# Pattern Extraction for Behaviours of Multi-Stage Threats via Unsupervised Learning

**Ahmed A. Alghamdi Giles Reger**

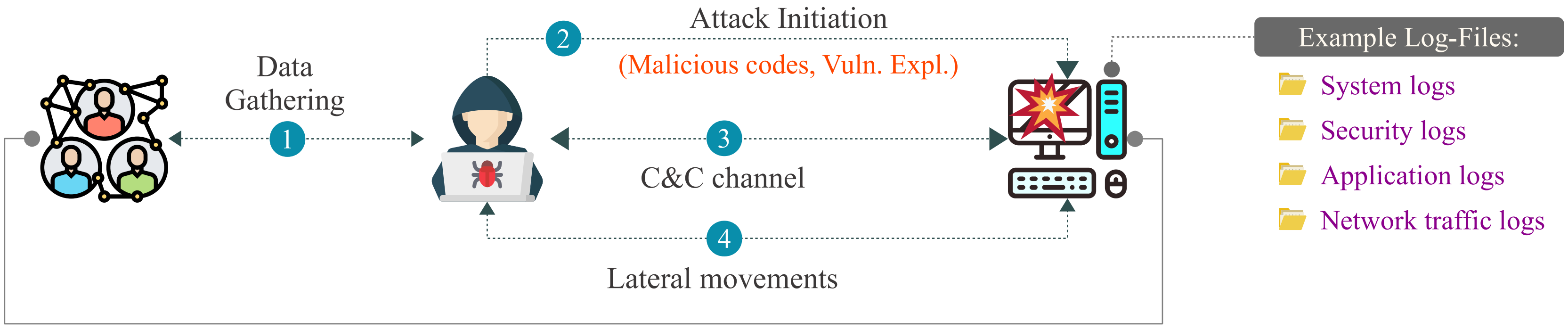
Presenter: Ahmed Abdulrahman Alghamdi

Email: ahmed.alghamdi@manchester.ac.uk

June 17th, 2020

# Introduction

- What are multi-stage threats and why they are difficult to be detected?
- What traces can be collected from such attacks?
- How those logs can be analysed to identify such attacks?



1 Data Gathering

2 Attack Initiation

(Malicious codes, Vuln. Expl.)

3 C&C channel

4 Lateral movements

Example Log-Files:

- System logs
- Security logs
- Application logs
- Network traffic logs

# Motivation

01 Logs of systems, applications, and network activities can provide a significant amount of information that can be analysed using machine learning to determine abnormal or new patterns of behaviours.

02 Many models are proposed by researchers for this purpose. Those models either rely on:

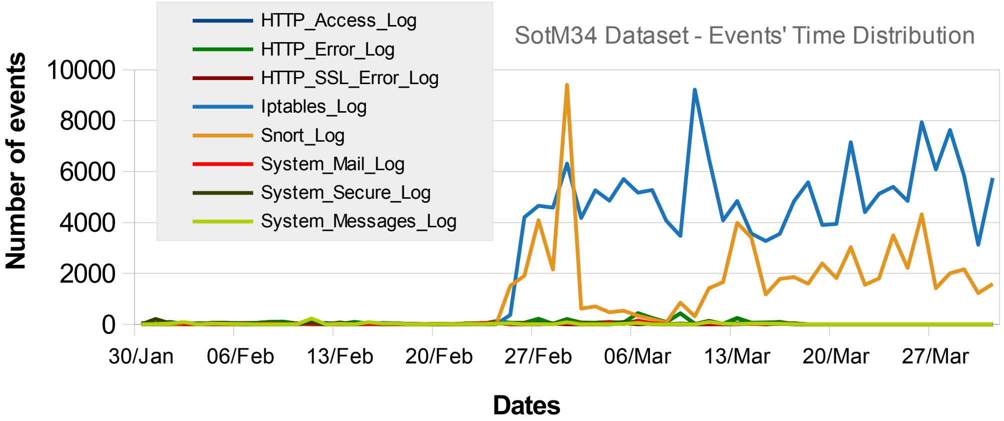
- **Supervised machine learning** algorithms, which require enough labelled real-life training data (difficult to obtain for multi-stage threats like APT)
- **Unsupervised machine learning** algorithms that suffer from the high-performance requirements, low accuracy rates, and the parameters adjustment method.

03 In our paper, we propose an unsupervised framework that collects and analyses heterogeneous log-files to extract patterns and correlations of behaviours without the need of training data or previous knowledge of threats.

# Experimental Dataset

For the evaluation of our framework, we used the SotM34[1] dataset, which consists of heterogeneous log-files collected from a Honeynet that was targeted by several attacks in which some of them have successfully managed to compromise the targeted device.

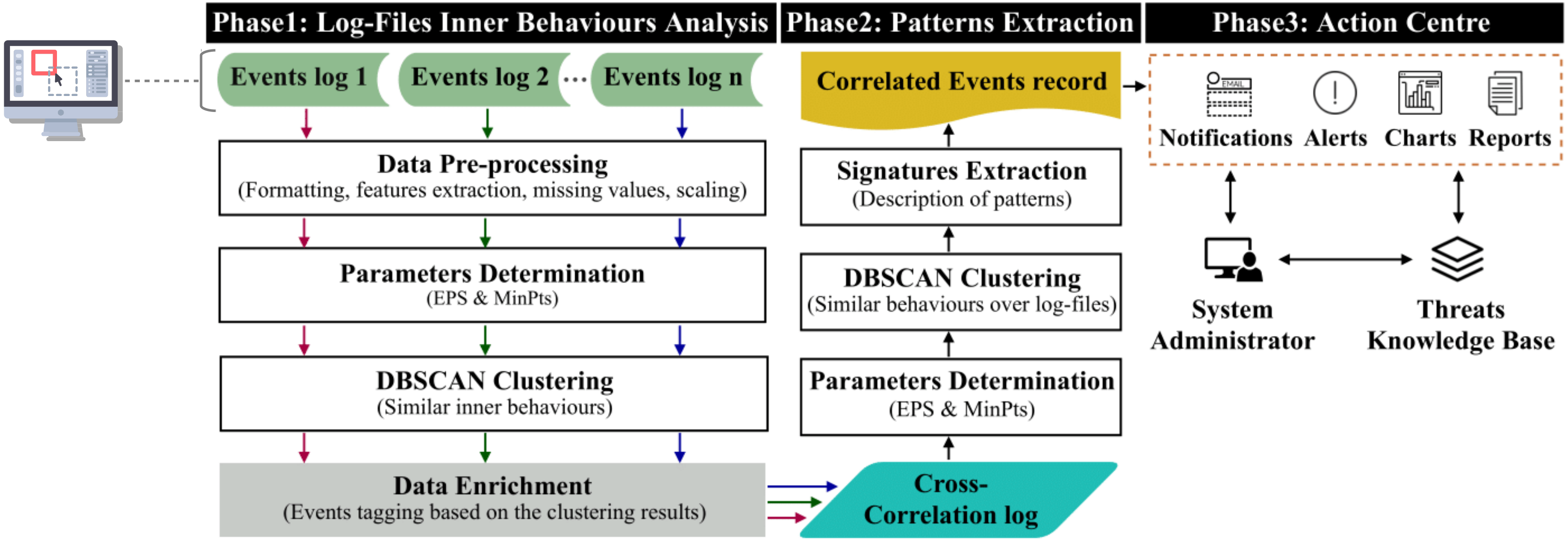
The dataset is provided in its original format (TXT) without pre-processing or labelling. Therefore, we have produced a pre-processed and labelled version of the dataset.



[1]: SotM34, available at: http://honeynet.onofri.org/scans/scan34/

# Our framework design

> To extract inner-behaviours and their correlations from the collected heterogeneous log-files, the proposed framework is constructed by integrating two main phases of automated unsupervised data analysis along with an action centre.



Phase1: Log-Files Inner Behaviours Analysis

Events log 1 · Events log 2 ··· Events log n

**Data Pre-processing**
(Formatting, features extraction, missing values, scaling)

**Parameters Determination**
(EPS & MinPts)

**DBSCAN Clustering**
(Similar inner behaviours)

**Data Enrichment**
(Events tagging based on the clustering results)

Phase2: Patterns Extraction

**Cross-Correlation log**

**Parameters Determination**
(EPS & MinPts)

**DBSCAN Clustering**
(Similar behaviours over log-files)

**Signatures Extraction**
(Description of patterns)

**Correlated Events record**

Phase3: Action Centre

Notifications · Alerts · Charts · Reports

System Administrator

Threats Knowledge Base

# Phase1: Log-Files’ Inner Behaviours Analysis

The goal of this phase is to identify sets of behaviours inside each imported log-file using an unsupervised clustering method. This phase consists of 4 main steps:

1 **Data Pre-Processing**
Converting events in the dataset from their original format (txt) into numerical data

2 **Parameters Calculation**
Calculating the EPS and MinPts parameters for the clustering process

3 **Data Clustering**
Using DBSCAN for identifying inner behaviours inside log-files

4 **Data Enrichment**
Predicted labels resulting from the clustering step are used to tag the events

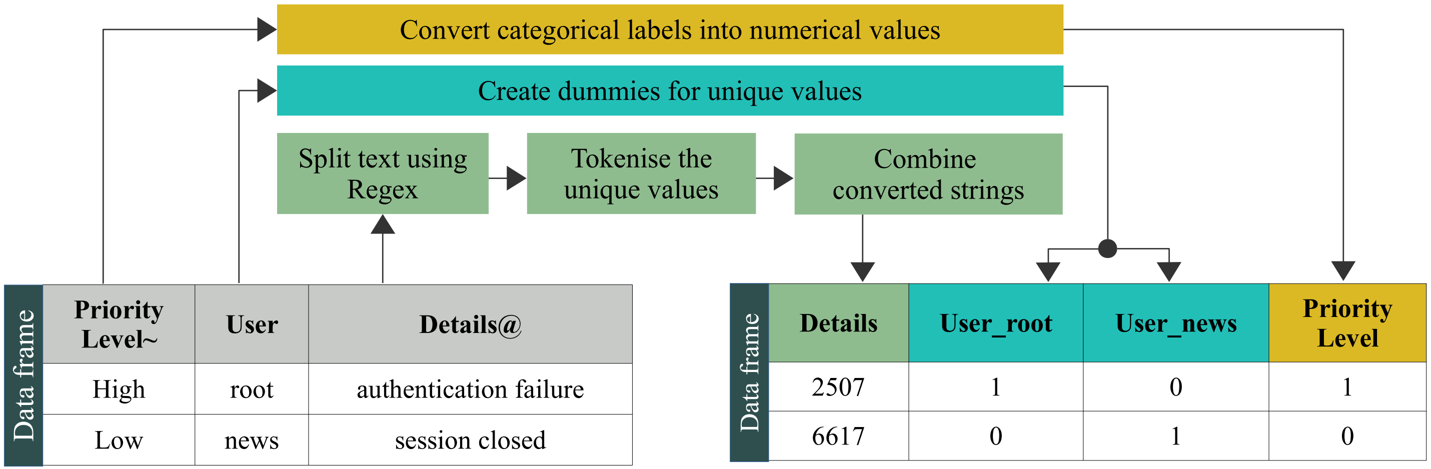
# Data Pre-Processing

- Converting imported log-files into data frames with feature sets and dealing with missing values.
- Non-numeric values in the dataset are converted into numerical values as follows:

● Ordinal Categorical Data: values of this type have logical order or scale between them (e.g. Priority Levels)

● Nominal Categorical Data: Descriptive values with no connections between them (e.g. Names of devices).

● Text Data: web links, directories, and symbols. (e.g. : ”File does not exist: /var/www/html/scripts”)



Convert categorical labels into numerical values

Create dummies for unique values

Split text using Regex

Tokenise the unique values

Combine converted strings

Data frame

| Priority Level~ | User | Details@ |
|---|---|---|
| High | root | authentication failure |
| Low | news | session closed |

Data frame

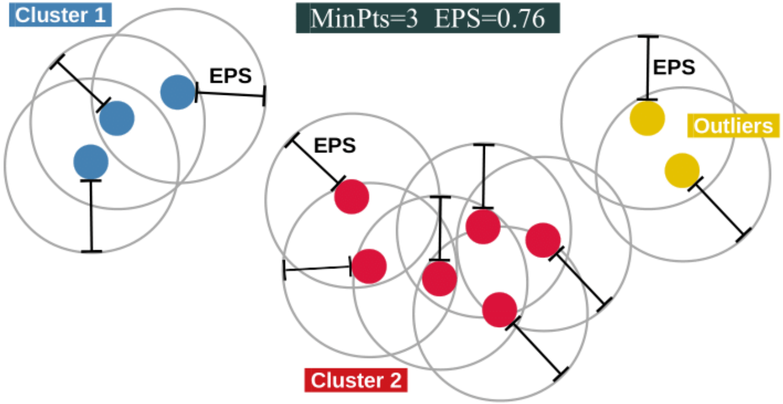
| Details | User_root | User_news | Priority Level |
|---|---|---|---|
| 2507 | 1 | 0 | 1 |
| 6617 | 0 | 1 | 0 |

# Clustering Algorithm

- Many unsupervised clustering algorithms such as k-means, DBSCAN, Grid-Clustering, and EM were evaluated for the purpose of selecting the most suitable one in terms of speed, robustness, and the parameters tuning.
- DBSCAN was selected for our framework due to its abilities to handle outliers and the identification of clusters without prior knowledge of the actual number of clusters.

The DBSCAN clustering algorithm uses two main parameters to perform the analysis:

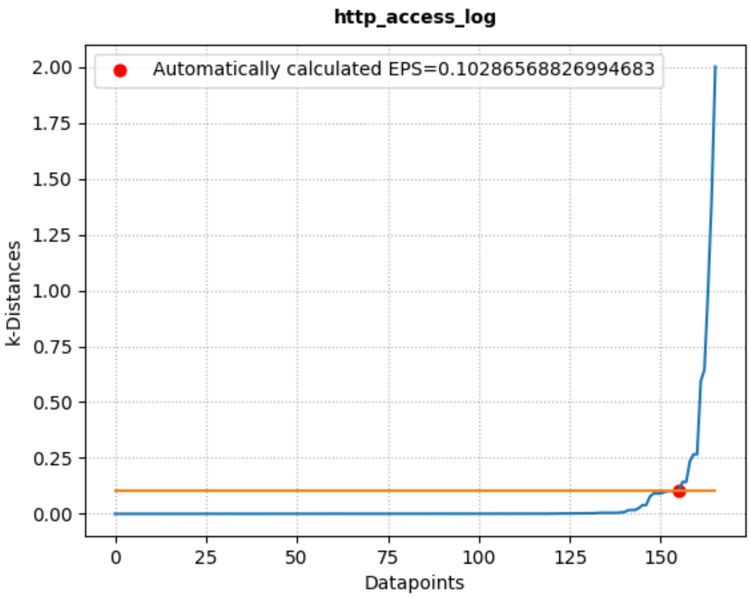
- **MinPts:** The minimum points required to form a cluster.
- **Epsilon:** The maximum distance between two examples for them to be considered in the same cluster.



# Parameters Tuning

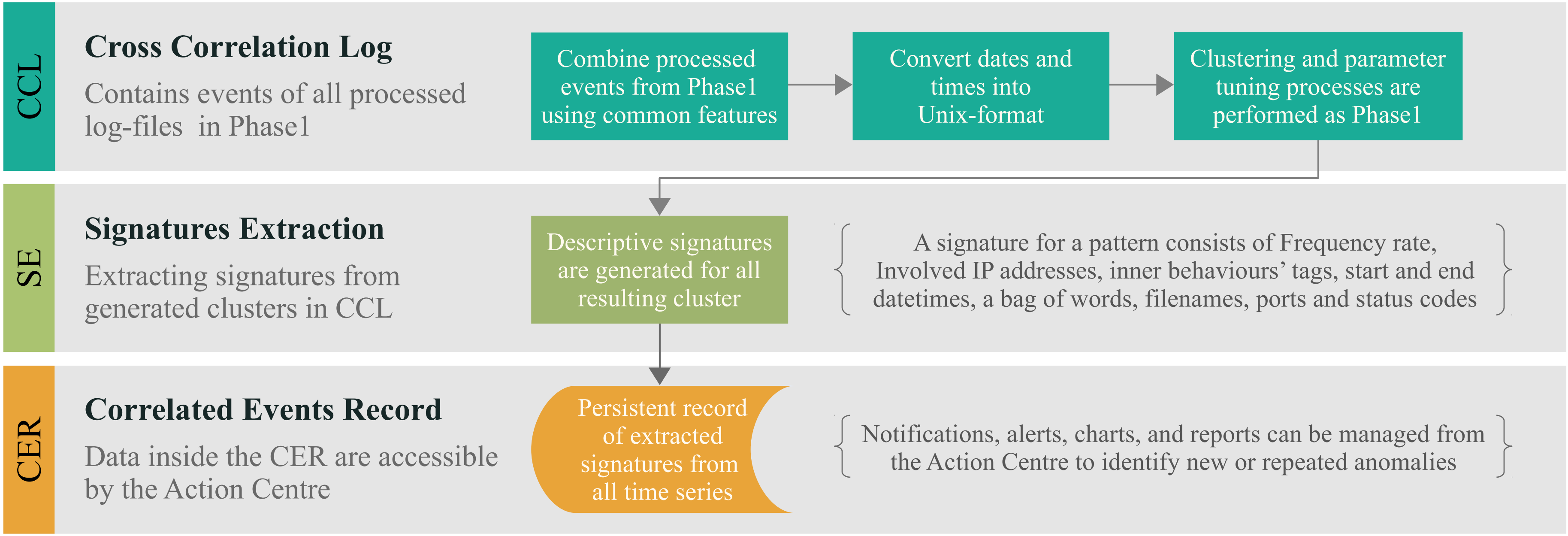
- For the EPS parameter, we calculate the optimal value by finding the distances between all data points, and then calculate the mean of the unique slops between those distances. Finally, the calculated mean is used as a value for the EPS parameter.

- The value for the MinPts parameter is selected based on the processing methodology of our framework.



# Phase2: Patterns Extraction

The aim of this phase is to correlate behaviours based on a set of common features that exist in log-files. The following points explain components of this analysis phase:



CCL

**Cross Correlation Log**

Contains events of all processed log-files in Phase1

Combine processed events from Phase1 using common features → Convert dates and times into Unix-format → Clustering and parameter tuning processes are performed as Phase1

SE

**Signatures Extraction**

Extracting signatures from generated clusters in CCL

Descriptive signatures are generated for all resulting cluster

{ A signature for a pattern consists of Frequency rate, Involved IP addresses, inner behaviours' tags, start and end datetimes, a bag of words, filenames, ports and status codes }

CER

**Correlated Events Record**

Data inside the CER are accessible by the Action Centre

Persistent record of extracted signatures from all time series

{ Notifications, alerts, charts, and reports can be managed from the Action Centre to identify new or repeated anomalies }

# Results Discussion

**Phase1:** Results reveal that the applied methodology can process high numbers of events in short times. For examples, the Snort file which contains about 69 thousand events, was processed in less than 10 seconds. Also, the results show that the framework has grouped events successfully with high accuracy rates due to the followed approach of data preprocessing and clustering.

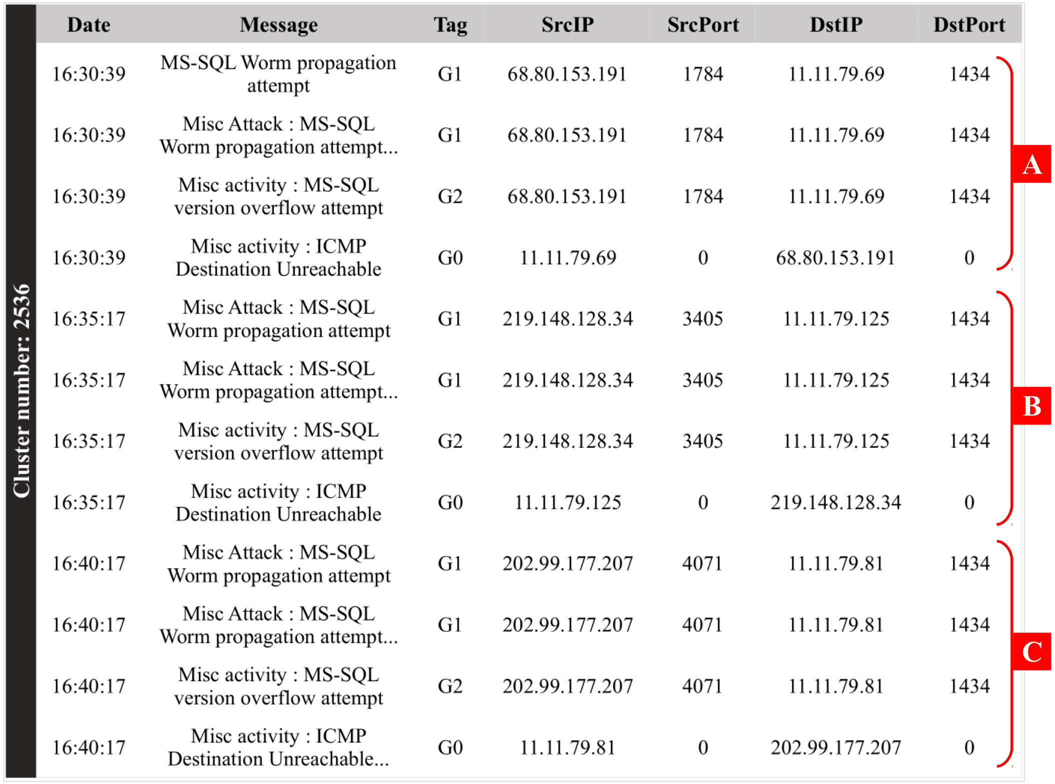
| | No. of Events | Pre-processing | EPS Function | DBSCAN clustering | Resulting Clusters | Homo-geneity | Comple-teness | V-measure | ARI | AMI |
|---|---|---|---|---|---|---|---|---|---|---|
| **HTTP Access** | 3554 | 0.179 sec. | 0.005 sec. | 0.182 sec. | 20 | 99.93% | 93.86% | 96.80% | 97.37% | 93.77% |
| **HTTP Error** | 3692 | 0.095 sec. | 0.008 sec. | 0.351 sec. | 24 | 99.87% | 91.22% | 95.35% | 99.22% | 91.08% |
| **HTTP SSL Error** | 374 | 0.017 sec. | 0.003 sec. | 0.003 sec. | 4 | 100% | 100% | 100% | 100% | 100% |
| **Syslog Messages** | 1166 | 0.052 sec. | 0.0124 sec | 0.053 sec. | 24 | 100% | 85.64% | 92.26% | 97.01% | 85.07% |
| **Syslog Secure** | 1587 | 0.005 sec. | 0 sec. | 0.055 sec. | 6 | 100% | 100% | 100% | 100% | 100% |
| **Syslog Mail** | 1172 | 0.091 sec. | 0.005 sec. | 0.041 sec. | 12 | 99.64% | 82.71% | 90.39% | 87.49% | 82.45% |
| **SNORT** | 69039 | 2.839 sec. | 0.008 sec. | 6.763 sec. | 31 | 99.99% | 94.03% | 96.92% | 92.70% | 94.02% |

# Results Discussion

**Phase2:** The preprocessing of CCL (80584 events) was applied in 1.9 seconds, while the clustering function was applied in 3.8 sec.

A total of 3009 clusters were generated from the clustering process. Those clusters represent patterns of behaviours from over two months of recorded events.

The pairs of source/destination IP addresses were used to evaluate the homogeneity of the resulting clusters. 72.15% of the clusters contain unique pairs of IP addresses. In contrast, the rest of clusters contain overlapping sessions which indicate similarities between those sessions in times or in types of behaviours.



Cluster number: 2536

| Date | Message | Tag | SrcIP | SrcPort | DstIP | DstPort |
|---|---|---|---|---|---|---|
| 16:30:39 | MS-SQL Worm propagation attempt | G1 | 68.80.153.191 | 1784 | 11.11.79.69 | 1434 |
| 16:30:39 | Misc Attack : MS-SQL Worm propagation attempt... | G1 | 68.80.153.191 | 1784 | 11.11.79.69 | 1434 |
| 16:30:39 | Misc activity : MS-SQL version overflow attempt | G2 | 68.80.153.191 | 1784 | 11.11.79.69 | 1434 |
| 16:30:39 | Misc activity : ICMP Destination Unreachable | G0 | 11.11.79.69 | 0 | 68.80.153.191 | 0 |
| 16:35:17 | Misc Attack : MS-SQL Worm propagation attempt | G1 | 219.148.128.34 | 3405 | 11.11.79.125 | 1434 |
| 16:35:17 | Misc Attack : MS-SQL Worm propagation attempt... | G1 | 219.148.128.34 | 3405 | 11.11.79.125 | 1434 |
| 16:35:17 | Misc activity : MS-SQL version overflow attempt | G2 | 219.148.128.34 | 3405 | 11.11.79.125 | 1434 |
| 16:35:17 | Misc activity : ICMP Destination Unreachable | G0 | 11.11.79.125 | 0 | 219.148.128.34 | 0 |
| 16:40:17 | Misc Attack : MS-SQL Worm propagation attempt | G1 | 202.99.177.207 | 4071 | 11.11.79.81 | 1434 |
| 16:40:17 | Misc Attack : MS-SQL Worm propagation attempt... | G1 | 202.99.177.207 | 4071 | 11.11.79.81 | 1434 |
| 16:40:17 | Misc activity : MS-SQL version overflow attempt | G2 | 202.99.177.207 | 4071 | 11.11.79.81 | 1434 |
| 16:40:17 | Misc activity : ICMP Destination Unreachable... | G0 | 11.11.79.81 | 0 | 202.99.177.207 | 0 |

# Summary

### Results Summary

Our results demonstrate that the framework can:

(i) efficiently cluster inner-behaviours of log-files with high accuracy rates

(ii) extract patterns of malicious behaviour and correlations between those patterns from real-world data

### Further Work

- Applying patterns recognition techniques on the collected data to allow further analysis of malicious behaviours. This will allow users to better understand relations between threats.

### GitHub Repository

The source code and the dataset are available on:
https://github.com/aag1990/UAHL

### Contact Details

Email: ahmed.alghamdi@manchester.ac.uk
Linkedin: www.linkedin.com/in/alghamdi-ahmed