HACS 200: Honeypot Project

Group 1B: HUNNYPOT

David Hardy, Abhinav Inavolu, John McKee, Ivan Mladenov, Matthew Ritter

Section 1: Executive Summary

Section 1.1: Project Overview

The goal of this project was to use high-interaction honeypots to study whether certain honey has an impact on attacker engagement. Our honey was split into three organization types, two levels of CPU allocations, and two levels of RAM allocations (12 configurations). We wanted to test whether any of these variables had a significant impact on how much time was spent on our machines or how many commands an attacker would execute as a result of these variables (for more detail on the time spent/number of commands executed, see section 6.2, here we will just group them as attacker engagement). We used an Ubuntu Virtual Machine (VM) running Linux Containers (LXC) to spawn and recycle honeypots for attackers to interact with continuously. Using various scripts and tools to monitor, recycle, kick out attackers, randomize containers, etc., we were able to collect over 12,000 data points for statistical analysis and draw conclusions for our research question and hypotheses.

Section 1.2: Research Question and Hypothesis

Research Question: How do organization type, RAM allocation, and CPU allocation affect attacker engagement?

Null Hypothesis: Organization type, RAM allocation, and CPU allocation have no impact on attacker engagement.

Section 1.3: Collected Data

The data we collected (to answer our research question, not any other supplemental observations we made later on in the project), was the IP address of each attacker (to filter out repeat attacks), the number of commands the attacker ran on the container, and the time spent on the honeypot. See section 6.2 for more information about collected data.

Section 1.4: Findings

We found that organization type did not significantly impact attacker engagement. This variable was deemed irrelevant however as attackers did not interact with any of the user data. In contrast, we

were able to identify statistically significant differences in attacker engagement during their first attack (unique IPs) based on RAM and CPU allocations.

Section 2: Background Research

Section 2.1: Similar Research

We looked at a study done by Connolly et al. "*An Empirical Study of Ransomware Attacks on Organizations*," which identifies a lack of security awareness as a key vulnerability. This study analyzed the effects of ransomware on different types of organizations based on levels of security, size, sectors, attack types, and security posture. They took a look at the same sectors we tested within our project analyzing hospitals, banks, restaurants, car dealerships, government entities, educational institutions, and more. The goal of the study was to see if those variables mentioned earlier affected the impact of the ransomware.

Essentially, their research question was very similar to ours. After analyzing different attacks, the researchers concluded that "several factors, including 'organization sector', 'security posture' and 'attack type', influence the degree of severity of ransomware attacks" (Connolly et al. 2020). The organization sector relates to our project because we looked at different sectors. Their results were linked to more private versus public sector organizations which is something that we didn't take into consideration. We were strictly looking at the type of organization. Since they found the sector had an effect we wanted to see a difference in attacker behavior based on our organization types.

However, they did conclude that their "results also indicate that 'organization size', 'crypto-ransomware propagation class' and 'attack target' have no significant impact on the severity level of ransomware attacks" (Connolly et al. 2020). Regarding organization size, that is something we tried to simulate with the resources of our online environment. Changing the RAM allocation and CPU allocation would mimic an organization's size, assuming that a larger organization has greater computing power (more RAM and CPU) (Connolly et al. 2020). In the ransomware research, their size included the number of exposed interfaces to the internet, employee numbers, and resource usage. We couldn't simulate the employee size, but we wanted to see if we would find similar results with resource usage.

Section 2.2: Other Research

Research has suggested that attackers don't necessarily discriminate when it comes to the size of organizations. While it can be anticipated that larger enterprise organizations might deter attackers due to increased capabilities as well as overall resources, this may work on the opposite token meaning attackers see these organizations thus as larger targets. On the other hand, smaller organizations that may not hold as many resources devoted to cybersecurity measures are targeted specifically for this reason due to not having as strong defensive capabilities, so they're perceived as weaker targets (Verizon, 2022). Studies conducted within Verizon's Data Breach Investigations Report provide data that shows cyber attacks occur across various industries as well as different organizations with a range of defense capabilities. This report was a reason why we decided to test different organizations. Our null hypothesis was that organization types, resource allocations, and quantities don't impact attacker engagement. Verizon's report supported not rejecting the null hypothesis.

We found another research study on cyber deception techniques to improve honeypot performance (Javadpour et al. 2024). We wanted to model our honey after some of the research we looked at within this particular study. The study highlights how you should model your honey to create fake databases that will lure attackers in. A method was brought up in the research and it's something we took into account when making our file names and generating our honey. The research method is that "the filenames are created by combining different parts of real filenames, and the file contents are generated by extracting data from different websites." We took a look at sample organizations like hospitals and found different configurations that they use for their filing systems. We tried to make our honey as real as possible because the research concluded that these types of tactics actually do increase attacker behavior. If an attacker thinks our honey is real, they will interact with it (Javadpour et al. 2024).

Finally, we researched the role of malware deployment since it is also essential when monitoring attacker engagement. With our project, we are assuming attackers will download malware since that's a common practice across attackers (Verizon 2022). Within our general overview, another assumption present is that online systems with lesser security capabilities will have higher engagement rates. This

relates to malware deployment research conducted by the National Security Agency (NSA 2020) which identifies the usage of malware in web server cyber attacks as a fairly standard strategy. The NSA has also pointed out that internet-facing web servers are targeted just as frequently as non-internet-facing web servers where attackers frequently deploy malware such as backdoors. We had some attackers use malware which supported our assumption that we would get attackers using malware. However, we saw no correlation with the use of malware and the configuration they were in, but when we did our initial resource, we thought based on the NSA findings, we would see more malware in the smaller organizations. We assumed that attackers would see RAM usage and CPU usage then attribute that to possible security within the system since those traits typically correlate based on research from the NSA (NSA 2020).

Section 3: Experiment Design Changes

Section 3.1: Change in Research Question

Changes in our independent variables (IVs) led to corresponding changes in our research question. On September 26th, based on our first shift in IVs, we modified our research question from "How does the type of organization, the amount of data, and the amount of resources allocated to the system affect attacker engagement" to "How do the organization type, RAM allocation, and disk space allocation affect attacker engagement?" On October 8th, based on our second shift in IVs, we modified our research question again to "How do organization type, RAM allocation, and CPU allocation affect attacker engagement?"

Section 3.2: Initial Experiment Design

As of September 10th, the IVs hypothesized to influence an attacker's level of engagement were organization type, amount of data, and amount of resources (RAM and disk space). We represent organization type based on the type of user data being stored in the honeypot. Our original plan was to store user data in an Apache web server.

Section 3.3: Design Modifications

Based on recommendations from teaching staff, on September 26th, we shifted our IVs to organization type, amount of RAM, and amount of disk space. Essentially, we dropped the amount of data and split the amount of resources into two separate variables for RAM and disk space. The next design modification, on October 8th, was shifting our independent variable of disk space allocation to CPU allocation. Container disk space allocation proved challenging to implement using LXC and Bash. After recommendations from the teaching staff, it became clear that CPU allocation was a significantly simpler and more manipulable variable.

On October 1st, we modified our experiment to reflect a change from using an Apache web server to using a MySQL database. However, on October 15th, the team collectively agreed to shift away from Apache and MySQL; just moving the honey data into the home directory of our system to allow for

increased simplicity and quicker recycling. We predicted that most attacks would be from bots and that most bots would not have the complexity required to engage with a web server or database. This change also significantly simplified our recycling script, making it roughly 40-60% faster on average.

Section 4: Research Question and Hypothesis

Section 4.1: Research Question and Hypotheses

Our research question is: How do organization type, RAM allocation, and CPU allocation affect attacker engagement? We predict that organizations storing higher proportions of personally identifiable information (PII), having more RAM, or having more CPU will be seen as a larger target than an organization with the opposite. We theorize that being perceived as a larger target will cause attackers to engage more. While Connolly et. al. concluded that organization size had no significant impact on the severity level of ransomware attacks, we are representing organization size differently and are testing for attack engagement rather than the severity level of a ransomware attack.

H_0: Organization type, RAM allocation, and CPU allocation have no impact on attacker engagement.

H_1: Attacker engagement is greater in organizations storing higher proportions of Personally Identifiable Information (PII) (i.e. restaurant < hospital < bank).

H_2: Attacker engagement is greater in organizations with greater RAM allocation.

H_3: Attacker engagement is greater in organizations with greater CPU allocation.

Section 4.2: Term Definitions

We defined an attacker as any unauthorized entity interacting with our honeypots to exploit vulnerabilities, execute malicious actions, or probe for weaknesses. We defined an attack as any action performed by an attacker to exploit vulnerabilities, execute malicious code, or probe for weaknesses. We defined PII as any data that can be used to identify a person such as name, address, medical records, bank account number, etc. We defined attacker engagement as a combination of an attack duration and the number of commands executed during the attack.

Section 5: Experiment Design

Section 5.1: Honeypot Configurations

We have a total of 12 configurations for our honeypots. See Appendix B, Figure 2 for a diagram of our configurations. Each honeypot has an organization type, RAM allocation, and CPU allocation. We have three organization types: restaurant, hospital, and bank. We have two levels of RAM allocation: low (2 GB) and high (4 GB). We have two levels of CPU allocation: low (512 shares) and high (1024 shares).

Section 5.2: Session Management

For session management, our team decided to implement both an idle time and a max total time. These constraints help us maximize data collection while also allowing attackers to carry out their attacks with minimal interference. They also facilitate timely and efficient honeypot recycling.

Section 5.3: Data Integrity

In order to ensure data integrity, we take periodic backups of our honeypot. This includes backups of the MITM logs, health logs, scripts, and honey. The backup process is integrated into our parser code so whenever we fetch log files for data analysis, we also take a backup of our honeypot. The backup is stored on both our local machines and our team's GitHub repository.

Section 5.4: Containerized Honeypots

We deployed our honeypots using LXC containers. We chose containers over virtual machines as they are more lightweight which enables us to recycle our honeypots quicker while also minimizing resource consumption. This allows us to run multiple honeypots concurrently without compromising on performance.

Section 5.5: Network Design

Each container has three associated Network Address Translation (NAT) rules. One rule routes all SSH traffic destined to an IP address to its associated MITM server instead. The other two rules masquerade the internal container IP address with the external IP address. See Appendix B, Figure 3 for a visualization of this. Each container has OpenSSH-Server installed and enabled to allow attackers to SSH

into them. Once an attacker connects, we disallow any other attackers from connecting to the same honeypot. We allow attackers access on their first login attempt. The firewall rules provided by the Division of IT prevent container-to-container movement among other constraints.

Section 5.6: Health Monitoring

To maintain operational stability and for ease of debugging, our team made our internal health logging system that tracks activity on each IP address. This includes events such as an attacker connecting, a MITM server getting created/killed, a container getting created/destroyed, etc. These logs allow us to track when an IP is down, operating slower than expected, and for troubleshooting when something goes wrong. See Appendix B, Figure 4 for an excerpt of one of the log files.

Section 5.7: Log Management

Log files generated by the MITM server are stored in a log folder. The logs are then further subdivided into 12 folders for each of the 12 configurations. The log files are named based on their configuration, timestamp, and IP they were on. This organization format and naming convention ensure that data is easily accessible and well-organized for analysis.

Section 5.8: Honeypot Setup

Our scripts follow a cyclical automation flow for managing our honeypots. Main.sh is called by a cron job on every reboot. It sets up firewall rules and log folders, destroys leftover containers, sets up initial NAT rules, etc. Main then calls our create script for each of our 5 IP addresses. The create script randomly selects a configuration, creates and starts a new container, sets up required NAT rules, starts a MITM server using a forever process, and calls the tail script. The tail script waits for an attacker to connect and then calls the recycle script once the attacker exits, reaches the idle time, or reaches the max total time. The recycle script first calls the destroy script which deletes the old container, deletes the old MITM server, and deletes old NAT rules. The recycle script then calls the create script and the cycle continues. See Appendix B, Figure 1 for a flowchart depicting this.

Section 6: Data Collection

Section 6.1: Collection Method

We collected attack data using the MITM server given to us by ACES. The MITM server generates a log file for each attack. The folder containing all the log files is zipped and downloaded onto our local machine for data processing. We have a data processing script that takes care of downloading logs, taking a backup of our host machine (honeypot scripts, logs, health logs, honey), data preprocessing, data processing, and some data analysis in addition to what is in our Google Colab. See section 7 for more information about the data analysis in our script.

Section 6.2: Collected Data

We gathered extensive metadata from each attack to help find potential trends. Here is a list of all the metadata collected from each attack:

Field	**Description**
Source IP	Attacker IP address
Client ID	SSH client identifier (e.g. SSH-2.0-Go)
Username	Username attacker authenticated with
Password	Password attacker authenticated with
Start Time	Time the attacker connects to the honeypot in EST
End Time	Time of the last command or time the attacker exits honeypot if no commands in EST
Duration	Duration of the attack in milliseconds
Number of Keystrokes	Number of keystrokes from the attacker
List of Commands	List of all the commands entered by the attacker
Execution Mode	Whether the attacker used interactive or non-interactive mode
CPU Usage	The amount of CPU resources used by the honeypot in MiB
Total Block Usage	The amount of storage used by the honeypot in MiB
Memory Usage	The amount of memory used by the honeypot in MiB

Kernel Memory Usage	The amount of kernel memory used by the honeypot in MiB
Outgoing Data	The amount of outgoing data from the honeypot in MiB
Incoming Data	The amount of incoming data to the honeypot in MiB

We had 12,130 attacks as of 4:20 pm EST on November 29th, 2024. The attacks are fairly evenly distributed among the 12 configurations.

Section 6.3: Excluded Data

We excluded a total of 13 attacks. Over the past month, we received 13 attacks from within UMD's network. These attacks originated from the IP: 129.2.19.33. This IP belongs to UMD and has a hostname of "ticketing.lib.umd.edu". All 13 attacks were identical, They had a username of "LIBRSYSISVC" and the attacks consisted of fetching some system information and checking if the user has superuser access. We decided to exclude these attacks as we theorize that these are part of some internal tests run by the DIT and therefore not considered attacks by our definition. See section 4.2 for our definition of an attack.

Section 6.4: Data Preprocessing

Our analysis has three stages: data preprocessing, data processing, and data analysis. All our analysis is done using Python scripts. Our preprocessor has two parts to it. The first part takes a backup of our host machine by zipping the desired files and downloading them onto our local machine. The backup is stored as a zip and the log files are copied and unzipped for parsing. The second part of the preprocessor parses through all the logs and extracts metadata about each attack. The metadata is stored as JSON where each entry represents a single attack. See Appendix B, Figure 5 for an example. The preprocessor also handles any unattached log files or test attack log files. These attacks are ignored and not included in the preprocessed data.

Section 6.5: Data Processing

Our data processor parses the JSON generated by the preprocessor and generates CSV files for our Colab as well as other data formats for our own additional analysis. The processor generates two CSV files, one for all attacks and one for attacks from unique IP addresses. The CSV files include the configuration, number of commands, and duration. It excludes any attacks that reach the timeout time while generating the CSV files.

For our further analysis, the processor also builds a pandas dataframe with 2 fields: an IP address and the number of attacks from that IP address. The processor then uses two publicly available databases, GeoLite2 and Simple Maps World Cities. We use these databases to convert each IP address to a city name and latitude-longitude pair. The processor then uses scikit-learn to cluster attackers. The final output is a data frame that includes the clustered attackers, with each cluster labeled and associated with city names, latitude/longitude pairs, and the number of attacks from that location. This data frame is used for further analysis discussed in section 7.

Section 6.6: Data Categorization

Our initial plan was to categorize the data based on configuration, dividing it into 12 categories—one for each configuration. After inspecting attacker behavior, we realized that none of them interacted with our honey, so we decided to also categorize our data by only RAM and CPU allocation and excluded the organization type. This drops us to 4 configurations. This change had no impact on our processor as we made the necessary changes in our Colab.

Section 7: Data Analysis

Section 7.1: Statistical Strategies

Because our project relied on a wide variety of honeypot configurations, we needed many unique statistical tests to analyze our data accurately. Mainly, we relied on Analysis of Variance (ANOVA), Kruskal-Wallis, and Dunn's Multiple Comparison tests to compare activity across different container configurations. Every test provides unique utilities to our analysis. The ANOVA tests allow us to compare the variance across configurations without the constraint of a t-test which is generally used to compare exactly two data sets. Since we had twelve configurations, the ANOVA allowed us to check whether there was variance across all of these configurations. The Kruskal-Wallis test ranks the data and generates an H-statistic to tell us whether there is any statistical significance across the medians of our honeypots. Similarly, it operates on more than two datasets, which is useful since we are comparing so many configurations. Finally, Dunn's test is useful when we've noted a statistical significance across all configurations. This test breaks down the analysis into pair-by-pair comparisons, analyzing the data between two specific configurations at a time, and generating a table of comparisons between all possible pairs.

We ran the tests including organization type, however, we noted that none of the organization honey was interacted with during the attacks. As a result, we added extra tests where we only compared the configurations by RAM and CPU amounts and ran the same tests with the new constraints. The following two sections will break down the statistical tests into our original (including the organization type) and updated (excluding the organization type) configuration sets. Our violin plots can be found in Appendix B: Figures 13-20.

Section 7.2: Results (Organization/RAM/CPU)

For attacks where we counted repeat attacks from an IP address, we had no statistical significance in our ANOVA and Kruskal-Wallis tests because the p-values for both tests were much higher than our alpha levels of 0.1 (as shown in Appendix B, Figure 6). As a result, we could not reject the null

hypothesis, meaning that we didn't find significant differences between the number of commands or the time spent inside the container across different configurations.

The results for unique IP addresses (i.e. ignoring repeat attacks from an address) were a little different. For the time spent on each configuration, we found no statistical significance across different configurations through our ANOVA and Kruskal-Wallis tests (as shown in Appendix B, Figure 7). As a result, we could not reject the null hypothesis, so we did not find evidence of significant differences between the time spent in the container across different configurations when ignoring repeat attacks from an IP address. However, for the number of commands executed on each container, we had a significant p-value in both ANOVA and Kruskal-Wallis. As a result, we could reject the null hypothesis, so there was a significant difference in the number of commands executed for unique IP addresses. Since we rejected the null hypothesis, we followed up by running Tukey's and Dunn's tests on the pairs of configurations to see if any of the configurations stood out the most in significance. From Appendix B, Figure 8, we don't see any significance between specific pairs of configurations. From Appendix B, Figure 9, we see the low p-values occur between comparisons of configurations including the bank organization honey

At this point, we would have concluded that the bank honey had some statistical significance, however after further analysis of our log files we found that none of the attackers had actually interacted with our honey. As a result, we decided to add extra hypotheses: ones that exclude the organization honey and just see if there is any significance between our RAM and CPU variables alone.

Section 7.3: Results (RAM/CPU only)

Now that we merged organizations and only focused on RAM and CPU, we reran our statistical tests.

For attacks where we counted repeat attacks from an IP address, we had no statistical significance in our ANOVA and Kruskal-Wallis tests because the p-values for both tests were much higher than our alpha levels of 0.1 (as shown in Appendix B, Figure 10). As a result, we could not reject the null

hypothesis, meaning that there was no significant difference between the number of commands or the time spent inside the container across different configurations where we ignored the organization honey.

The results for unique IP addresses (i.e. ignoring repeat attacks from an address) were, again, a little different. For the time spent on each configuration, we found no statistical significance across different configurations through our ANOVA and Kruskal-Wallis tests (as shown in Appendix B, Figure 11). As a result, we could not reject the null hypothesis, so there was no significant difference between the time spent in the container across different configurations when ignoring repeat attacks from an IP address (when ignoring the organization honey). However, for the number of commands executed on each container, we had a significant p-value for both ANOVA and Kruskal-Wallis As a result, we could reject the null hypothesis, so there was a significant difference in the number of commands executed for unique IP addresses when we ignored the organization honey. Since we rejected the null hypothesis, we followed up by running Tukey's and Dunn's tests on the pairs of configurations to see if any RAM/CPU configurations stood out the most in significance. From Appendix B, Figure 11, we observe no significance from Tukey's Test however there is significance for the high-RAM/low-CPU and low-RAM/high-CPU configurations when compared to the high-RAM/high-CPU and low-RAM/low-CPU configurations.

Section 7.4: Supplemental Analysis

We did some additional analysis to try and find other patterns in our data. For example, when filtering our data points by IP address, we realized that just 15 IP addresses accounted for nearly 30% of all attacks. We were also interested in where our attacks were originating from so we created a proportional symbol map (see Appendix B: Figure 10). We mapped the location of each IP address and clustered nearby attacks. We noticed that the vast majority of our attacks were from China, Russia, England, Bulgaria, Germany and Amsterdam. We noticed specifically that the attacks came from towns with a large number of data centers. For example, Slough, an industrial hub just west of London with a high density of data centers, accounted for roughly 7% of attacks. We also noticed that roughly 30% of our attacks came from DigitalOcean, a multinational cloud services provider, data centers.

Since we could not find any significant differences between configuration and duration or number of commands, we also tried to find correlations between duration, number of commands, distance from College Park, and start_time. We ran Pearson correlation tests however all of the correlations were negligible (i.e. $| r | < 0.2$).

Section 8: Conclusion

Section 8.1: Interpretations of the Results

After our statistical analysis and observations made throughout this research project, we noted that we needed to do additional analysis. Since none of the attackers were interacting with our honey, rather, grabbing information about the container's system, we changed our focus to looking at the RAM and CPU amounts. This is because we knew hackers were looking at the container resources, so now our research question was more guided and the statistics would provide some useful insights. From these changes, we were able to reject the null hypothesis, meaning that there is a significant difference in the number of commands entered from a unique attacker is related to the RAM and CPU configuration of the host system.

Section 8.2: Summary Points

Despite our initial struggles with the network configurations and deployment, we were able to collect a sizable sample for our statistical analysis for the culmination of this project. We were able to gather enough data and did a thorough enough analysis which allowed us to do additional analysis for more directed and accurate results, and we were able to derive interesting conclusions as a result. Similarly, we did some supplementary analysis on the location of the attacks, the binaries that were installed on the systems, and other interesting points.

Section 8.3: Further Work and Open Questions

Had we had more time to work on this project, we would have posed more research questions and other qualitative questions, coupled with the fact that we would have had more data to work with. For example, the map that we had of our attack origins was very interesting, and we could have had time to analyze whether the distance affected the time spent in our containers or how many commands they were able to execute (due to the delay or connection stability or any other possible factors). Similarly, some of the attackers downloaded scripts that would link to binaries pulled from random IP addresses on the internet. We were able to get our hands on some of the binaries and did some reverse engineering with

Ghidra, however, we didn't have enough time to trace exactly which backdoors the attackers were exploiting, which we would do if we could continue working on this project.

Appendix A

Lessons Learned

As the project progressed, we realized that we should've been more thorough during the planning phase. While our development phase was organized and timely with clear goals, communication, and consistent progress, the analysis portion of our project lacked the same level of organization. Initially, we had poor communication on who was responsible for what part of the Colab and log parsing code. This led to two group members completing the same task separately or certain tasks being left unfinished. Fortunately, we were able to overcome this setback swiftly however it could have been avoided altogether had we spent more time planning it out earlier in the semester.

Surprises

After analyzing our data, we encountered a few surprises. One major surprise was that nearly 70% of attacks ran no commands. Another notable surprise was that nearly 20% of the attacks—amounting to two-thirds of the attacks with commands—executed only the uname command. Our third and final surprise was that none of the attacks interacted with our honey. The few attacks that ran a command other than uname, if at all, tried to modify the .ssh directory, download malware, or an assortment of other uninteresting behavior.

Pitfalls and Failures

We faced several setbacks throughout the development process. Initially, the IP addresses we were assigned were faulty but we were able to get new IPs assigned after meeting with TAs during office hours. Of the newly assigned IPs, only one worked but we were able to deploy on that IP. Soon after we were assigned a second working IP, our destroy script faced issues with having more than one IP and we had to have our IPs undeployed as a result of this. We were able to fix this issue quickly and get our IPs redeployed again. Since then, both of our IPs have been working smoothly with no issues.

An inefficiency was our container naming convention. We named our containers based on their configuration, epoch time of creation, and which IP they were on. This was redundant and led to lengthy

container names and log file names. Naming them based on which IP they were on (69 or 106) and a timestamp of time of creation in ISO 8601 would've led to much shorter container names and log file names while also being significantly easier to understand.

Another inefficiency was the way we organized our log files. We had one folder for each configuration. This made it difficult to find specific log files. Adding subfolders in each configuration folder for the day the log file was created would've greatly improved organization.

Feedback

One piece of feedback our group has is that the standups felt a bit repetitive. It felt like every group was saying the same thing every week. It also would've been helpful if the first Technical Logs and Health Logs assignments were graded so we know what the expectations are for assignments two and three. Apart from that, the course was well structured. Having all the assignments and their deadlines posted at the beginning of the semester was extremely helpful for time and project management.

Appendix B

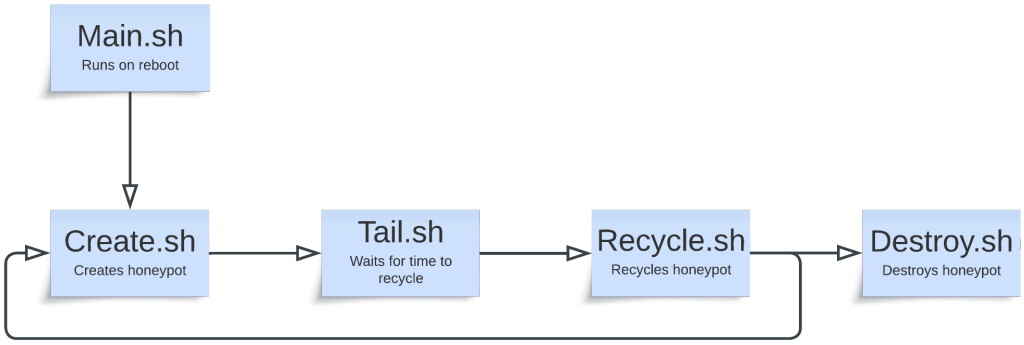

Figure 1: Flow of Scripts

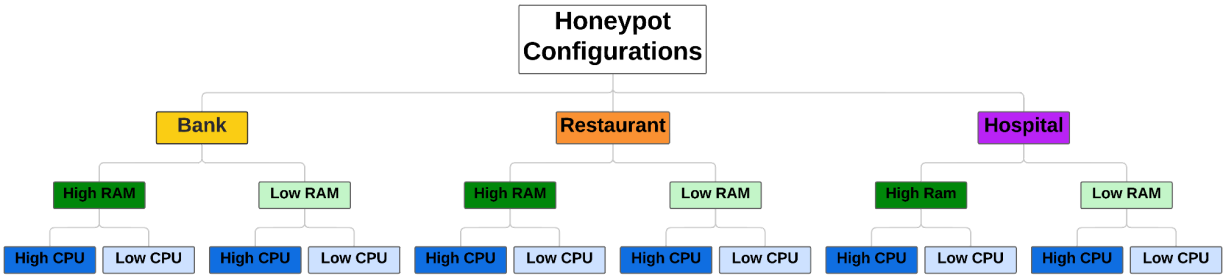

Figure 2: Honeypot Configurations

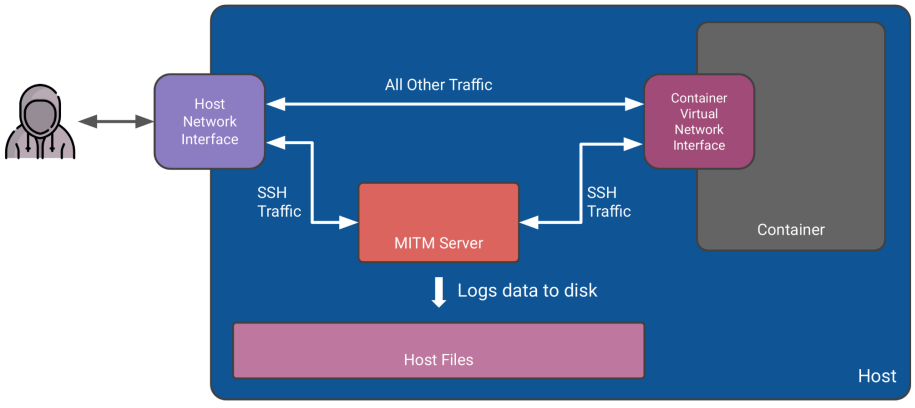

Figure 3: Honeypot MITM Setup

```
2024-11-24 23:41:23: Waiting for attacker
2024-11-24 23:47:14: Attacker connected, monitoring log file
2024-11-24 23:47:22: Attacker exited
2024-11-24 23:47:22: Running recycle script
2024-11-24 23:47:22: Running destroy script
2024-11-24 23:47:27: Killing MITM
2024-11-24 23:47:27: Removing NAT Rules
2024-11-24 23:47:29: Destroyed Container
2024-11-24 23:47:29: Running create script
2024-11-24 23:47:29: Creating container
2024-11-24 23:47:29: Waiting for container to be created
2024-11-24 23:47:43: Created Container
2024-11-24 23:47:43: Waiting for container to start
2024-11-24 23:47:47: Container started
2024-11-24 23:47:47: Installing Binaries
2024-11-24 23:48:10: Openssh is running
2024-11-24 23:48:10: Starting MITM
2024-11-24 23:48:12: MITM is running
2024-11-24 23:48:12: Running tail script
2024-11-24 23:48:12: Waiting for attacker
```

Figure 4: Excerpt of Internal Health Logs


```
  "Attack-ID 10378": {
      "Configuration": "restaurant_low_high",
      "Source_IP": "164.90.215.117",
      "Destination_IP": "128.8.238.106",
      "Client_ID": "SSH-2.0-Go",
      "Attacker_Username": "es",
      "Attacker_Password": "123456",
      "Start_Time": "2024-11-20 18:57:25.865000",
      "End_Time": "2024-11-20 18:57:27.609000",
      "Duration": 1.744,
      "Keystrokes_Count": 0,
      "Commands": [
          "uname -s -v -n -r -m"
      ],
```

```
    "Noninteractive_Mode": true,
    "CPU_Usage": "26.81 seconds",
    "Total_BlkIO_Use": "142.95 MiB",
    "Memory_Usage": "318.06 MiB",
    "Kernel_Memory_Usage": "24.54 MiB",
    "Outgoing_Data": "135.91 KiB",
    "Incoming_Data": "19.76 MiB"
}
```


Figure 5: Example Entry in Preprocessed Data

```
###### Duration_Seconds ######
### ANOVA ###
                                 sum_sq       df         F    PR(>F)
C(Organization)                1.645725e+03      2.0  1.361058  0.256429
C(RAM)                         5.353543e-01      1.0  0.000886  0.976261
C(CPU)                         1.983352e+02      1.0  0.328057  0.566816
C(Organization):C(RAM)         1.248625e+03      2.0  1.032647  0.356095
C(Organization):C(CPU)         4.211696e+02      2.0  0.348319  0.705881
C(RAM):C(CPU)                  4.972841e+01      1.0  0.082253  0.774271
C(Organization):C(RAM):C(CPU)  2.476134e+03      2.0  2.047829  0.129060
Residual                       7.224071e+06  11949.0       NaN       NaN

### Kruskal Wallis ###
H-statistic: 9.3208571155332, P-value: 0.5923


###### Num_Commands ######
### ANOVA ###
                                 sum_sq       df         F    PR(>F)
C(Organization)                0.740135      2.0  0.445266  0.640664
C(RAM)                         0.075756      1.0  0.091150  0.762725
C(CPU)                         0.115909      1.0  0.139462  0.708823
C(Organization):C(RAM)         0.924346      2.0  0.556088  0.573463
C(Organization):C(CPU)         0.238618      2.0  0.143553  0.866276
C(RAM):C(CPU)                  0.216651      1.0  0.260675  0.609666
C(Organization):C(RAM):C(CPU)  0.445610      2.0  0.268079  0.764852
Residual                       9930.993633  11949.0       NaN       NaN

### Kruskal Wallis ###
H-statistic: 6.14618959282061, P-value: 0.8634
```

Figure 6: Statistical Tests for All Attacks with Organization

```
###### Duration_Seconds ######
### ANOVA ###
                                 sum_sq      df         F    PR(>F)
C(Organization)              4.298147e+01     2.0  0.051921  0.949405
C(RAM)                       8.713313e+02     1.0  2.105111  0.146899
C(CPU)                       4.619848e+01     1.0  0.111614  0.738334
C(Organization):C(RAM)       9.702643e+02     2.0  1.172065  0.309850
C(Organization):C(CPU)       1.033454e+03     2.0  1.248397  0.287094
C(RAM):C(CPU)                4.200133e+01     1.0  0.101474  0.750087
C(Organization):C(RAM):C(CPU)  1.160973e+02     2.0  0.140244  0.869151
Residual                     1.423858e+06  3440.0       NaN       NaN

### Kruskal Wallis ###
H-statistic: 9.00106991746474, P-value: 0.6218


###### Num_Commands ######
### ANOVA ###
                                 sum_sq      df         F    PR(>F)
C(Organization)                0.383497     2.0  0.284386  0.752493
C(RAM)                         1.149085     1.0  1.704236  0.191822
C(CPU)                         0.157376     1.0  0.233408  0.629039
C(Organization):C(RAM)         0.469475     2.0  0.348145  0.706022
C(Organization):C(CPU)         0.669467     2.0  0.496451  0.608730
C(RAM):C(CPU)                  2.074275     1.0  3.076407  0.079525
C(Organization):C(RAM):C(CPU)  5.035650     2.0  3.734246  0.023988
Residual                    2319.428728  3440.0       NaN       NaN
```

Figure 7: Statistical Tests for Unique IP Attacks with Organization

```
### Tukey ###
             Multiple Comparison of Means - Tukey HSD, FWER=0.10
=================================================================================
       group1              group2        meandiff p-adj   lower  upper  reject
---------------------------------------------------------------------------------
      bank_high_high        bank_high_low  -0.1194 0.5406 -0.2829 0.0442  False
      bank_high_high        bank_low_high  -0.1081 0.7185 -0.2749 0.0588  False
      bank_high_high         bank_low_low   0.036    1.0 -0.1388 0.2109  False
      bank_high_high   hospital_high_high  -0.1206 0.6992 -0.3041 0.0629  False
      bank_high_high    hospital_high_low  -0.0498 0.9996 -0.2309 0.1313  False
      bank_high_high    hospital_low_high  -0.0451 0.9999 -0.2323 0.1422  False
      bank_high_high     hospital_low_low  -0.0026    1.0 -0.1945 0.1894  False
      bank_high_high restaurant_high_high  -0.0709 0.9973 -0.2817 0.1399  False
      bank_high_high  restaurant_high_low  -0.0425    1.0 -0.2441 0.1591  False
      bank_high_high  restaurant_low_high   0.0262    1.0  -0.189 0.2415  False
      bank_high_high   restaurant_low_low  -0.0135    1.0 -0.2288 0.2018  False
       bank_high_low        bank_low_high   0.0113    1.0 -0.1684  0.191  False
       bank_high_low         bank_low_low   0.1554 0.3294 -0.0318 0.3426  False
       bank_high_low   hospital_high_high  -0.0012    1.0 -0.1965 0.1941  False
       bank_high_low    hospital_high_low   0.0696  0.995 -0.1235 0.2626  False
       bank_high_low    hospital_low_high   0.0743 0.9932 -0.1245 0.2731  False
       bank_high_low     hospital_low_low   0.1168 0.8486 -0.0865 0.3201  False
       bank_high_low restaurant_high_high   0.0485    1.0 -0.1727 0.2696  False
       bank_high_low  restaurant_high_low   0.0769 0.9948 -0.1355 0.2893  False
       bank_high_low  restaurant_low_high   0.1456 0.7219 -0.0798  0.371  False
       bank_high_low   restaurant_low_low   0.1058 0.9591 -0.1196 0.3312  False
       bank_low_high         bank_low_low   0.1441 0.4778  -0.046 0.3342  False
       bank_low_high   hospital_high_high  -0.0125    1.0 -0.2106 0.1856  False
       bank_low_high    hospital_high_low   0.0583 0.9991 -0.1376 0.2541  False
       bank_low_high    hospital_low_high    0.063 0.9986 -0.1385 0.2646  False
       bank_low_high     hospital_low_low   0.1055 0.9254 -0.1004 0.3114  False
       bank_low_high restaurant_high_high   0.0372    1.0 -0.1865 0.2608  False
       bank_low_high  restaurant_high_low   0.0656 0.9989 -0.1494 0.2805  False
       bank_low_high  restaurant_low_high   0.1343 0.8253 -0.0935 0.3621  False
       bank_low_high   restaurant_low_low   0.0945  0.984 -0.1333 0.3224  False
        bank_low_low   hospital_high_high  -0.1566 0.4638 -0.3614 0.0482  False
        bank_low_low    hospital_high_low  -0.0858 0.9812 -0.2885 0.1169  False
        bank_low_low    hospital_low_high  -0.0811 0.9904 -0.2893 0.1271  False
        bank_low_low     hospital_low_low  -0.0386    1.0  -0.251 0.1738  False
        bank_low_low restaurant_high_high  -0.1069 0.9615 -0.3366 0.1227  False
        bank_low_low  restaurant_high_low  -0.0785 0.9956 -0.2997 0.1427  False
        bank_low_low  restaurant_low_high  -0.0098    1.0 -0.2435 0.2239  False
        bank_low_low   restaurant_low_low  -0.0496    1.0 -0.2833 0.1842  False
  hospital_high_high    hospital_high_low   0.0708 0.9973 -0.1395  0.281  False
  hospital_high_high    hospital_low_high   0.0755 0.9961   -0.14  0.291  False
  hospital_high_high     hospital_low_low    0.118 0.8987 -0.1016 0.3376  False
  hospital_high_high restaurant_high_high   0.0497    1.0 -0.1866  0.286  False
  hospital_high_high  restaurant_high_low   0.0781 0.9968   -0.15 0.3062  False
  hospital_high_high  restaurant_low_high   0.1468 0.7886 -0.0935 0.3871  False
  hospital_high_high   restaurant_low_low    0.107 0.9723 -0.1332 0.3473  False
   hospital_high_low    hospital_low_high   0.0048    1.0 -0.2088 0.2183  False
   hospital_high_low     hospital_low_low   0.0472    1.0 -0.1704 0.2649  False
   hospital_high_low restaurant_high_high  -0.0211    1.0 -0.2556 0.2134  False
   hospital_high_low  restaurant_high_low   0.0073    1.0 -0.2189 0.2335  False
   hospital_high_low  restaurant_low_high    0.076 0.9983 -0.1624 0.3145  False
   hospital_high_low   restaurant_low_low   0.0363    1.0 -0.2022 0.2747  False
   hospital_low_high    hospital_low_low   0.0425    1.0 -0.1803 0.2653  False
   hospital_low_high restaurant_high_high  -0.0258    1.0 -0.2651 0.2134  False
   hospital_low_high  restaurant_high_low   0.0026    1.0 -0.2286 0.2337  False
   hospital_low_high  restaurant_low_high   0.0713 0.9992 -0.1719 0.3145  False
   hospital_low_high   restaurant_low_low   0.0315    1.0 -0.2116 0.2747  False
    hospital_low_low restaurant_high_high  -0.0683 0.9995 -0.3113 0.1746  False
    hospital_low_low  restaurant_high_low  -0.0399    1.0 -0.2749 0.1951  False
    hospital_low_low  restaurant_low_high   0.0288    1.0  -0.218 0.2756  False
    hospital_low_low   restaurant_low_low   -0.011    1.0 -0.2578 0.2358  False
restaurant_high_high  restaurant_high_low   0.0284    1.0 -0.2222  0.279  False
restaurant_high_high  restaurant_low_high   0.0971 0.9936 -0.1646 0.3589  False
restaurant_high_high   restaurant_low_low   0.0574    1.0 -0.2044 0.3191  False
 restaurant_high_low  restaurant_low_high   0.0687 0.9996 -0.1857 0.3231  False
 restaurant_high_low   restaurant_low_low    0.029    1.0 -0.2254 0.2833  False
 restaurant_low_high   restaurant_low_low  -0.0398    1.0 -0.3051 0.2256  False
---------------------------------------------------------------------------------
```

Figure 8: Tukey Test for Unique IP Attacks with Organization

```
### Kruskal Wallis ###
H-statistic: 26.111380149659905, P-value: 0.0062

### Dunn's ###
                      bank_high_high  bank_high_low  bank_low_high  bank_low_low  hospital_high_high  hospital_high_low
bank_high_high              1.000000       0.000095       0.000592      0.334799            0.000279           0.042736
bank_high_low               0.000095       1.000000       0.716856      0.012077            0.884140           0.159837
bank_low_high               0.000592       0.716856       1.000000      0.033285            0.636365           0.292509
bank_low_low                0.334799       0.012077       0.033285      1.000000            0.015000           0.327700
hospital_high_high          0.000279       0.884140       0.636365      0.015000            1.000000           0.153812
hospital_high_low           0.042736       0.159837       0.292509      0.327700            0.153812           1.000000
hospital_low_high           0.020150       0.306583       0.493256      0.200562            0.282290           0.749807
hospital_low_low            0.108862       0.103862       0.197479      0.512416            0.102103           0.785538
restaurant_high_high        0.037044       0.368943       0.550447      0.237884            0.336381           0.756680
restaurant_high_low         0.077408       0.183729       0.312364      0.396851            0.173035           0.961273
restaurant_low_high         0.048112       0.344927       0.516760      0.271772            0.315152           0.806319
restaurant_low_low          0.174830       0.124495       0.217284      0.597270            0.118864           0.753426
                      hospital_low_high  hospital_low_low  restaurant_high_high  restaurant_high_low  restaurant_low_high  restaurant_low_low
bank_high_high                 0.020150          0.108862              0.037044             0.077408             0.048112            0.174830
bank_high_low                  0.306583          0.103862              0.368943             0.183729             0.344927            0.124495
bank_low_high                  0.493256          0.197479              0.550447             0.312364             0.516760            0.217284
bank_low_low                   0.200562          0.512416              0.237884             0.396851             0.271772            0.597270
hospital_high_high             0.282290          0.102103              0.336381             0.173035             0.315152            0.118864
hospital_high_low              0.749807          0.785538              0.756680             0.961273             0.806319            0.753426
hospital_low_high              1.000000          0.567681              0.984788             0.732285             0.968443            0.556480
hospital_low_low               0.567681          1.000000              0.587251             0.837333             0.633458            0.949331
restaurant_high_high           0.984788          0.587251              1.000000             0.738616             0.956794            0.572940
restaurant_high_low            0.732285          0.837333              0.738616             1.000000             0.784836            0.801581
restaurant_low_high            0.968443          0.633458              0.956794             0.784836             1.000000            0.615207
restaurant_low_low             0.556480          0.949331              0.572940             0.801581             0.615207            1.000000
```

Figure 9: Kruskal Wallis and Dunn's Test for Unique Attacks with Organization

```
######### All Merged #########
###### Duration_Seconds ######
### ANOVA ###
                     sum_sq       df         F    PR(>F)
C(RAM)         1.299752e-01      1.0  0.000215  0.988303
C(CPU)         2.293294e+02      1.0  0.379274  0.538004
C(RAM):C(CPU)  4.828024e+01      1.0  0.079848  0.777509
Residual       7.229849e+06  11957.0       NaN       NaN

### Kruskal Wallis ###
H-statistic: 2.083798272486556, P-value: 0.5552


###### Num_Commands ######
### ANOVA ###
                     sum_sq       df         F    PR(>F)
C(RAM)             0.072623      1.0  0.087419  0.767490
C(CPU)             0.115416      1.0  0.138929  0.709355
C(RAM):C(CPU)      0.203519      1.0  0.244981  0.620640
Residual        9933.331455  11957.0       NaN       NaN

### Kruskal Wallis ###
H-statistic: 3.552770929528399, P-value: 0.3140
```

Figure 10: Statistical Tests for All Attacks without Organization

```
###### Duration_Seconds ######
### ANOVA ###
                    sum_sq      df         F    PR(>F)
C(RAM)         9.362832e+02     1.0  2.263854  0.132516
C(CPU)         4.276454e+01     1.0  0.103401  0.747805
C(RAM):C(CPU)  6.387877e+01     1.0  0.154453  0.694340
Residual       1.426022e+06  3448.0       NaN       NaN

### Kruskal Wallis ###
H-statistic: 0.6721548968677761, P-value: 0.8797


###### Num_Commands ######
### ANOVA ###
                   sum_sq      df         F    PR(>F)
C(RAM)           1.106570     1.0  1.640354  0.200363
C(CPU)           0.141793     1.0  0.210191  0.646647
C(RAM):C(CPU)    2.276776     1.0  3.375043  0.066277
Residual       2325.992822  3448.0       NaN       NaN

### Tukey ###
  Multiple Comparison of Means - Tukey HSD, FWER=0.10
=====================================================
  group1   group2  meandiff p-adj   lower  upper  reject
-----------------------------------------------------
high_high high_low  -0.0334 0.8113 -0.1198 0.0529  False
high_high low_high  -0.0129 0.9872 -0.1017 0.0759  False
high_high  low_low   0.0571 0.4763  -0.034 0.1482  False
 high_low low_high   0.0205 0.9557  -0.071  0.112  False
 high_low  low_low   0.0906 0.1195 -0.0032 0.1843  False
 low_high  low_low     0.07 0.3385  -0.026  0.166  False
-----------------------------------------------------

### Kruskal Wallis ###
H-statistic: 7.846562126143783, P-value: 0.0493

### Dunn's ###
           high_high  high_low  low_high   low_low
high_high   1.000000  0.049373  0.044527  0.887751
high_low    0.049373  1.000000  0.924292  0.051462
low_high    0.044527  0.924292  1.000000  0.046333
low_low     0.887751  0.051462  0.046333  1.000000
```

Figure 11: Statistical Test for Unique IP Attacks without Organization

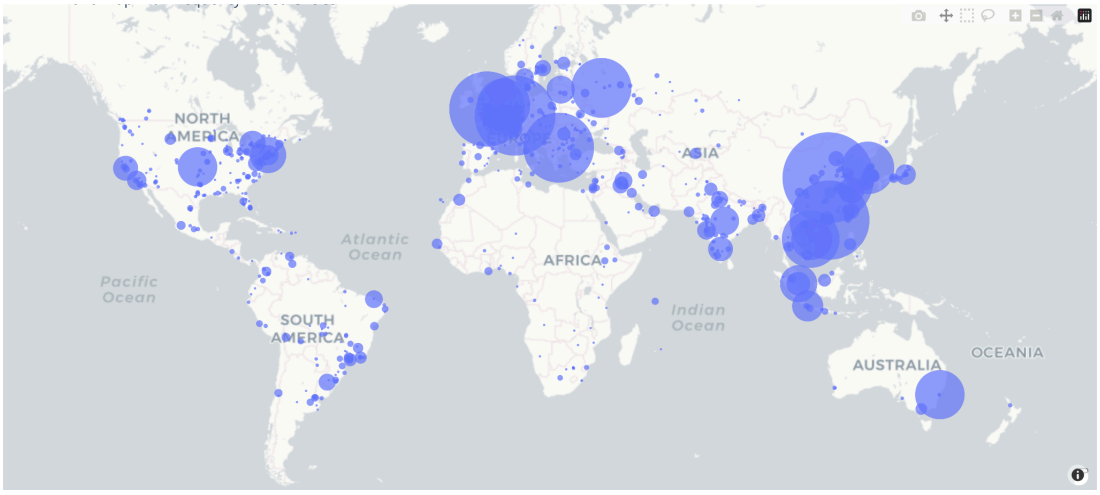

Figure 12: Proportional Symbol Map for Attack Origins

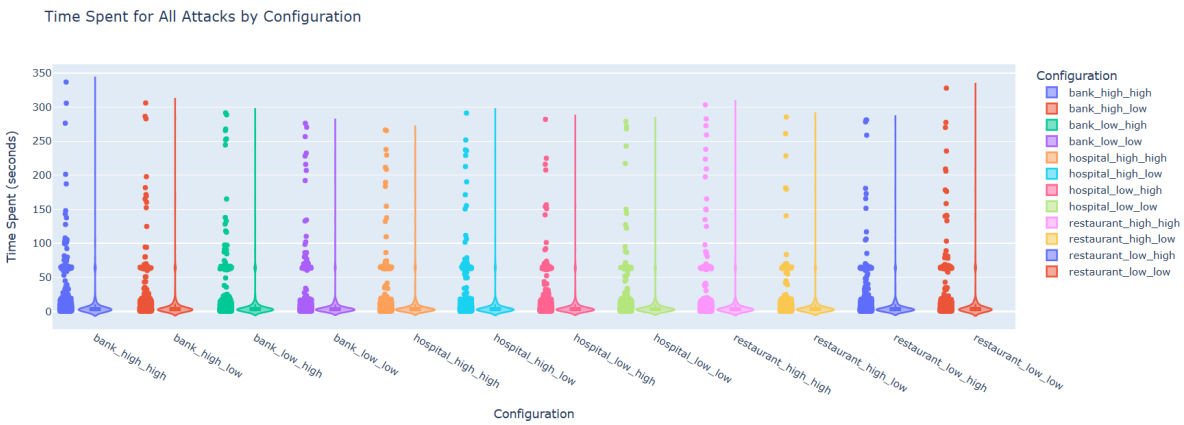

Figure 13: Time Spent for All Attacks by Configuration

Figure 14: Time Spent for Attacks from Unique IPs by Configuration

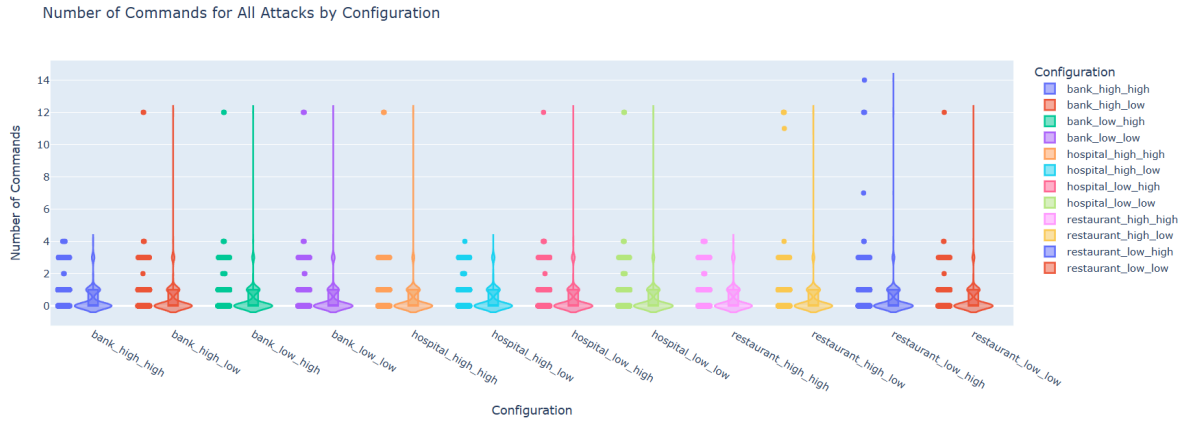

Figure 15: Number of Commands for All Attacks by Configuration

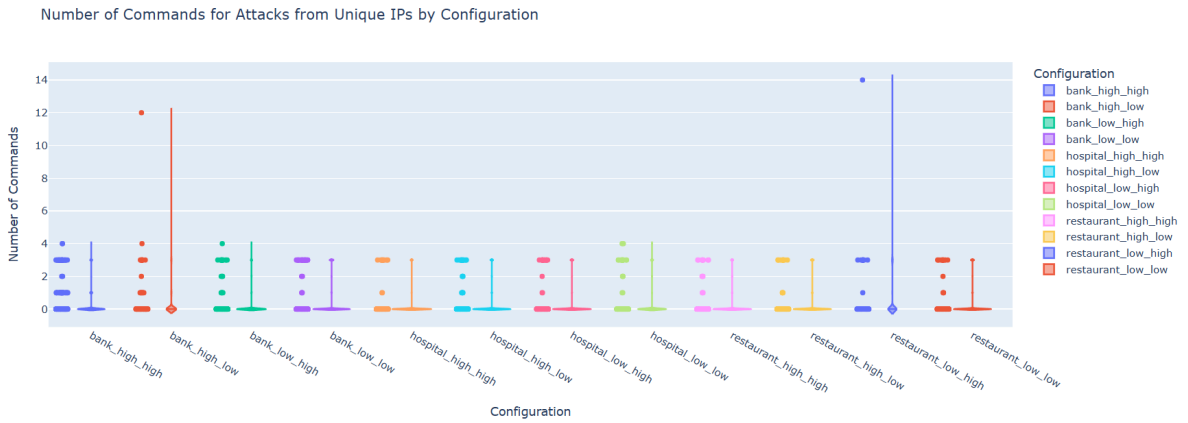

Figure 16: Number of Commands for Attacks from Unique IPs by Configuration

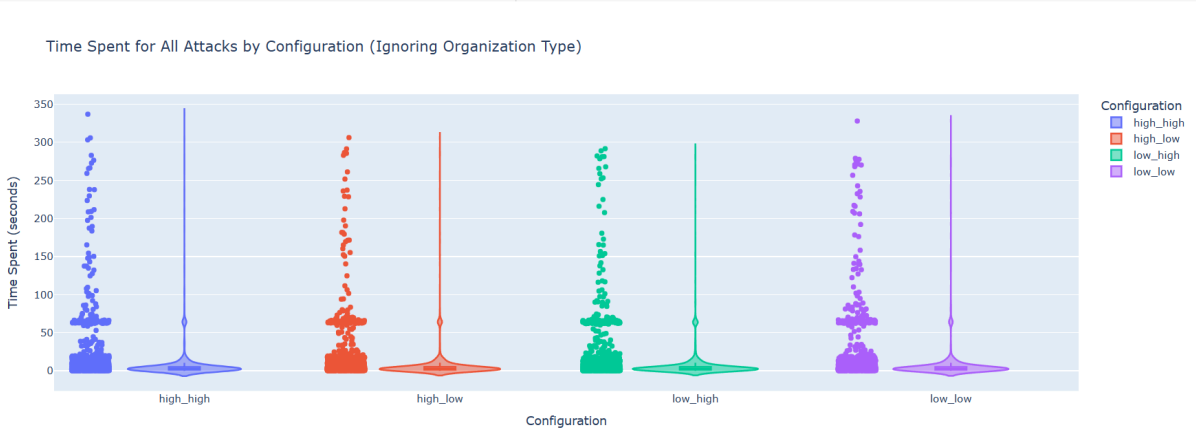

Figure 17: Time Spent for All Attacks by Configuration (Ignoring Organization Type)

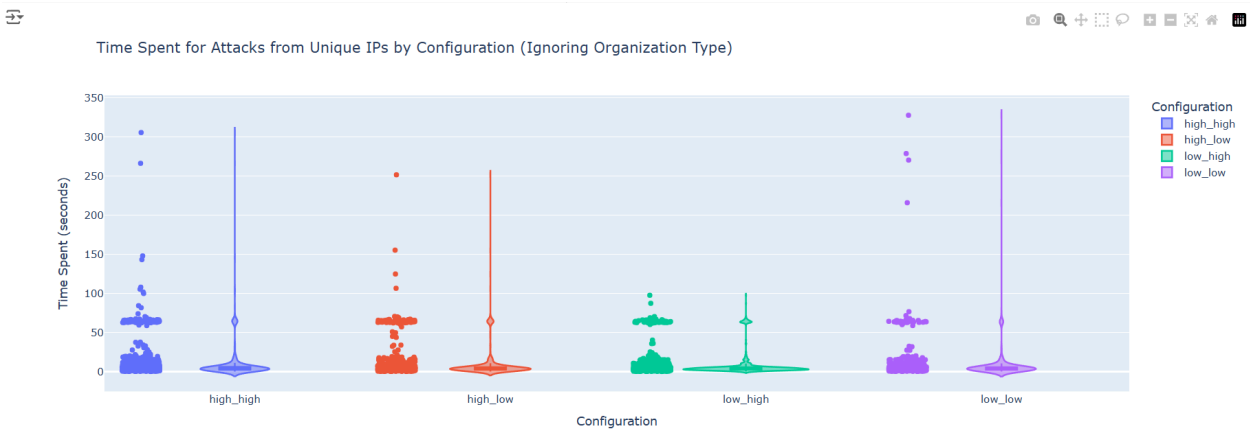

Figure 18: Time Spent for Attacks from Unique IPs by Configuration (Ignoring Organization Type)

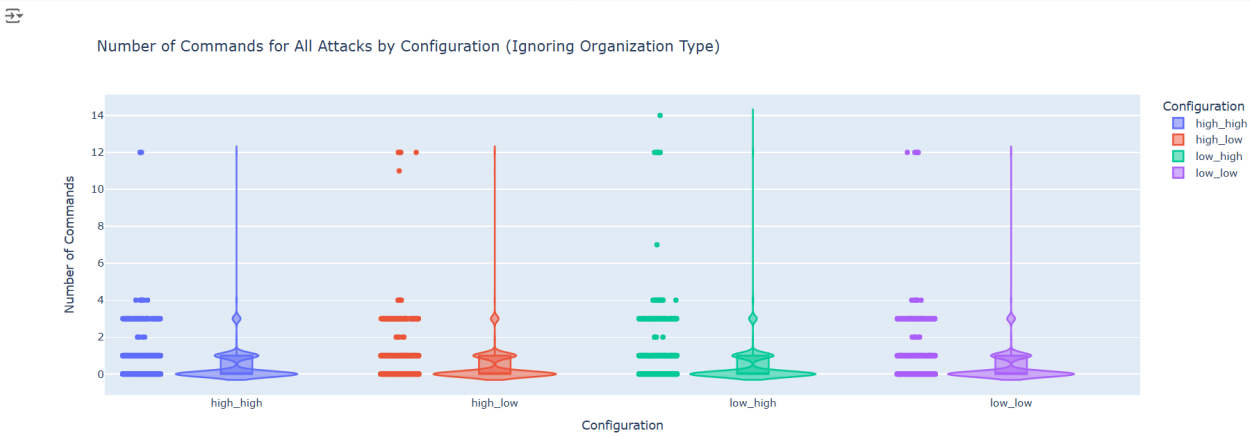

Figure 19: Number of Commands for All Attacks by Configuration (Ignoring Organization Type)

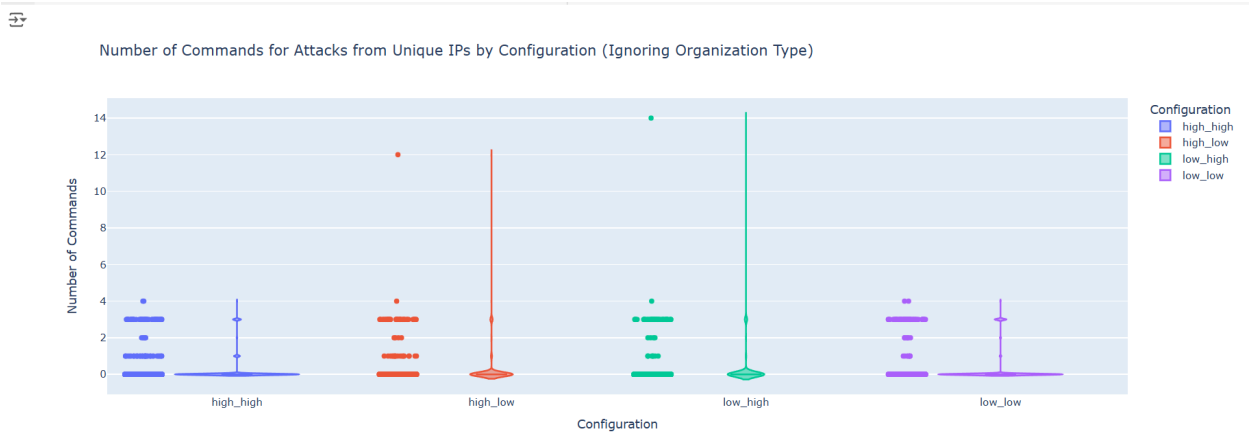

Figure 20: Number of Commands for Attacks from Unique IPs by Configuration (Ignoring Organization Type)

Works Cited

Cherdantseva, et al. “A Review of Cyber Security Risk Assessment Methods for SCADA Systems” Department of International Politics, Aberystwyth University, UK, 30 Oct.2015 https://www.researchgate.net/publication/283568912_A_Review_of_cyber_security_risk_assessment_methods_for_SCADA_systems

National Security Agency, “CSI: Detect and Prevent Web Shell Malware” National Security Agency, 22 Apr. 2020 https://media.defense.gov/2020/Jun/09/2002313081/-1/-1/0/CSI-DETECT-AND-PREVENT-WEB-SHELL-MALWARE-20200422.PDF.

Javadpour, Amir, et al. "A Comprehensive Survey on Cyber Deception Techniques to Improve Honeypot Performance." *Computers & Security*, vol. 140, May 2024, article no. 103792, https://www.sciencedirect.com/science/article/pii/S0167404824000932

Verizon Communications, Inc. “Verizon Data Breach Investigations Report.” Verizon, 2022 https://www.verizon.com/business/en-gb/resources/2022-data-breach-investigations-report-dbir.pdf