Recycle/Split) Size 9

Method

η_1: median of Commands(Recycle/Split) Size 3
η_2: median of Commands(Recycle/Split) Size 9
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands(Recycle/Split) Size 3	1134	4
Commands(Recycle/Split) Size 9	1100	4

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
0.0000000	(-0.0000000, 0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1288692.00	0.159
Adjusted for ties	1288692.00	0.156

Figure C13:

COMMANDS_RECYCLE_SPLIT_MWU.CSV

Mann-Whitney: Commands(Recycle/Split) Size 3, Commands(Recycle/Split) Size 12

Method

η_1: median of Commands(Recycle/Split) Size 3
η_2: median of Commands(Recycle/Split) Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands(Recycle/Split) Size 3	1134	4
Commands(Recycle/Split) Size 12	1068	5

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0000000	(0.0000000, -0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1233918.00	0.309
Adjusted for ties	1233918.00	0.305

Figure C14:

COMMANDS_RECYCLE_SPLIT_MWU.CSV

Mann-Whitney: Commands(Recycle/Split) Size 6, Commands(Recycle/Split) Size 9

Method

η_1: median of Commands(Recycle/Split) Size 6
η_2: median of Commands(Recycle/Split) Size 9
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands(Recycle/Split) Size 6	1135	4
Commands(Recycle/Split) Size 9	1100	4

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0000000	(-0.0000000, 0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1273769.00	0.751
Adjusted for ties	1273769.00	0.749

Figure C15:

COMMANDS_RECYCLE_SPLIT_MWU.CSV

Mann-Whitney: Commands(Recycle/Split) Size 6, Commands(Recycle/Split) Size 12

Method

η_1: median of Commands(Recycle/Split) Size 6
η_2: median of Commands(Recycle/Split) Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands(Recycle/Split) Size 6	1135	4
Commands(Recycle/Split) Size 12	1068	5

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0000000	(-0.0000000, -0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1222252.50	0.056
Adjusted for ties	1222252.50	0.054

Figure C16:

COMMANDS_RECYCLE_SPLIT_MWU.CSV

Mann-Whitney: Commands(Recycle/Split) Size 9, Commands(Recycle/Split) Size 12

Method

η_1: median of Commands(Recycle/Split) Size 9
η_2: median of Commands(Recycle/Split) Size 12
Difference: $\eta_1 - \eta_2$

Descriptive Statistics

Sample	N	Median
Commands(Recycle/Split) Size 9	1100	4
Commands(Recycle/Split) Size 12	1068	5

Estimation for Difference

Difference	CI for Difference	Achieved Confidence
-0.0000000	(0.0000000, -0.0000000)	95.00%

Test

Null hypothesis H_0: $\eta_1 - \eta_2 = 0$
Alternative hypothesis H_1: $\eta_1 - \eta_2 \neq 0$

Method	W-Value	P-Value
Not adjusted for ties	1159251.00	0.021
Adjusted for ties	1159251.00	0.020

Figure C17:

Number of Commands	Size 3	Size 6	Size 9	Size 12
1	5231	5055	4770	5107
2	14	25	17	13
3	389	500	583	430
9	1	1	3	3

Figure D1:

COMMAND_BINNING.CSV

Chi-Square Test for Association: cmd, Worksheet columns

Rows: cmd Columns: Worksheet columns

	Size 3	Size 6	Size 9	Size 12	All
uname -a	4867	4681	4419	4712	18679
	4795.7	4708.9	4475.7	4698.7	
	1.0592	0.1654	0.7176	0.0377	
echo > /dev/shm/ifconfig; chmod	389	452	476	407	1724
	442.6	434.6	413.1	433.7	
	6.4973	0.6955	9.5815	1.6403	
PATH=/dev/shm/:/tmp/:./:/var/tm	197	204	168	198	767
	196.9	193.4	183.8	192.9	
	0.0000	0.5857	1.3551	0.1328	
nproc; uname -a	14	25	17	13	69
	17.7	17.4	16.5	17.4	
	0.7792	3.3252	0.0132	1.0937	
echo PROC:`nproc` VER:`uname -a	84	86	105	101	376
	96.5	94.8	90.1	94.6	
	1.6279	0.8148	2.4664	0.4354	
cat /proc/cpuinfo | grep name |	83	84	73	89	329
	84.5	82.9	78.8	82.8	
	0.0255	0.0136	0.4314	0.4705	
All	5634	5532	5258	5520	21944

Cell Contents
Count
Expected count
Contribution to Chi-square

Chi-Square Test

	Chi-Square	DF	P-Value
Pearson	33.965	15	0.003
Likelihood Ratio	33.577	15	0.004

Figure D2:

COMMAND_BINNING.CSV

Chi-Square Test for Association: cmd, Worksheet columns

Rows: cmd Columns: Worksheet columns

	Size 3	Size 6	All
uname -a	4867	4681	9548
	4817.6	4730.4	
	0.5063	0.5157	
echo > /dev/shm/ifconfig; chmod	389	452	841
	424.3	416.7	
	2.9434	2.9977	
PATH=/dev/shm/:/tmp/:./:/var/tm	197	204	401
	202.3	198.7	
	0.1405	0.1431	
nproc; uname -a	14	25	39
	19.7	19.3	
	1.6384	1.6686	
echo PROC:`nproc` VER:`uname -a	84	86	170
	85.8	84.2	
	0.0368	0.0375	
cat /proc/cpuinfo | grep name |	83	84	167
	84.3	82.7	
	0.0189	0.0193	
All	5634	5532	11166

Cell Contents
- *Count*
- *Expected count*
- *Contribution to Chi-square*

Chi-Square Test

	Chi-Square	DF	P-Value
Pearson	10.666	5	0.058
Likelihood Ratio	10.712	5	0.057

Figure D3:

COMMAND_BINNING.CSV

Chi-Square Test for Association: cmd, Worksheet columns

Rows: cmd Columns: Worksheet columns

	Size 3	Size 9	All
uname -a	4867	4419	9286
	4803.3	4482.7	
	0.8453	0.9058	
echo > /dev/shm/ifconfig; chmod	389	476	865
	447.4	417.6	
	7.6304	8.1761	
PATH=/dev/shm/:/tmp/:./:/var/tm	197	168	365
	188.8	176.2	
	0.3561	0.3816	
nproc; uname -a	14	17	31
	16.0	15.0	
	0.2583	0.2767	
echo PROC:`nproc` VER:`uname -a	84	105	189
	97.8	91.2	
	1.9373	2.0759	
cat /proc/cpuinfo | grep name |	83	73	156
	80.7	75.3	
	0.0660	0.0707	
All	5634	5258	10892

Cell Contents
- *Count*
- *Expected count*
- *Contribution to Chi-square*

Chi-Square Test

	Chi-Square	DF	P-Value
Pearson	22.980	5	0.000
Likelihood Ratio	22.982	5	0.000

Figure D4:

COMMAND_BINNING.CSV

Chi-Square Test for Association: cmd, Worksheet columns

Rows: cmd Columns: Worksheet columns

	Size 3	Size 12	All
uname -a	4867	4712	9579
	4838.5	4740.5	
	0.16845	0.17193	
echo > /dev/shm/ifconfig; chmod	389	407	796
	402.1	393.9	
	0.42472	0.43349	
PATH=/dev/shm/:/tmp/:./:/var/tm	197	198	395
	199.5	195.5	
	0.03179	0.03245	
nproc; uname -a	14	13	27
	13.6	13.4	
	0.00961	0.00981	
echo PROC:`nproc` VER:`uname -a	84	101	185
	93.4	91.6	
	0.95474	0.97445	
cat /proc/cpuinfo | grep name |	83	89	172
	86.9	85.1	
	0.17319	0.17676	
All	5634	5520	11154

Cell Contents
- *Count*
- *Expected count*
- *Contribution to Chi-square*

Chi-Square Test

	Chi-Square	DF	P-Value
Pearson	3.561	5	0.614
Likelihood Ratio	3.563	5	0.614

Figure D5:

COMMAND_BINNING.CSV

Chi-Square Test for Association: cmd, Worksheet columns

Rows: cmd Columns: Worksheet columns

	Size 6	Size 9	All
uname -a	4681	4419	9100
	4665.5	4434.5	
	0.0512	0.0539	
echo > /dev/shm/ifconfig; chmod	452	476	928
	475.8	452.2	
	1.1888	1.2508	
PATH=/dev/shm/:/tmp/:./:/var/tm	204	168	372
	190.7	181.3	
	0.9242	0.9724	
nproc; uname -a	25	17	42
	21.5	20.5	
	0.5581	0.5872	
echo PROC:`nproc` VER:`uname -a	86	105	191
	97.9	93.1	
	1.4522	1.5279	
cat /proc/cpuinfo | grep name |	84	73	157
	80.5	76.5	
	0.1528	0.1607	
All	5532	5258	10790

Cell Contents
- *Count*
- *Expected count*
- *Contribution to Chi-square*

Chi-Square Test

	Chi-Square	DF	P-Value
Pearson	8.880	5	0.114
Likelihood Ratio	8.893	5	0.113

Figure D6:

COMMAND_BINNING.CSV

Chi-Square Test for Association: cmd, Worksheet columns

Rows: cmd Columns: Worksheet columns

	Size 6	Size 12	All
uname -a	4681	4712	9393
	4701.6	4691.4	
	0.0903	0.0904	
echo > /dev/shm/ifconfig; chmod	452	407	859
	430.0	429.0	
	1.1291	1.1316	
PATH=/dev/shm/:/tmp/:./:/var/tm	204	198	402
	201.2	200.8	
	0.0385	0.0385	
nproc; uname -a	25	13	38
	19.0	19.0	
	1.8797	1.8838	
echo PROC:`nproc` VER:`uname -a	86	101	187
	93.6	93.4	
	0.6173	0.6187	
cat /proc/cpuinfo | grep name |	84	89	173
	86.6	86.4	
	0.0777	0.0779	
All	5532	5520	11052

Cell Contents
- *Count*
- *Expected count*
- *Contribution to Chi-square*

Chi-Square Test

	Chi-Square	DF	P-Value
Pearson	7.673	5	0.175
Likelihood Ratio	7.741	5	0.171

Figure D7:

COMMAND_BINNING.CSV

Chi-Square Test for Association: cmd, Worksheet columns

Rows: cmd Columns: Worksheet columns

	Size 9	Size 12	All
uname -a	4419	4712	9131
	4454.5	4676.5	
	0.2832	0.2698	
echo > /dev/shm/ifconfig; chmod	476	407	883
	430.8	452.2	
	4.7496	4.5241	
PATH=/dev/shm/:/tmp/:./:/var/tm	168	198	366
	178.6	187.4	
	0.6235	0.5939	
nproc; uname -a	17	13	30
	14.6	15.4	
	0.3821	0.3639	
echo PROC:`nproc` VER:`uname -a	105	101	206
	100.5	105.5	
	0.2018	0.1923	
cat /proc/cpuinfo | grep name |	73	89	162
	79.0	83.0	
	0.4602	0.4384	
All	5258	5520	10778

Cell Contents
- *Count*
- *Expected count*
- *Contribution to Chi-square*

Chi-Square Test

	Chi-Square	DF	P-Value
Pearson	13.083	5	0.023
Likelihood Ratio	13.089	5	0.023

Figure E1:

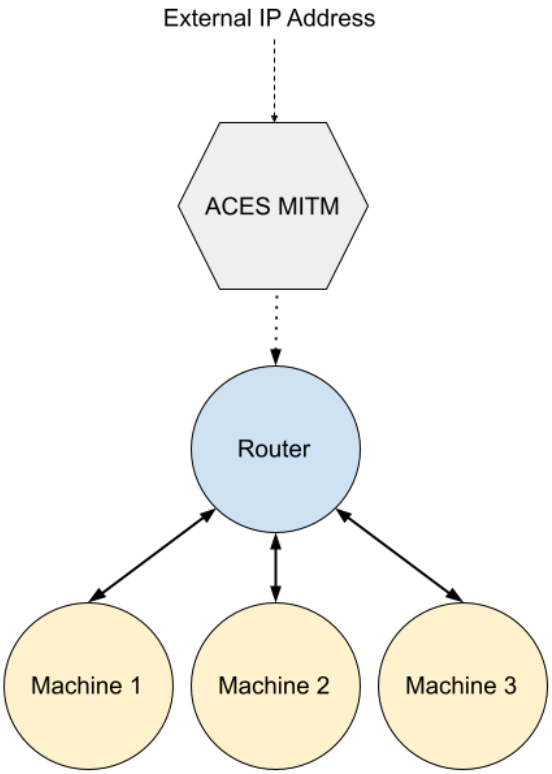

Figure E2:

```
→ sudo lxc-attach 128.8.238.29-honeypot-router
root@128:/# cat ~/.ssh/config
Host machine0_BP3SDFOI
        HostName 10.0.3.136
        User root

Host machine1_1MGIE965
        HostName 10.0.3.153
        User root

Host machine2_Y2F9BD06
        HostName 10.0.3.248
        User root
```

Appendix C

Script C1: Recycle Script (recycle_script.rb)

```
#!/usr/bin/ruby

require 'rufus-scheduler'
require './network'
require './ssh_key_utils'
require './mitm'
require './data_utils'
require './nat_rules'
require './logger'

# param: ip address
if ARGV.length != 1
  puts "Usage: #{$0} [external ip address]"
  exit(1)
end
EXTERNAL_IP = ARGV[0]
logger = HoneypotLogger.new(EXTERNAL_IP)
logger.log "beginning recycling script"

# get relevant directories
HOME_DIR = `cd && pwd`.chomp
HONEYPOT_DIR = "#{HOME_DIR}/#{EXTERNAL_IP}_files"
`mkdir #{HONEYPOT_DIR}`
`cd #{HONEYPOT_DIR} && sudo rm -rf *`

# create network
network_size = [3, 6, 9, 12].sample
n = Network.create_fresh(network_size, "#{EXTERNAL_IP}-honeypot")
n.create_and_start_all
# n.create_and_start_all_with_random_honey
n.write_to_file "#{HONEYPOT_DIR}/network_layout.txt"
logger.log "created network size=#{network_size}"
sleep(5)

# debug: check whether containers are up
logger.log "checking whether containers are up"
([n.router] + n.containers).each do |container|
  counter = 0
  while !container.running?
    counter += 1
    logger "#{container.name} FAILED TO START; counter=#{counter}"
    container.start
    sleep([2, 4, 6, 8, 10].sample)
  end
  logger.log "#{container.name} up"
  sleep(2)

  #if container.running?
    #logger.log "#{container} up"
  #else
    #logger.log "#{container} IS NOT UP, attempting again"
    #container.start
    #sleep(2)
    #if container.running?
      #logger.log "#{container} up after second attempt"
    #else
      #logger.log "#{container} STILL NOT UP"
    #end
  #end
end

# create MITM
port = MITM.get_port_from_external_ip(EXTERNAL_IP)
mitm = MITM.new(n, port)
logger.log "created mitm: external ip #{EXTERNAL_IP}, port #{port}"
sleep(3)

# connect MITM to router container and external IP
initialize_ssh(n.router)
sleep(3)
mitm.start("#{HONEYPOT_DIR}/mitm.log")
sleep(3)
logger.log "mitm started"
```

```
n.containers.each_with_index do |container, index|
  # set up ssh keys and alias
  initialize_ssh(container)
  place_public_key(n.router, container)
  random = [*('A'..'Z'),*('0'..'9')].shuffle[0,8].join
  add_alias(n.router, container, "machine#{index}_#{random}")
  logger.log "connected container #{index} to router"

  # upload random honey
  honeytype = ["healthcare", "financial", "PII"].sample
  num = [0, 1, 2].sample
  honey_dir = `pwd`.chomp + "/honey/#{honeytype}"
  honey_filename = "#{honeytype}#{num}.tar.gz"
  container.upload_honey(honey_dir, honey_filename)
end

# add banner to router container
random = [*('A'..'Z'),*('0'..'9')].shuffle[0,16].join
n.router.run "echo \'PrintMotd yes\n\' >> /etc/ssh/sshd_config"
n.router.run "echo \'------------------------------------------------------------\nAuthorized access only!\nThis is a
router machine for internal use only.\nUnique identifier: #{random}\nNumber of machines accessible by ssh:
#{network_size}\nPlease contact the IT department if you need to be given permission to access the
network.\n------------------------------------------------------------\n\' > /etc/motd"
n.router.run "sudo service ssh restart"
sleep(3)

# allow connections in firewall rules
allow_container_connections(n)

# enforce key login on containers
n.containers.each do |container|
  enforce_key_login(container)
end

# create log files
n.redirect_auth_logs_to HONEYPOT_DIR
logger.log "auth log files redirected"

# connect containers to internet through external IP
allow_network_internet(n, EXTERNAL_IP)

# connect honeypot network to external ip
mitm.connect_to_external_ip(EXTERNAL_IP)
`sudo iptables -A INPUT -p tcp -s 0.0.0.0/0 -d 127.0.0.1 --dport #{port} -j ACCEPT`
logger.log "mitm connected to external ip"
logger.log "ready to accept attackers"

# wait until attacker enters
`sudo chmod a+r #{HONEYPOT_DIR}/mitm.log`
`sudo tail -n 0 -f "#{HONEYPOT_DIR}/mitm.log" | grep -Eq "Compromising the honeypot"`
logger.log "attacker entered"

# only allow that ip address
attacker_ip = `cd #{HONEYPOT_DIR} && cat mitm.log | grep "Threshold: 2, Attempts: 2" | awk '{ print $8 }' | cut -d','
-f1`.chomp
logger.log "attacker ip address: #{attacker_ip}"
# allow attacker only in
`sudo iptables -I INPUT -p tcp -s #{attacker_ip} -d 127.0.0.1 --dport #{port} -j ACCEPT`
# don't allow anyone else in
`sudo iptables -D INPUT -p tcp -s 0.0.0.0/0 -d 127.0.0.1 --dport #{port} -j ACCEPT`

# put ssh in attacker's home directory
attacker_username = `cat #{HONEYPOT_DIR}/mitm.log | grep "Adding the following credentials" | cut -d':' -f4 | colrm 1
2`.chomp
logger.log "attacker username: #{attacker_username}"
sleep(1)
n.router.run "cp -r ~/.ssh /home/#{attacker_username}/.ssh"
n.router.run "chmod a+x /home/#{attacker_username}/.ssh"
n.router.run "chmod a+r /home/#{attacker_username}/.ssh -R"
logger.log "put .ssh in attacker's home directory"

# give attacker sudo on every machine
([n.router] + n.containers).each do |container|
  logger.log "granting sudo for container #{container.name}"
  container.run "sudo adduser #{attacker_username} sudo"
  container.run "echo '#{attacker_username} ALL=(ALL) NOPASSWD: ALL' | sudo EDITOR='tee -a' visudo"
end
logger.log "attacker granted sudo access"

# timer to destroy honeypot
scheduler = Rufus::Scheduler.new
```

```
scheduler.in '30m' do
  logger.log "beginning honeypot destruction"

  # disconnect mitm (kick attacker out)
  mitm.disconnect_from_external_ip(EXTERNAL_IP)
  mitm.stop
  logger.log "mitm stopped"
  sleep(2)

  # connect containers to internet through external IP
  disallow_network_internet(n, EXTERNAL_IP)
  sleep(2)

  # disallow connections in firewall rules
  disallow_container_connections(n)
  sleep(2)

  # process and package logs/data
  get_duration_calculations(EXTERNAL_IP)
  get_mitm_commands(EXTERNAL_IP)
  package_honeypot_data(EXTERNAL_IP)
  clear_honeypot_dir(EXTERNAL_IP)
  logger.log "logs retrieved"
  sleep(1)

  # stop honeypot containers
  n.stop_and_destroy_all
  logger.log "honeypot destroyed"
  sleep(3)

  # call recycle script again
  `nohup ./recycle_script.rb #{EXTERNAL_IP} &`
  exit(0)
end
scheduler.join
```

Script C2: Single-Use Data Pipeline (data_pipeline.rb)

```
#!/usr/bin/ruby

########################
# PROCESSING FUNCTIONS #
########################

def process_size_each(home_dir, size)
  data_path = home_dir + "size_#{size}_data/"
  data_files = `cd #{data_path} && ls`.split
  results = {}
  for file in data_files
    `cd #{data_path} && sudo tar -xzf #{file}`
    res = yield data_path, file
    key = file.split(".")[0]
    results[key] = res
    `cd #{data_path} && sudo rm *.processed *.txt *.log`
  end
  return results # hash: timestamp => produced value
end

def process_each(home_dir, &block)
  results = {}
  for size in [3, 6, 9, 12]
    results[size] = process_size_each(home_dir, size, &block)
  end
  return results # hash: size => (timestamp => produced value)
end

####################
# UTILITY FUNCTIONS #
####################

# MITM log line -> UNIX epoch time
def get_time(line)
  timestamp = `echo "#{line.chomp}" | awk '{print $1, $2}'`.chomp
  `date -d"#{timestamp}" +%s.%N`.chomp.to_f
end

# prints ruby dictionary of arrays in Python-readable format
```

```
def print_as_py_dict(results)
  puts "{"
  results.map { |k,v| puts "#{k}: #{v.values}," }
  puts "}"
end

######################
# METRIC CALCULATIONS #
######################

def calc_recycles_and_sessions(home_dir)
  results = process_each(home_dir) do |data_dir, file|
    attacker_entered = false
    num_sessions = 0
    cutoff = nil
    File.foreach(data_dir + "mitm.log").each do |line|
      attacker_entered = true if line =~ /Threshold: 2, Attempts: 2/
      next if !attacker_entered
      if line =~ /Attacker authenticated/
        cutoff = get_time(line) + 60*30 if !cutoff
        next if get_time(line) > cutoff
        num_sessions += 1
      end
    end
    num_sessions
  end
  results.map do |size, data|
    puts "#{size}: #{data.values},"
  end
end

def calc_time_recycle(home_dir)
  results = process_each(home_dir) do |data_dir, file|
    attacker_entered = false
    times = []
    last_time = nil
    cutoff = nil
    File.foreach(data_dir + "mitm.log").each do |line|
      attacker_entered = true if line =~ /Threshold: 2, Attempts: 2/
      next if !attacker_entered
      if line =~ /Attacker authenticated/
        curr_time = get_time(line)
        cutoff = curr_time + 60*30 if !cutoff
        next if curr_time > cutoff
        last_time = curr_time
      elsif last_time && line =~ /Attacker closed connection/
        times.append(get_time(line) - last_time)
        last_time = nil
      end
    end
    if last_time
      last_line = `cd #{data_dir} && cat mitm.log | tail -n 2 | head -n 1`.chomp
      final = get_time(last_line)
      times.append([cutoff, final].min - last_time)
    end
    times
  end
  results.map do |size, data|
    totals = data.map do |timestamp, times|
      times.sum
    end
    puts "#{size}: #{totals},"
  end
end

def calc_time_session(home_dir)
  results = process_each(home_dir) do |data_dir, file|
    attacker_entered = false
    times = []
    last_time = nil
    cutoff = nil
    File.foreach(data_dir + "mitm.log").each do |line|
      attacker_entered = true if line =~ /Threshold: 2, Attempts: 2/
      next if !attacker_entered
      if line =~ /Attacker authenticated/
        curr_time = get_time(line)
        cutoff = curr_time + 60*30 if !cutoff
        next if curr_time > cutoff
        last_time = curr_time
      elsif last_time && line =~ /Attacker closed connection/
        times.append(get_time(line) - last_time)
```

```
        last_time = nil
      end
    end
    if last_time
      last_line = `cd #{data_dir} && cat mitm.log | tail -n 2 | head -n 1`.chomp
      final = get_time(last_line)
      times.append([cutoff, final].min - last_time)
    end
    times
  end
  results.map do |size, data|
    lst = []
    data.map do |timestamp, times|
      lst += times
    end
    puts "#{size}: #{lst},"
  end
end

def calc_commands_recycle_split(home_dir)
  results = process_each(home_dir) do |data_dir, file|
    File.read(data_dir + "mitm_commands.processed").split(/;|\n */).length
  end
  results.map do |size, data|
    puts "#{size}: #{data.values},"
  end
end

def calc_commands_recycle_lines(home_dir)
  results = process_each("/home/sumit/") do |data_dir, file|
    File.read(data_dir + "mitm_commands.processed").split(/\n */).length
  end
  results.map do |size, data|
    puts "#{size}: #{data.values},"
  end
end

def calc_commands_session_split(home_dir)
  results = process_each("/home/sumit/") do |data_dir, file|
    attacker_entered = false
    commands = []
    curr_commands = 0
    cutoff_time = nil
    File.foreach(data_dir + "mitm.log").each do |line|
      attacker_entered = true if line =~ /Threshold: 2, Attempts: 2/
      next if !attacker_entered
      if line =~ /Attacker authenticated/
        cutoff_time = get_time(line) + 60*30 if !cutoff_time
      elsif cutoff_time && line =~ /command/
        next if get_time(line) > cutoff_time
        stripped = line.chomp.split(':')[3..].join(':')
        num = stripped.chomp.split(/;|\n */).length
        if line =~ /Noninteractive/
          commands.append num
        else
          curr_commands += num
        end
      elsif cutoff_time && line =~ /Attacker closed connection/
        commands.append(curr_commands) if curr_commands > 0
        curr_commands = 0
      end
    end
    commands.append(curr_commands) if curr_commands > 0
    commands
  end
  results.map do |size, data|
    lst = []
    for v in data.values
        lst += v
    end
    puts "#{size}: #{lst},"
  end
end

def calc_command_binning(home_dir)
  results = process_each("/home/sumit/") do |data_dir, file|
    attacker_entered = false
    commands = []
    curr_commands = []
    cutoff_time = nil
    File.foreach(data_dir + "mitm.log").each do |line|
```

```
      attacker_entered = true if line =~ /Threshold: 2, Attempts: 2/
      next if !attacker_entered
      if line =~ /Attacker authenticated/
        cutoff_time = get_time(line) + 60*30 if !cutoff_time
      elsif cutoff_time && line =~ /command/
        next if get_time(line) > cutoff_time
        curr_commands.append line.chomp
      elsif cutoff_time && line =~ /Attacker closed connection/
        commands.append(curr_commands) if curr_commands.length > 0
        curr_commands = []
      end
    end
    commands.append(curr_commands) if curr_commands.length > 0
    commands
  end
  results.map do |size, data|
    hash = {}
    data.each do |timestamp, sessions|
      sessions.each do |session|
        session.each do |line|
          l = line.split(':')[3..].join(':').strip
          if hash.has_key?(l)
            hash[l] += 1
          else
            hash[l] = 1
          end
        end
      end
    end
    puts "#{size}: {"
    for k, v in hash
      puts "'#{k}': #{v},"
    end
    puts "},"
  end
end
```

Script C3: Duration Preprocessing (duration_calculation.sh)

```
#!/bin/bash

entryExitCount=`cat $1 | grep "sshd\[.*\]: pam_unix(sshd:session)" | wc -l`
pairCount=$((entryExitCount / 2))
timeTotal=0
n1=1
for (( c=1; c<=$pairCount; c++ ))
do
   n2=$((2 * c))
   entryMonthDate=`cat $1 | grep "sshd\[.*\]: pam_unix(sshd:session)" | head -$n1 | tail -1 | colrm 7`
   exitMonthDate=`cat $1 | grep "sshd\[.*\]: pam_unix(sshd:session)" | head -$n2 | tail -1 | colrm 7`
   entryTimeStamp=`cat $1 | grep "sshd\[.*\]: pam_unix(sshd:session)" | head -$n1 | tail -1 | awk '{print $3}'`
   exitTimeStamp=`cat $1 | grep "sshd\[.*\]: pam_unix(sshd:session)" | head -$n2 | tail -1 | awk '{print $3}'`
   entryMonthDateStamp=`date -d"$entryMonthDate" "+%Y-%m-%d"`
   exitMonthDateStamp=`date -d"$exitMonthDate" "+%Y-%m-%d"`
   entryStamp=`date -d"$entryMonthDateStamp $entryTimeStamp" +%s`
   exitStamp=`date -d"$exitMonthDateStamp $exitTimeStamp" +%s`
   timeElapsed=$((exitStamp - entryStamp))
   timeTotal=$((timeTotal + timeElapsed))
   n1=$((n1 + 2))
done
echo "$timeTotal" # seconds total spent in container
```

Script C4: Data Packaging (data_utils.rb)

```
require './network'

def home_dir
  `cd && pwd`.chomp
end

def get_honeypot_dir ip
  "#{home_dir}/#{ip}_files"
end
```

```
def package_honeypot_data ip
  timestamp = `date +%s`.chomp
  honeypot_dir = get_honeypot_dir(ip)
  `echo #{ip} >> #{honeypot_dir}/external_ip.txt`
  `cd #{honeypot_dir} && tar -czf #{timestamp}.tar.gz *`
  honeypot_size = Network.create_from_file("#{honeypot_dir}/network_layout.txt").size
  destination = "#{home_dir}/size_#{honeypot_size}_data"
  `mkdir #{destination}`
  `mv #{honeypot_dir}/#{timestamp}.tar.gz #{destination}`
end

def get_duration_calculations ip
  honeypot_dir = get_honeypot_dir(ip)
  network = Network.create_from_file("#{honeypot_dir}/network_layout.txt")
  `touch #{honeypot_dir}/duration.processed`
  File.open("#{honeypot_dir}/duration.processed", "a") do |file|
    file.puts "#{network.router.name} " + `./duration_calculation.sh
#{honeypot_dir}/#{network.router.name}.log`.chomp
    for container in network.containers
      file.puts "#{container.name} " + `./duration_calculation.sh #{honeypot_dir}/#{container.name}.log`.chomp
    end
  end
end

def get_mitm_commands ip
  `./mitm_scrape.sh #{get_honeypot_dir(ip)}`
end

def clear_honeypot_dir ip
  `cd #{get_honeypot_dir(ip)} && rm -f *`
end
```

All other scripts used for our honeypot implementation and data analysis can be found on our GitHub page: https://github.com/SumitNawathe/HoneypotResearchProject

Works Cited

Bagchi, Kallol Kumar, and Zaiyong Tang. *Network Size, Deterrence Effects and Internet Attack Incident Growth*. p. 21.

Bar, Ariel, et al. "Identifying Attack Propagation Patterns in Honeypots Using Markov Chains Modeling and Complex Networks Analysis." *2016 IEEE International Conference on Software Science, Technology and Engineering (SWSTE)*, 2016, pp. 28–36. *IEEE Xplore*, https://doi.org/10.1109/SWSTE.2016.13.

Cao, Phuong M., et al. *CAUDIT: Continuous Auditing of SSH Servers to Mitigate Brute-Force Attacks*. p. 17.

Curella, Flavio. *Faker*. 15.1.1, https://github.com/joke2k/faker.

Katakwar, Harsh, et al. "Influence of Network Size on Adversarial Decisions in a Deception Game Involving Honeypots." *Frontiers in Psychology*, vol. 11, 2020. *Frontiers*, https://www.frontiersin.org/articles/10.3389/fpsyg.2020.535803.

"Lateral Movement Explained | What Is Lateral Movement?" *Crowdstrike.Com*, https://www.crowdstrike.com/cybersecurity-101/lateral-movement/. Accessed 18 Sept. 2022.

Rege, A., et al. "A Qualitative Exploration of Adversarial Adaptability, Group Dynamics, and Cyber-Intrusion Chains." *Journal of Information Warfare*, vol. 16, no. 3, 2017, pp. 1–16.

Rocklin, Matthew. *FakeStockData*.

Walonoski, Jason, et al. *Synthea*. 3.1.1, https://github.com/synthetichealth/synthea.