Scuola universitaria professionale
della Svizzera italiana

SUPSI

University of Applied Sciences and Arts of Southern Switzerland
Department of Innovative Technologies

Data Security and Blockchain

HONEYPOT CYBER ATTACKS DATA ANALYSIS

Carlo Grigioni
carlo.grigioni@student.supsi.ch

Christian Pala
christian.pala@student.supsi.ch

Andrea Wey
andrea.wey@student.supsi.ch

Professor: Angelo Consoli
SUPSI, Lugano Switzerland

18/05/2023

Table of Contents

List of Tables

List of Figures

1. Introduction

In the evolving landscape of cyber security, the use of machine learning in securing networks and systems from malicious activities is an emerging field [1].

This report presents an analysis performed on a dataset of cyber attack attempts, collected from a honeypot, provided by our Data Security and Blockchain Course Professor, Angelo Consoli [2]. A honeypot is a decoy system intended to attract and trap individuals attempting to gain unauthorized access to a computer system.

The dataset comprises ten files, each containing Unix-timestamped commands executed by potential intruders, for a total of 11957 observations, collected between the start of February and the middle of March 2021.

Our aim is to understand the common patterns of behavior in the commands executed by the intruders, and if possible, to categorize the intrusions based on the techniques used.

We approach this task by performing Natural Language Processing (NLP), in particular, pattern recognition and extraction on the command data using regular expressions on the command data to extract salient features. We then apply two separate clustering pipelines to group similar commands together.

This report is structured as follows:

- *Data*: An overview of the dataset and an explanation of its significance.
- *Feature Extraction*: A detailed account of the preprocessing steps and techniques used to extract meaningful features from the command data.
- *Clustering*: A description of the clustering methodology used to group similar command sequences together.
- *Results*: A presentation and analysis of the results obtained from the information extraction and clustering procedures.
- *Limitations and Conclusions*: A discussion on the limitations of the current approach, suggestions for future work, and final thoughts.

We hope that the findings of this study will provide useful insights to improve the effectiveness of honeypot systems.

2. Data

The raw data for this analysis were obtained from 10 comma-separated values (CSV) files each containing Unix timestamps, and shell commands executed by potential intruders in a honeypot.

Each line in the CSV files is structured as follows: ”timestamp”, and ”command”, where the timestamp is a Unix timestamp in milliseconds and the command is a string of the command executed by the potential intruder.

In order to process the data into a more manageable and analyzable format, we implemented a Python script to parse the CSV files and consolidate them into a single data frame. The parsing process involved the following steps:

- Use a regular expression to match and extract the timestamp and command from each line in the CSV files. This was necessary due to the rich syntax of shell commands, making it hard to parse correctly without a dedicated method.
- Convert the Unix timestamp (in milliseconds) to a DateTime object, for human convenience.
- Merge and store all files in a single data frame, sorted by date.

We also included logging functionalities to ensure we were parsing all rows correctly and to capture other possible sources of errors.

3. Feature Extraction

The feature and information extraction phase was the most important part of the project. We operated on two levels. We identified patterns of behavior highlighted by the commands executed, for instance, information gathering, or privilege escalation; we also looked for relevant data embedded in the commands, Internet Protocol (IP) addresses, and Uniform Resource Locators (URLs) as primary examples. Table 1 presents the features we extracted from the shell commands.

Behavioral Patterns	**Data**
Admin commands	
Execution attempts	Bash scripts
File information gathering	Directories
Network communication	URLs
Network information gathering	IP addresses, Port numbers
Privilege escalation	Passwords and Secure Socket Shell (SSH) keys
Running processes information gathering	
System information gathering	
User information gathering	

Table 1: Features extracted from the command data, sorted alphabetically.

The specific details on the pattern matching we used to extract the relevant information, are publicly available on our GitHub repository [3] for this project. The main strategy for the behavioral features was to search for commands associated with said activity, for instance, looking for **uname** as an example of system information gathering.

3.1 Post-processing

After the extraction, the results were cleaned to remove uninformative elements. We encoded the data information we extracted for the machine learning part of the project, by adding their count for each command as a feature. We ensured all extractions were collecting useful information, before continuing with the clustering phase.

4. Clustering

The first clustering strategy we employed to group attackers, involved using the Uniform Manifold Approximation and Projection (UMAP) [4] technique for dimensionality reduction on the features we engineered, followed by Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [5] to group the reduced features.

4.1 Dimensionality Reduction with UMAP

Given the high dimensionality of our feature space, we first performed dimensionality reduction using the UMAP algorithm. This technique preserves the local and global structure of the data, which is essential for maintaining the nuances of the command patterns in lower-dimensional space.

4.2 Clustering with HDBSCAN

Once the dimensionality of our dataset was reduced, we used the HDBSCAN algorithm to cluster the data points. HDBSCAN is a density-based clustering method, which is well-suited to our task as it does not require the pre-specification of the number of clusters and can discover clusters of varying shapes and sizes. It also classifies some points as noise, which is useful for identifying anomalous commands that do not fit into any common patterns.

4.3 Alternative clustering approach

For the second clustering strategy, we sought to directly analyze the command strings themselves. This involved a more complex pre-processing phase but allowed us to avoid the manual feature engineering step used in the first approach.

4.3.1 Preprocessing

Our preprocessor replaces specific patterns in the command strings with generalized tokens. For instance, all IP addresses are replaced with the token "IPADDRESS", URLs with "URL", and so forth. This was designed to help identify the structure and function of the commands, rather than focusing on the specific data they contain.

4.3.2 Tokenization

We used the WordPunctTokenizer from the Natural Language Toolkit (NLTK) [6] to tokenize the preprocessed command strings. This tokenizer splits the strings into words and punctuation, which allows us to analyze the syntactical structure of the commands.

4.3.3 Dimensionality reduction and DBSCAN Clustering

After tokenization, the commands were transformed into a numerical feature space using the HashingVectorizer from the sci-kit-learn library. This creates a fixed-size vector for each command, with each element representing the frequency of a certain token in the command. We then used the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm to cluster the commands. DBSCAN is another density-based clustering method that, like HDBSCAN, does not require the pre-specification of the number of clusters and can identify the noise. The choice of DBSCAN in this scenario allowed us to deal with the high dimensionality of the data and the presence of outliers.

4.3.4 Visualizing the clusters

To visualize the clusters, we first performed dimensionality reduction on the feature vectors using Truncated Singular Value Decomposition (SVD). This is a matrix factorization technique that can reduce the dimensionality of the data while preserving its structure, similar to Principal Component Analysis (PCA), but more memory efficient. To compare the two clustering methods, we later decided to just re-use UMAP for this part of the work.

4.4 Comparison

In summary, this second approach provides a complementary perspective on the command data, focusing on the structure and syntax of the commands rather than specific patterns of behavior or data elements. Combining the insights from both clustering strategies can give a more comprehensive understanding of the attacks and their patterns.

5. Results

In this section, we discuss the results of our analysis, in particular, we highlight the information and data we extracted from the shell commands and present the results of our clustering procedures.

5.1 Attack Information

We categorized the command categories by extending the same pattern-matching techniques we developed for feature engineering. Figure 1 shows that most of the commands with malicious code aim at collecting and processing information, with the important exception of privilege escalation which is almost always a necessary step for a successful attack. This common pattern across attacks suggests a robust filter design could help machine learning models isolate attack categories better.

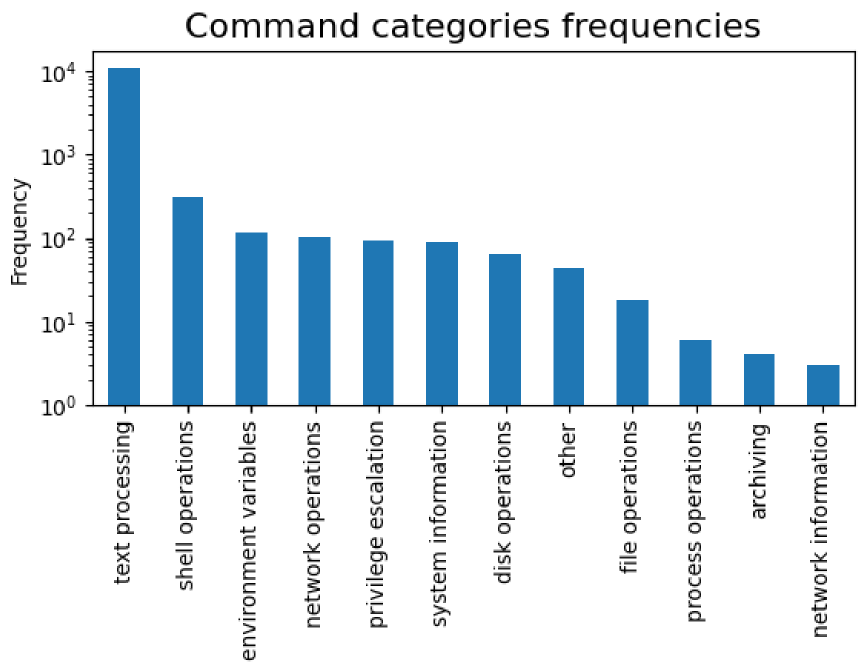

Figure 1: Command Categories

5.1.1 Bash files

Interestingly we found 100 instances of attempts to download the script ISIS.sh from 104.208.155.37, which according to opentracker is a Microsoft Azure cloud service hosted at their Boydton data center in Virginia, as well as 59 attempts to download xxxbins.sh from an IP belonging to another hosting and cloud provider in Seattle, Hostwinds Llc. Extrapolating from the data, it seems likely that hosting scripts on the cloud, ready for download on victims' machines, is a common strategy, at least during the data collection period for this dataset, in 2021. Another interesting point is that attackers often use both the wget and curl commands simultaneously when downloading scripts from external machines, the redundancy is likely to increase the success rate but also makes the pattern more detectable.

5.1.2 Directories

By far the most common directory requested by the attackers on the Unix honeypot was /proc/cpuinfo, with more than 90% of attackers requesting information on the CPU usage, another indicator that most of the command code is gathering information on the victim, regardless of the attack method. Other notable directories include home, root, and tmp, the latter being often used to store downloaded malicious code, as well as the mount folder, finally var/run to gather information on running programs is also very common.

5.1.3 IP addresses and ports

All the IPs we detected are from cloud providers. 104.208.155.37 and 142.11.216.5 are by far the most active IP addresses we captured in the dataset, they are used to download scripts on the honeypot, and both belong to cloud providers in the US, Microsoft Azure, and Hostwinds Llc respectively. We also found the same pattern in Europe, the more active sources being an Albanian-based cloud provider, under the IP being 109.104.151.108, and OVH Groupe SAS in France using 154.223.166.124, see our codebase [3] for more details. The only port we were able to detect, 8281, was used in 8 different shell commands.

5.1.4 Passwords

Using pattern matching, we did not find instances of plain-text passwords from the victims' machines being embedded in the code; we did however detect password changes initiated by the attackers. Besides noting that the passwords are not very creative and look randomly generated, the other interesting observation is we have 1108 cases where the same new password combination was used at least twice, suggesting we are dealing with the same attacker, especially in cases where the password was used in 5 or 10 different attacks.

5.1.5 SSH Keys

Here we noticed a very interesting pattern, we found 10800 attacks using the same SSH key, more than 90% of the total observations. Every time, the command sequence was used to gain persistence on the victim's machine via SSH, after performing privilege escalation. The fact the SSH key is the same every time leads us to hypothesize most of the attackers are part of the same group, alternatively, but less likely, the SSH key has been shared or sold amongst different hacker groups.

5.1.6 URLs

Besides the URLs hosting bash scripts we mentioned previously, we found interesting information on banned (as of 2023) domains, reported in table 2 All the domains with high frequency were

Domain	**Frequency**
bestony.club	28
myfrance.xyz	19
strtbiz.site	18
dns.cyberium.cc	1
samp-lsgw.net	1
kranskerstuff.kozow.com	1
antiq.ucoz.net	1
venus.lol	1

Table 2: Frequency of detected known malicious domains

linked to the spread of the Mēris botnet in 2021, used for distributed denial-of-service (DDoS) attacks thanks to a RouterOS vulnerability. We found this information on the MikroTik blog, from the producers of the RouterOS software: Meris botnet post. It's possible that the honeypot was using RouterOS to act as a router.

5.1.7 Summary

We believe a large portion of the commands logged belong to the same group of attackers. Since a large portion of the attacks focuses on CPU, system information gathering, privilege escalation, and establishing persistence, we checked for evidence of crypto-mining by attempting to extract common mining software and wallets. We found no evidence, based on this and the evidence provided above, we think many of the attacks aim at compromising the honeypot by establishing persistence. It's likely many attacks are performed by the same group of attackers, we would be able to identify further the attackers if we could hear from experts about whether they know the ISIS.sh and xxxbins.sh scripts.

5.2 Clustering

Bot clustering attempts achieved similar results, with the engineered one being more fine-grained and detailed in the number of clusters, finding 169 different groups as opposed to the 24 the NLP cluster detected, in both cases 2 groups of attackers are highlighted, with more dispersion in the case of the pure NLP procedure.

5.2.1 Feature clustering

Figure 2 shows how our clustering procedure clearly identifies two cluster groupings with strong similarity within the group, as well as some outlier attacks not close to any of the others. Since this is a visualization done on data that has been dimensionally reduced with UMAP, it's not possible to recover the dimensions used. Examining some of the commands belonging to each cluster, it seems that downloading and starting bash or other types of scripts was one of the main reasons to separate the two clusters.

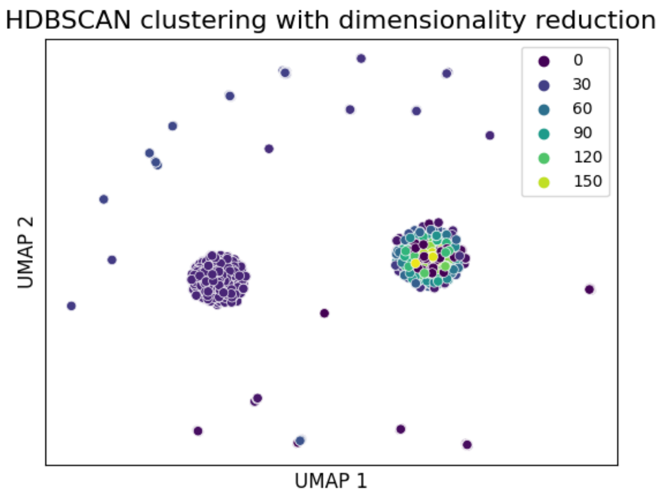

Figure 2: Own clustering
Source: Own work

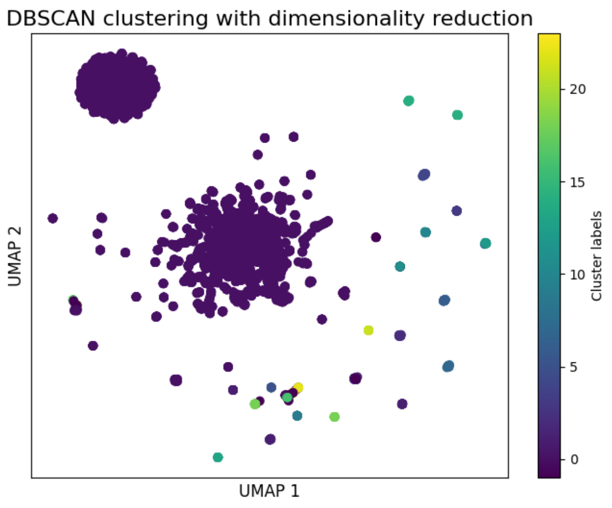

Figure 3: NLP clustering.
Source: Own work

5.2.2 NLP clustering

Directly working with the semantics and syntax of the commands using NLP also produced similar results, as shown in Figure 3. The difference lies in the granularity. In this case, the algorithm detects only 24 clusters, but the two main cluster groups are still well-delineated, reinforcing the results we obtained with our hand-crafted features.

6. Limitations

Our study, despite providing valuable insights, has some constraints. The data, obtained from a single honeypot over one month, may not fully reflect the variability of attacks across different systems and regions and lacks information on attack durations. Our feature extraction approach, while functional, could benefit from more advanced methods, such as comprehensive NLP information retrieval strategies or better-crafted patterns with expert input.

While we were successful in deriving relevant information, our interpretation was limited by our domain knowledge, particularly when understanding the function of certain bash scripts. Additionally, our clustering methods, though robust and illustrative, are difficult to interpret due to the inherent nature of the technique. Further research could potentially refine these machine learning approaches for more interpretable and accurate detection of cyber attack patterns.

7. Conclusions

We conducted a comprehensive analysis of a collection of shell commands derived from a honeypot, used to simulate a potential target for attackers. Our findings shed light on key patterns and methodologies used by attackers, revealing significant details about how these attacks are structured and executed. We've thoroughly enjoyed dissecting this data, and hope that our findings provide valuable insights. Thank you for taking the time to read our report.

References

[1] M. Musser and A. Garriott, "Machine learning and cybersecurity: Hype and reality," Tech. Rep., Jun. 2021. DOI: 10.51593/2020ca004. [Online]. Available: https://doi.org/10.51593/2020ca004.

[2] A. Consoli, *Data Security and Blockchain slides*, Accessed on May 17th, 2023.

[3] C. Grigioni, C. Pala, and A. Wey, *Github project repository*, https://github.com/ChristianPala/honeypot_analysis, Accessed on May 17th, 2023.

[4] L. McInnes, J. Healy, and J. Melville, *Umap: Uniform manifold approximation and projection for dimension reduction*, 2018. DOI: 10.48550/ARXIV.1802.03426. [Online]. Available: https://arxiv.org/abs/1802.03426.

[5] L. McInnes, J. Healy, and S. Astels, *Hdbscan: Hierarchical density based clustering*, Online Documentation, 2017. [Online]. Available: https://hdbscan.readthedocs.io/en/latest/.

[6] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.