Imperial College London

Undergraduate Research Opportunities Programme

Imperial College London

Department of Mathematics

Statistical cyber-security: Searching for patterns in commands issued by malicious intruders

Author:
Hanyang Liu

Supervisor:
Prof. Nick Heard
Dr. Francesco Sanna Passino

August 27, 2021

Abstract

Artificial intelligence is an active research and development area within cyber-security. The potential to automatically detect intruders within a computer network by building sophisticated, time-varying statistical models of their evolving behaviours, allowing outlying anomalous behaviours to be identified, has attracted much interest from industry and government. Less attention has been paid to modelling the actions and objectives of network intruders, largely due to a lack of available data containing known malicious behaviour. In this paper, we propose a method to use hierarchical agglomerative clustering and multinomial-dirichlet distribution to cluster the sessions of commands. The clustering successfully reduces dimensions while maintaining expected predictive probability.

Contents

1 Introduction **5**
1.1 Objectives 5
1.2 Related Work 5

2 Background **6**
2.1 Exploratory Data Analysis 6
2.2 Data Pre-processing 6
2.3 Hierarchical Agglomerative Clustering 7

3 Model **8**
3.1 Clustering 8
3.2 Prediction 8
3.3 Model Architecture 9

4 Evaluation **11**

5 Conclusion **12**

A Appendix **13**

List of Figures

2.1 Exploratory Data Analysis. The top graph shows the number of appearance of top 10 frequent initial 3 commands. The middle graph shows the number of appearance of top 10 frequent initial 3 commands in sessions which contain top 100 frequent sequence of commands. The bottom graph shows the frequency of top 10 frequent initial 3 commands in sessions which contain top 100 frequent sequence of commands 6

3.1 Flowchart of Intuitive Model 9
3.2 Flowchart of Our Model 10

List of Tables

4.1 Expected predictive probabilities 11
4.2 Number of clusters 11

A.1 Microsoft honeypot dataset 13

Chapter 1

Introduction

1.1 Objectives

Artificial intelligence is an active research and development area within cyber-security. The potential to automatically detect intruders within a computer network by building sophisticated, time-varying statistical models of their evolving behaviours, allowing outlying anomalous behaviours to be identified, has attracted much interest from industry and government. Less attention has been paid to modelling the actions and objectives of network intruders, largely due to a lack of available data containing known malicious behaviour. This project will investigate a new data resource, where in collaboration with Microsoft a so-called "honeypot" has been planted in the college network to entice network intruders and record their actions. Discoveries from this research theme have the potential to provide enterprise situational awareness of the intentions of an intruder detected on a network, allowing a targeted response such as preventing access to the most sensitive data held by an organisation.

1.2 Related Work

Some work [SS21]studies ways to predict attackers' next commands from the shell commands using Levenshtein distance without analyzing network traffic.

Chapter 2

Background

2.1 Exploratory Data Analysis

There are 124883 out of 125280 sessions which appear only once in the Microsoft dataset. The dataset can be found in Appendix A.1 and sessions look like Appendix A.2. If we divide sessions into sequences of commands of length 12[SS21], many sequences seem to be naturally clustered by frequncies of appearance, as shown in Fig 2.1. Within each cluster, not only the distribution of initial few commands seem very homogeneous, but also the sequences themselves.

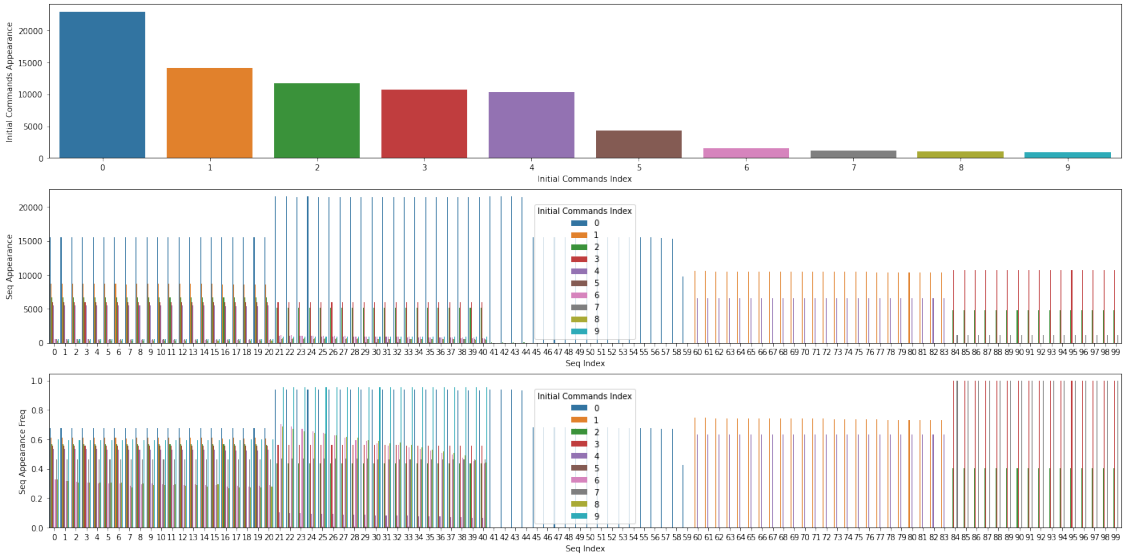

Figure 2.1: Exploratory Data Analysis. The top graph shows the number of appearance of top 10 frequent initial 3 commands. The middle graph shows the number of appearance of top 10 frequent initial 3 commands in sessions which contain top 100 frequent sequence of commands. The bottom graph shows the frequency of top 10 frequent initial 3 commands in sessions which contain top 100 frequent sequence of commands

This encourages us to try to cluster and predict the commands based on the initial commands in a session.

2.2 Data Pre-processing

There are 3 major objectives in data processing.

1. Seperate multiple commands in one command line.
2. Use **r'[a-zA-Z0-9_\.\-*]+'** as regexp to tokenise commands into words.
3. Replace the words which only appear in one session by rarecommand.

The sample code can be found in Appendix A.1

2.3 Hierarchical Agglomerative Clustering

Hierarchical clustering is a method which exhaustively merge pairs of clusters. Each observation starts in its own cluster. Iteratively, the closest (most similar) pairs of clusters merge, until there is only one cluster or the threshold is achived.

Chapter 3

Model

3.1 Clustering

We aim to cluster sessions and predict commands based on the first d commands of each session. Therefore, the model can be divided into clustering part and prediction part.

We need to assign each session a label, so that we can evaluate how homogeneous each cluster is, and we can use the labels as surrogates for predictions. We divide sessions into sequences of commands of length l_c[SS21]. For example, if a session is **A-B-C-D-E-F** and $l_c = 3$, we will have sequences **A-B-C**, **B-C-D**, **C-D-E**, and **D-E-F**. Sequences of commands are ranked by their number of appearances, i.e. the most frequent sequence is ranked 1, the second most frequent sequence is ranked 2. For each session x, we assign label y as the highest rank which can be achieved by the sequences obtained from x. We denote x_j to be the jth command of x, and $x_{:d}$ to be the initial d commands of x. Suppose we have N unique initial d commands and the lowest rank of our sessions is K, an N by K matrix of counts A can be constructed as following: A_{pq} is the number of sessions x for which $x_{:d}$ is the pth unique first d commands, and $y = q$.

Hierarchical Agglomerative Clustering method is used for our clustering. The idea is that we start with N clusters, and pairwisely merge closest clusters in a predefined metric until certain threshold. In our case, for each initial cluster p, we assume that the unknown categorical probability $\theta = (\theta_1, \ldots, \theta_K)$ follows a Dirichlet prior,

$$\theta \sim \text{Dirichlet}(\beta_1, \ldots, \beta_K), \tag{3.1}$$

with each $\beta_j > 0$. Hence we can calculate $P(A_{p,:}|\theta)$. Let us define E as the dissimilarity matrix among clusters. We then use $E_{pq} = \log(P(A_{p,:}, A_{q,:}|\theta)) - \log(P(A_{p,:}|\theta)) - \log(P(A_{q,:}|\theta))$ to calculate the gain in similarity after merging the cluster p and q. Finally, we can perform the agglomerative clustering as usual. We use minimum linkage for agglomerative clustering.

3.2 Prediction

We now come to prediction after finishing clustering. Suppose a particular cluster contains $M \leq N$ data points, corresponding to data matrix row positions $i_1, \ldots, i_M$ of the $N \times K$ data matrix of counts $A = (a_{ij})$. For $1 \leq j \leq K$, let

$$n_j = \sum_{\ell=1}^{M} a_{i_\ell j} \tag{3.2}$$

be the total frequency of category j in the cluster, and let $n_\bullet = \sum_j n_j$. Suppose the unknown categorical probabilities $\theta = (\theta_1, \ldots, \theta_K)$ for the cluster follow a Dirichlet prior,

$$\theta \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K), \tag{3.3}$$

with each $\alpha_j > 0$ and $\alpha_\bullet = \sum_j \alpha_j$. Then the marginal likelihood of the sequence of observed categories which gave rise to the bin frequencies $n_1, \ldots, n_k$ is

$$\frac{\Gamma(\alpha_\bullet)}{\Gamma(\alpha^*_\bullet)} \prod_{j=1}^{K} \frac{\Gamma(\alpha^*_j)}{\Gamma(\alpha_j)}, \tag{3.4}$$

where $\alpha^*_j = \alpha_j + n_j$ for $1 \leq j \leq K$ and $\alpha^*_\bullet = \sum_j \alpha^*_j = \alpha_\bullet + n_\bullet$. Furthermore, the posterior distribution for θ after observing $n_1, \ldots, n_K$ is

$$\theta \mid n_1, \ldots, n_K \sim \text{Dirichlet}(\alpha^*_1, \ldots, \alpha^*_K). \tag{3.5}$$

In particular, this means

$$\mathbb{E}(\theta_j \mid n_1, \ldots, n_K) = \frac{\alpha^*_j}{\alpha^*_\bullet}. \tag{3.6}$$

The quantity (3.6) is also the predictive probability for category j in that cluster. So if we have a new observation assigned to the cluster with label y_{N+1}, the prediction score we get from that observation is

$$\frac{\alpha^*_{y_{N+1}}}{\alpha^*_\bullet}. \tag{3.7}$$

For each leaf node, the expected predictive probability in that cluster is

$$\mathbb{E}(p(x)) = \sum_{j=1}^{K} (\frac{\alpha^*_j}{\alpha^*_\bullet})^2. \tag{3.8}$$

Therefore, we can calculate the weighted sum of the expected predictive probabilities across the clusters, where the cluster weight is proportional to the number of observations in that cluster. This can be an indication of our overall cluster result.

3.3 Model Architecture

We can now combine the clustering and prediction part to construct our model. Suppose we want to train depth d, which means we know the initial d commands of all sessions to help us cluster. The intuitive way of combination is like Fig 3.1. However, the main drawback is that it cannot

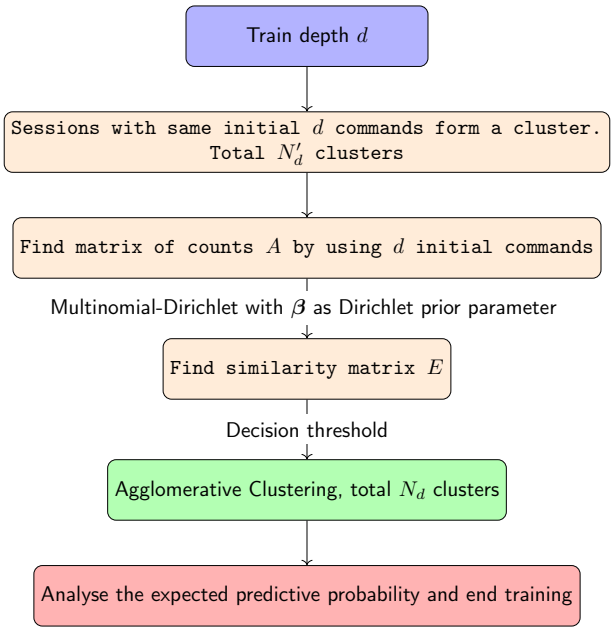

Figure 3.1: Flowchart of Intuitive Model

learn from previous clustering. For example, the initial 7 commands **A-B-C-D-E-F-G** and **A-B-C-D-E-F-H** are completely different in the intuitive model. The initial 6 commands, however,

are considered same. Many times the command **G** and **H** may only differ by a URL. Treating **A-B-C-D-E-F-G** and **A-B-C-D-E-F-H** and **I-I-I-I-I-I-I** as 3 completely different cluster is not reasonable.

Therefore, we propose a model which can both combine clustering and prediction, and learn from previous depth. Suppose we want to cluster in depth d. As shown in Fig 3.2, the idea we introduce

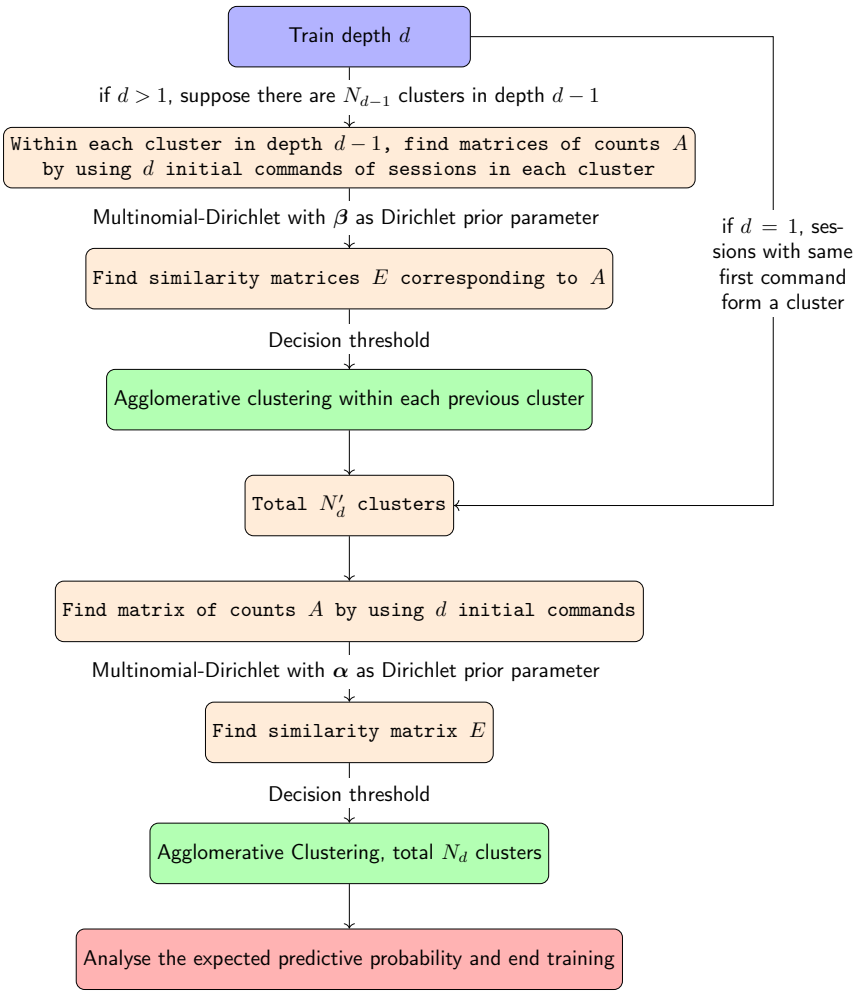

Figure 3.2: Flowchart of Our Model

is to cluster within each previous cluster in depth $d-1$ first. After this is done, we try to see if the existing clusters can be further combined. This assymetric clustering procedure gives advantage to sessions with different initial d commands, but in the same cluster in depth $d-1$.

Chapter 4

Evaluation

We set our Dirichlet prior $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to be 0.0001 in all categories, given we have investigated that K, the size of categories in a cluster, is around 1000 for this dataset. The cluster threshold we use is 0.3. We also experiment different length of sequence of commands, l_c.

In order to evaluate our performance, we design 2 baselines using $l_c = 12$ to calculate matrices A and E. **Baseline 1** is to keep every unique initial d commands in its own clusters for depth d. **Baseline 2** is to put all observations into a single cluster, i.e. not doing any clustering.

	Depth 1	Depth 2	Depth 3	Depth 4	Depth 5	Depth 6
$l\_c = 3$	0.476399	0.524108	0.539341	0.547319	0.551775	0.555026
$l\_c = 6$	0.460124	0.50814	0.524318	0.532581	0.536925	0.546313
$l\_c = 9$	0.453345	0.501228	0.527811	0.536087	0.541361	0.55031
$l\_c = 12$	0.450155	0.49781	0.530307	0.538657	0.544359	0.55324
Baseline 1	0.454241	0.517255	0.549589	0.568437	0.574061	0.603752
Baseline 2	0.432368	NA	NA	NA	NA	NA

Table 4.1: Expected predictive probabilities

	Depth 1	Depth 2	Depth 3	Depth 4	Depth 5	Depth 6
$l\_c = 3$	11	41	66	76	80	89
$l\_c = 6$	14	48	71	81	94	104
$l\_c = 9$	14	48	73	82	95	101
$l\_c = 12$	14	47	69	81	95	102
Baseline 1	1796	7380	14237	19537	19931	28116
Baseline 2	1	NA	NA	NA	NA	NA

Table 4.2: Number of clusters

Table 4.1 shows the result of our expected predictive probabilities. Table 4.2 shows the result of number of clusters. For Baseline 1, as the number of clusters grows drastically fast as depth increases, most clusters will have very few elements, except for the few dominant ones. Therefore, we have many small homogeneous clusters in Baseline 1 and we can expect the expected predictive probabilities increases as depth increases. For Baseline 2, it is meaningless to talk about depth as we put all observations into a single cluster. its expected predictive probability is lowest among all, as we expect. However, even if we put all obervations into 1 single cluster, the expected predictive probability, 0.432368, is still quite high. This can be due to the imbalance of dataset which can limit our performance gain as depth increases.

A sample clustering of initial commands can be found in Appendix A.3. We can see that for each l_c, the expected probabilities increase as depth increases, and the numbers of clustering are much fewer than Baseline 1.

Chapter 5

Conclusion

The result shows that our method can successfully allocate the sessions of commands from thousands of initial clusters to fewer than 100 clusters, while maintaining the expected predictive probability.

Appendix A

Appendix

Details of code can be found in https://github.com/Hanyang97/UROP

	Protocol	Commands	ID	TimesSeen	FirstSeen	LastSeen
0	Telnet	[enable, system...	14cc70cddce...	1546097	2019-07-...	2019-11-...
1	Telnet	[enable, system ...	ba4417b91e...	1149119	2019-07-...	2019-09-...
2	Telnet	[sh, ...	e421974df...	1279661	2019-07-...	2019-09-...
3	Telnet	[sh, bin...	5d9c6f7...	346137	2019-07-...	2019-11-...
4	Telnet	[shell, sh...	0d1edf8dad...	2435245	2019-07-...	2019-11-...

Table A.1: Microsoft honeypot dataset

```
from nltk.tokenize import RegexpTokenizer
from gensim.corpora import Dictionary


def clean_commands(dat, no_below=2, no_above=1.1):
    """
    This function
    1. splits multiple commands in the same line
    2. tokenize the commands
    3. replace rare commands by rarecommand

    :param dat: dataset
    :param no_below: Keep tokens which are
        contained in at least no_below documents.
    :param no_above: Keep tokens which are
        contained in no more than no_above documents
    (fraction of total corpus size, not an absolute number).

    :return sessins_token_list: tokenized list of sessions
        of commands
    :return dictionary: dictionary generated
    """
    # for commands splitted by ;
    sessions = []
    for session in dat['Commands']:
        sessions.append([])
        for command in session:
            sessions[-1] += command.split(';')
    # tokenizer
    tokenizer = RegexpTokenizer(r'[a-zA-Z0-9_\.\-\*]+')
    sessions_list = []
```

```
    commands_list = []
    for session in sessions:
        sessions_list.append([])
        commands_list.append([])
        for command in session:
            command_token = tokenizer.tokenize(command)
            sessions_list[-1] += [command_token]
            commands_list[-1] += command_token
    dictionary = Dictionary(commands_list)
    dictionary.filter_extremes(no_below, no_above)
    # repleace rare commands by rarecommand
    dictionary.id2token[-1] = 'rarecommand'
    dictionary.token2id['rarecommand'] = -1
    sessions_token_list = []
    for session in sessions_list:
        sessions_token_list.append([])
        commands_token_list = []
        for command in session:
            idxs = dictionary.doc2idx(command)
            commands_token_list.append(
                '␣'.join([dictionary[idx] for idx in idxs]))
        sessions_token_list[-1] += commands_token_list

    return sessions_token_list, dictionary
```

Listing A.1: Python code for tokeniser

```
['enable',
'system',
'shell',
'sh',
'>/tmp/.ptmx && cd /tmp/',
'>/var/.ptmx && cd /var/',
'>/dev/.ptmx && cd /dev/',
'>/mnt/.ptmx && cd /mnt/',
'>/var/run/.ptmx && cd /var/run/',
'>/var/tmp/.ptmx && cd /var/tmp/',
'>/.ptmx && cd /',
'>/dev/netslink/.ptmx && cd /dev/netslink/',
'>/dev/shm/.ptmx && cd /dev/shm/',
'>/bin/.ptmx && cd /bin/',
'>/etc/.ptmx && cd /etc/',
'>/boot/.ptmx && cd /boot/',
'>/usr/.ptmx && cd /usr/',
'/bin/busybox rm -rf lzrd oxdedfgt',
'/bin/busybox cp /bin/busybox lzrd; >lzrd;
/bin/busybox chmod 777 lzrd;
/bin/busybox lizrd',
'/bin/busybox cat /bin/busybox || while read i;
do echo $i; done < /bin/busybox',
'/bin/busybox lizrd']
```

Listing A.2: Sample sessions

```
('sh',
 'linuxshell',
 'bah',
 'hostname Ex0_1115',
 'bin busybox Ex0',
 'bin busybox ps'),
```

```
('sh',
 'linuxshell',
 'bah',
 'hostname Ex0_8572',
 'bin busybox Ex0',
 'bin busybox ps'),
('sh',
 'linuxshell',
 'bah',
 'hostname Ex0_0447',
 'bin busybox Ex0',
 'bin busybox ps')
```

Listing A.3: Sample cluster sclices

Bibliography

[SS21] Farhan Sadique and Shamik Sengupta. Analysis of attacker behavior in compromised hosts during command and control, 2021.