The Alan Turing Institute

Data Study Group Final Report

Imperial College London,
Los Alamos National Laboratory,
Heilbronn Institute

10 – 14 December 2018

Developing data science tools for improving enterprise cyber-security

https://doi.org/10.5281/zenodo.3558251

This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1

Contents

1 Executive summary

1.1 Topic overview

Cyber attacks pose a substantial risk on nations, companies and individuals, and with the ever rising use of technology and process automation, the risk of high impact attacks is daunting. Further, standard defences primarily use signature based methods that have serious flaws; for example, they are only useful against known malware. As such, there is increasing attention from both government and industry towards the use of statistical, machine learning and broader data science techniques for improving cyber-security approaches. Data-driven analytics which monitor network traffic, user behaviour and running processes, amongst other statistics, are becoming the most likely means for detecting the intruder. Timely detection of the attacker is paramount if they are to be prevented from achieving a malicious objective and this drives the need for both broader and deeper data analysis tools of such data.

1.2 Challenge overview

This Data Study Group (DSG) challenge aims to carry out a preliminary investigation of some statistical and machine learning tools for analysing certain types of cyber-relevant data sources. Specifically, we consider a unified repository released by Los Alamos National Laboratory (LANL) comprising both network flow records and process-level Windows service logs collected on the same enterprise computer network over a three-month period.

Three aspects tackled in this challenge include anomaly detection, data fusion, and visualisation. Within the DSG week, we have aimed to consider if fusion of the data sources can give a more coherent view of this network's behaviour and what visualisations can be used to aid a prioritisation of of potential threats for analysts. Other explorations developed during this study group have been provided and the potential applications or limitations described. This report does not provide a 'white paper' on cyber-security tools, but rather aims to detail the methods attempted by different groups of participants in this DSG.

1.3 Data overview

CSV format summary data is taken from the Unified Host and Network Dataset [29], available from `https://csr.lanl.gov/data/2017.html`. This data contains a subset of internal network and host event logs collected from the Los Alamos National Laboratory (LANL) enterprise network over the course of approximately 90 days. Specifically, the data considered is:

- Unique network flow connections per day between pairs of communicating devices on the network (21,286,000 events per day on average),
- Unique authentication events per day collected from computers running the Microsoft Windows operating system (177,000 events per day on average),
- Unique process start events and the parent processes that invoked them per day collected from computers running the Microsoft Windows operating system (622,000 events per day on average).

The labels for devices and user accounts, whilst deidentified to protect the security of LANLs operational IT environment, match across all the datasets to allow for data fusion and proper analysis. There are approximately 60,000 unique devices, 15,000 unique user accounts and 26,000 unique process names across the 90 days. More detailed information about the datasets can be found in [29] and in Section 2. In addition, labelled 'red team' data consisting of known malicious authentications, process starts and compromised devices is used for validation of any anomaly detection methods. This sample malicious behaviour data exists for days 57–90.

1.4 Main objectives

This challenge aims to take a first step towards exploring a broad range of methods to effectively analyse relevant data sources for cyber-security applications. The sponsors of this Data Study Group (DSG) have identified three aspects of potential interest.

1. Data fusion: Investigating how data sources can be merged to give a

more concise view of the available information, whilst maintaining completeness. This is with the aim of reducing dimensionality, developing more powerful network monitoring tools and potentially establishing situational awareness.

2. Visualisation: Exploring what meaningful and informative graphical visualisations of cyber data can be generated to aid an analyst in prioritising which potential threats require most urgent attention.

3. Anomaly detection: Detecting unlabelled malicious activity hidden within the data. This malicious behaviour is refered to as 'red team' data, as discussed in Section 1.3.

There are of course difficulties and challenges associated with these broad aims, most notably datasets in this domain are notoriously large (businesses generate terabytes of log data on a daily basis) and are highly heterogeneous (e.g. network flow data versus process logs). This raises questions about how to fuse data sources and how to prioritise identified anomalous behaviours. Cyber-security is a vast and complex area and therefore here, we have narrowed our focus to utilise the skill set of our group and chosen to use familiar methods to conduct initial explorations into this dataset, as outlined in Section 1.5.

1.5 Outline of approaches

This challenge is atypical in the sense it is open-ended. We aim to explore good directions for future development of data science tools towards the big goal of anomaly detection in cyber-security. During the DSG week, we have conducted the following explorations.

- Section 3 discusses visualisation of the data by self-organising maps, a type of unsupervised neural network. We thought that this was an interesting way of presenting the data which may not have been attempted before.

- As mentioned, the large dimensionality of the data poses difficulties for analysis. Therefore, we tackled methods for reducing this dimension via topic modelling in Section 4.1, and also considered how clustering of the data might give direction to the question of how it can be meaningfully fused in Section 4.2.

- Identifying anomalies was explored in two key ways. Section 4.3 considers user–process relationships, working on the premise that users in similar jobs/function would run comparable processes. Secondly, section 4.4 details methods that could allow for ranking of anomalies, including attribute-value frequency (AVF) and formal concept analysis (FCA).
- Another method of identifying unusual network activity is given in Section 4.5. Here the use of Quantile Additive Models (QAMs) is explored.
- The issue of identifying variation in the behaviour of normal activity versus attack, is explored in Section 4.6 via a very different method. Here procrustes anlaysis, a common technique for statistical shape analysis, has been used on adjacency embeddings for NetFlow data.
- Finally, in Section 4.7 we aimed to reconstruct the possible path of the red team attack in the computer network graph. This was so as to retrospectively have a better idea as to how the attack propagated.

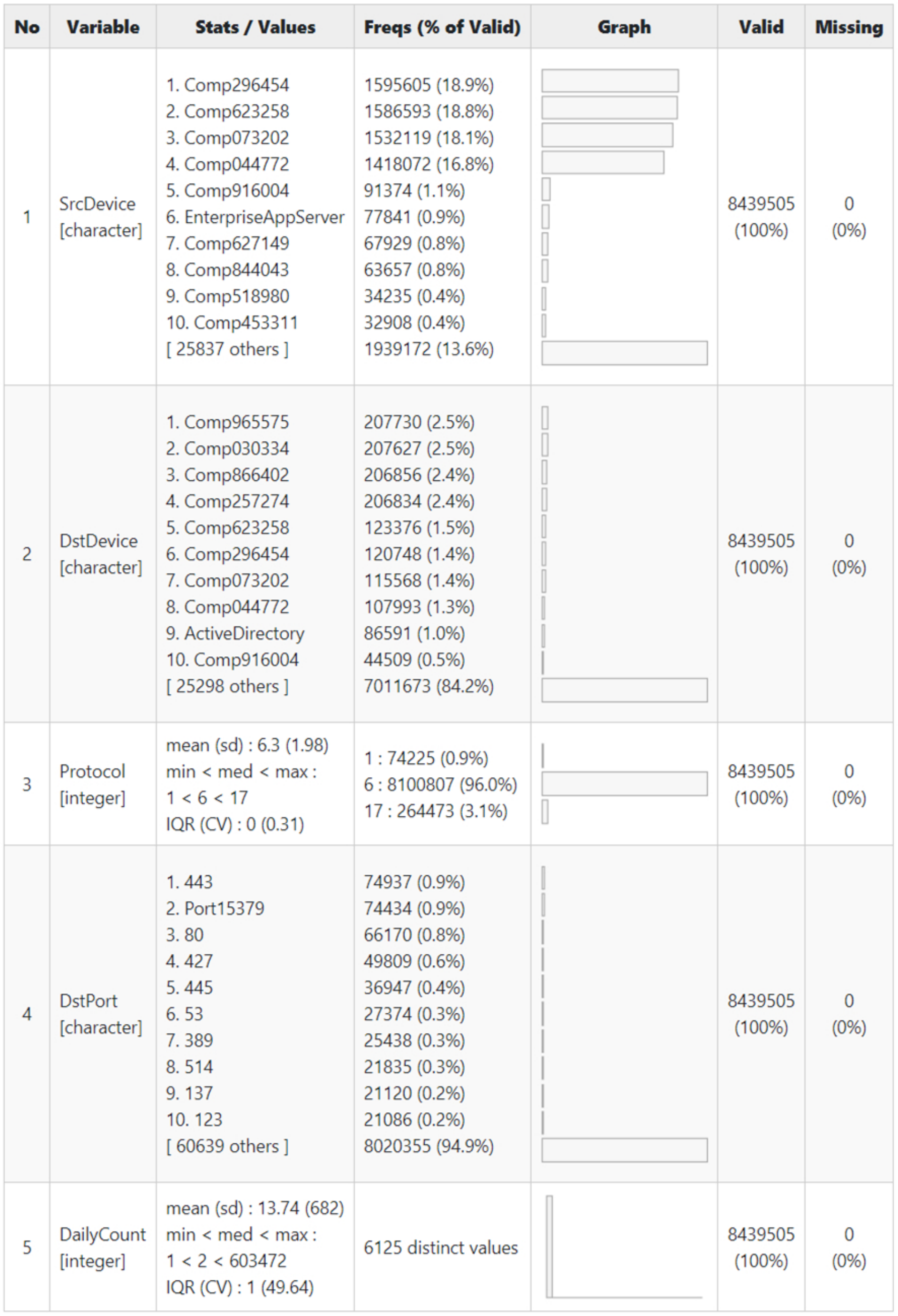
2 Detailed description of the dataset

In addition to the data aggregated per day, described in the previous section, two types of raw data were also provided from LANL, Imperial and Heilbronn for this challenge:

- *Netflow data.* Network event data contains logs of the network flow connections per day between pairs of communicating devices on the network (21,286,000 events per day on average). The raw data consists of NetFlow records exported from core network routers to a centralised collection server, limited to Protocols 1 (ICMP), 6 (TCP), and 17 (UDP). These records were transformed using bi-flowing, obtaining a directed flow. IP addresses and hostnames were mapped, and the failed mappings were anonymised as `IPxyz` rather than `CompXYZ`. A typical NetFlow record contains the following information: time and duration of the connection, source and destination devices, protocol, source and destination ports, bytes and packets sent and received.
- *Windows Host Log data.* The host event data is a subset of host event logs collected from computers running the Microsoft Windows operating system (OS). Examples of such events include successful logons, failed logons, and Windows shutting down, for example. On average there are approximately 177,000 authentication events per day. The host event data further provides process activity information. These have five attributes being *UserName, Device, ProcessName, ParentProcessName, DailyCount.* On average there are approximately 622,000 such events per day in this dataset.

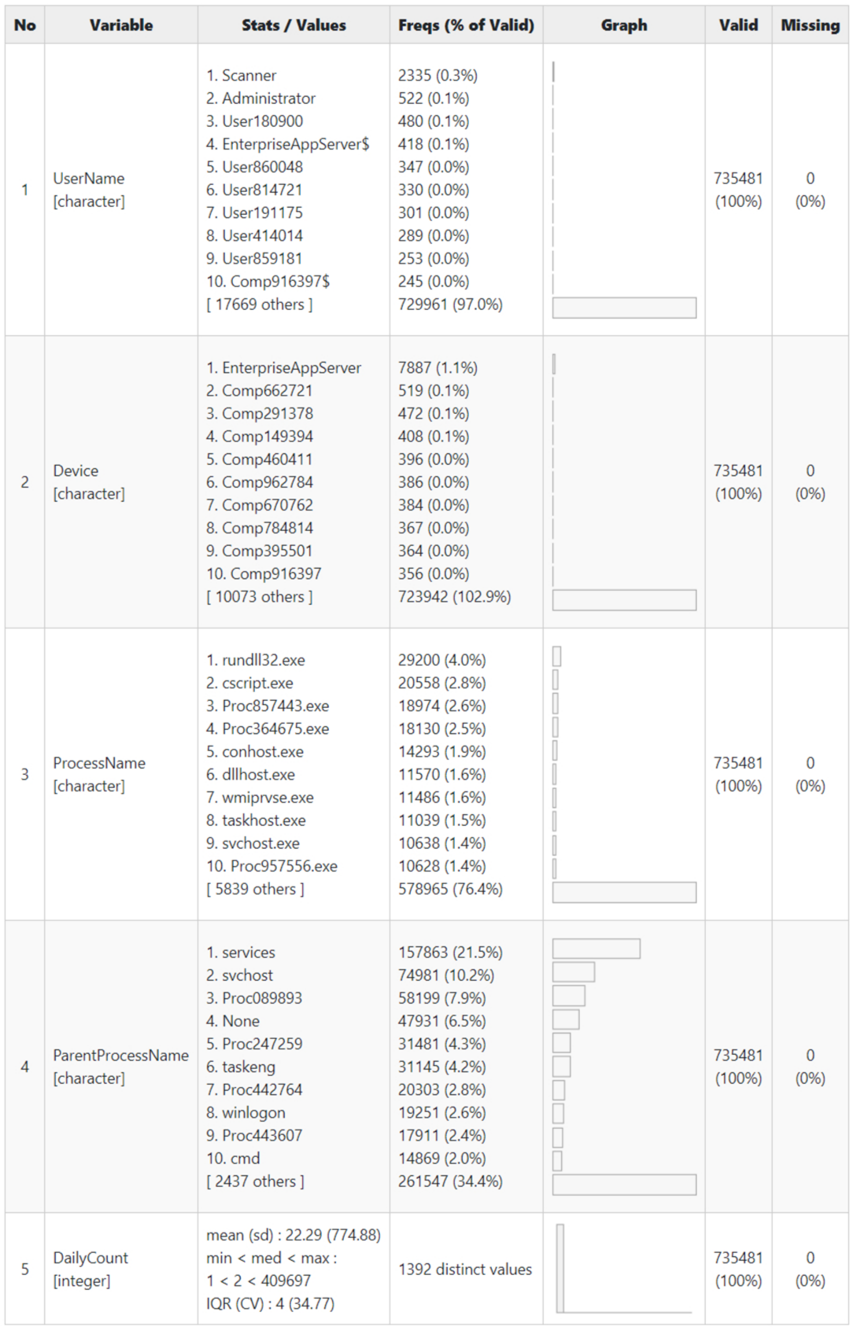
In summary, the following data sources were available:

- Full raw dataset, described above and in [30],
- Three aggregated datasets of daily summaries (`netflow`, `processes` and `authentication`),
- Labelled red-team data consisting of known malicious authentications from stolen credentials, malicious processes, and compromised hosts from Day 57 to 90.



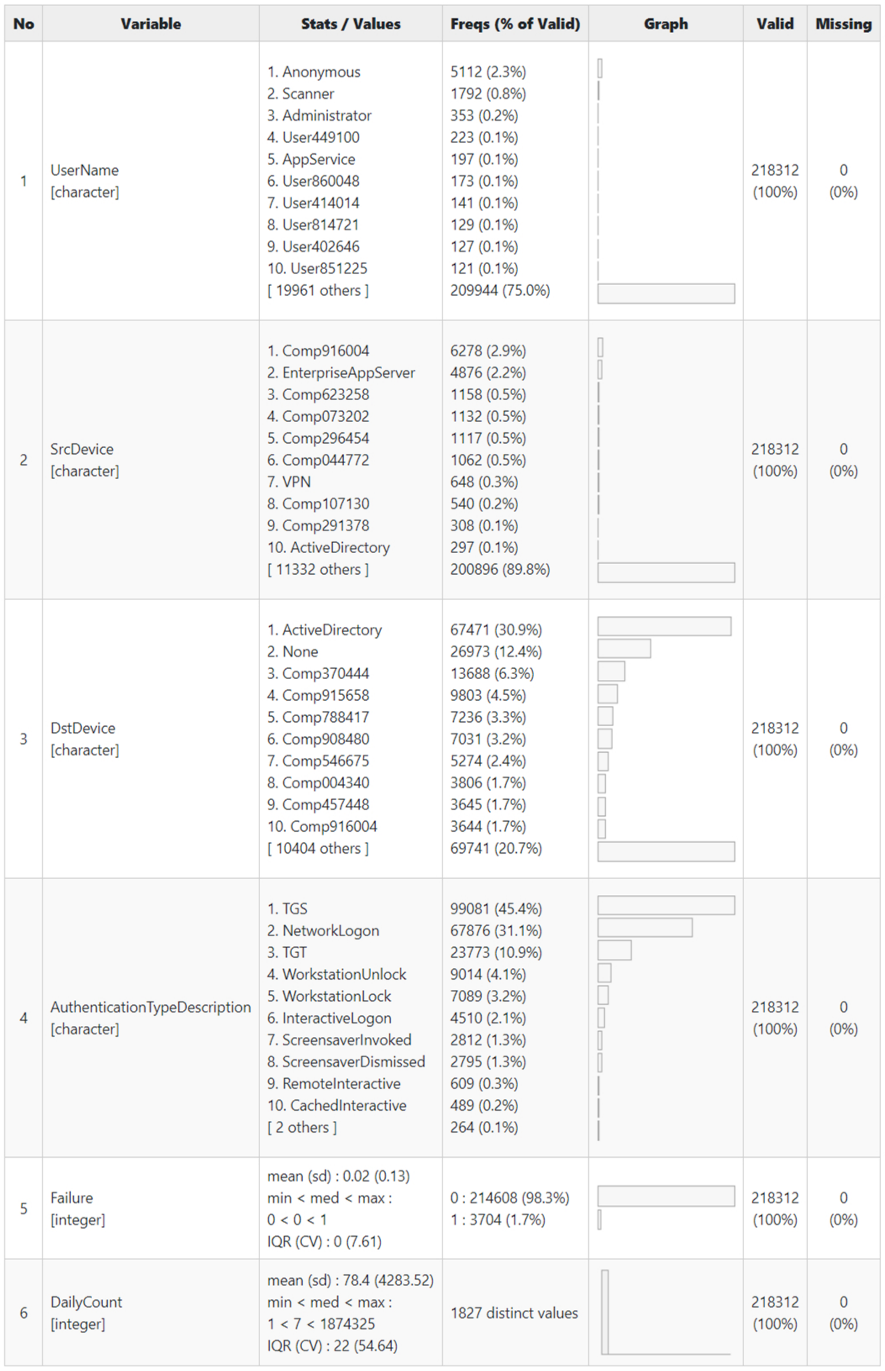
No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	SrcDevice [character]	1. Comp296454 2. Comp623258 3. Comp073202 4. Comp044772 5. Comp916004 6. EnterpriseAppServer 7. Comp627149 8. Comp844043 9. Comp518980 10. Comp453311 [25837 others]	1595605 (18.9%) 1586593 (18.8%) 1532119 (18.1%) 1418072 (16.8%) 91374 (1.1%) 77841 (0.9%) 67929 (0.8%) 63657 (0.8%) 34235 (0.4%) 32908 (0.4%) 1939172 (13.6%)		8439505 (100%)	0 (0%)
2	DstDevice [character]	1. Comp965575 2. Comp030334 3. Comp866402 4. Comp257274 5. Comp623258 6. Comp296454 7. Comp073202 8. Comp044772 9. ActiveDirectory 10. Comp916004 [25298 others]	207730 (2.5%) 207627 (2.5%) 206856 (2.4%) 206834 (2.4%) 123376 (1.5%) 120748 (1.4%) 115568 (1.4%) 107993 (1.3%) 86591 (1.0%) 44509 (0.5%) 7011673 (84.2%)		8439505 (100%)	0 (0%)
3	Protocol [integer]	mean (sd) : 6.3 (1.98) min < med < max : 1 < 6 < 17 IQR (CV) : 0 (0.31)	1 : 74225 (0.9%) 6 : 8100807 (96.0%) 17 : 264473 (3.1%)		8439505 (100%)	0 (0%)
4	DstPort [character]	1. 443 2. Port15379 3. 80 4. 427 5. 445 6. 53 7. 389 8. 514 9. 137 10. 123 [60639 others]	74937 (0.9%) 74434 (0.9%) 66170 (0.8%) 49809 (0.6%) 36947 (0.4%) 27374 (0.3%) 25438 (0.3%) 21835 (0.3%) 21120 (0.2%) 21086 (0.2%) 8020355 (94.9%)		8439505 (100%)	0 (0%)
5	DailyCount [integer]	mean (sd) : 13.74 (682) min < med < max : 1 < 2 < 603472 IQR (CV) : 1 (49.64)	6125 distinct values		8439505 (100%)	0 (0%)

Figure 1: *Example of NetFlow data.*



No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	UserName [character]	1. Scanner 2. Administrator 3. User180900 4. EnterpriseAppServer$ 5. User860048 6. User814721 7. User191175 8. User414014 9. User859181 10. Comp916397$ [17669 others]	2335 (0.3%) 522 (0.1%) 480 (0.1%) 418 (0.1%) 347 (0.0%) 330 (0.0%) 301 (0.0%) 289 (0.0%) 253 (0.0%) 245 (0.0%) 729961 (97.0%)		735481 (100%)	0 (0%)
2	Device [character]	1. EnterpriseAppServer 2. Comp662721 3. Comp291378 4. Comp149394 5. Comp460411 6. Comp962784 7. Comp670762 8. Comp784814 9. Comp395501 10. Comp916397 [10073 others]	7887 (1.1%) 519 (0.1%) 472 (0.1%) 408 (0.1%) 396 (0.0%) 386 (0.0%) 384 (0.0%) 367 (0.0%) 364 (0.0%) 356 (0.0%) 723942 (102.9%)		735481 (100%)	0 (0%)
3	ProcessName [character]	1. rundll32.exe 2. cscript.exe 3. Proc857443.exe 4. Proc364675.exe 5. conhost.exe 6. dllhost.exe 7. wmiprvse.exe 8. taskhost.exe 9. svchost.exe 10. Proc957556.exe [5839 others]	29200 (4.0%) 20558 (2.8%) 18974 (2.6%) 18130 (2.5%) 14293 (1.9%) 11570 (1.6%) 11486 (1.6%) 11039 (1.5%) 10638 (1.4%) 10628 (1.4%) 578965 (76.4%)		735481 (100%)	0 (0%)
4	ParentProcessName [character]	1. services 2. svchost 3. Proc089893 4. None 5. Proc247259 6. taskeng 7. Proc442764 8. winlogon 9. Proc443607 10. cmd [2437 others]	157863 (21.5%) 74981 (10.2%) 58199 (7.9%) 47931 (6.5%) 31481 (4.3%) 31145 (4.2%) 20303 (2.8%) 19251 (2.6%) 17911 (2.4%) 14869 (2.0%) 261547 (34.4%)		735481 (100%)	0 (0%)
5	DailyCount [integer]	mean (sd) : 22.29 (774.88) min < med < max : 1 < 2 < 409697 IQR (CV) : 4 (34.77)	1392 distinct values		735481 (100%)	0 (0%)

Figure 2: *Example of process data.*



No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	UserName [character]	1. Anonymous 2. Scanner 3. Administrator 4. User449100 5. AppService 6. User860048 7. User414014 8. User814721 9. User402646 10. User851225 [19961 others]	5112 (2.3%) 1792 (0.8%) 353 (0.2%) 223 (0.1%) 197 (0.1%) 173 (0.1%) 141 (0.1%) 129 (0.1%) 127 (0.1%) 121 (0.1%) 209944 (75.0%)		218312 (100%)	0 (0%)
2	SrcDevice [character]	1. Comp916004 2. EnterpriseAppServer 3. Comp623258 4. Comp073202 5. Comp296454 6. Comp044772 7. VPN 8. Comp107130 9. Comp291378 10. ActiveDirectory [11332 others]	6278 (2.9%) 4876 (2.2%) 1158 (0.5%) 1132 (0.5%) 1117 (0.5%) 1062 (0.5%) 648 (0.3%) 540 (0.2%) 308 (0.1%) 297 (0.1%) 200896 (89.8%)		218312 (100%)	0 (0%)
3	DstDevice [character]	1. ActiveDirectory 2. None 3. Comp370444 4. Comp915658 5. Comp788417 6. Comp908480 7. Comp546675 8. Comp004340 9. Comp457448 10. Comp916004 [10404 others]	67471 (30.9%) 26973 (12.4%) 13688 (6.3%) 9803 (4.5%) 7236 (3.3%) 7031 (3.2%) 5274 (2.4%) 3806 (1.7%) 3645 (1.7%) 3644 (1.7%) 69741 (20.7%)		218312 (100%)	0 (0%)
4	AuthenticationTypeDescription [character]	1. TGS 2. NetworkLogon 3. TGT 4. WorkstationUnlock 5. WorkstationLock 6. InteractiveLogon 7. ScreensaverInvoked 8. ScreensaverDismissed 9. RemoteInteractive 10. CachedInteractive [2 others]	99081 (45.4%) 67876 (31.1%) 23773 (10.9%) 9014 (4.1%) 7089 (3.2%) 4510 (2.1%) 2812 (1.3%) 2795 (1.3%) 609 (0.3%) 489 (0.2%) 264 (0.1%)		218312 (100%)	0 (0%)
5	Failure [integer]	mean (sd) : 0.02 (0.13) min < med < max : 0 < 0 < 1 IQR (CV) : 0 (7.61)	0 : 214608 (98.3%) 1 : 3704 (1.7%)		218312 (100%)	0 (0%)
6	DailyCount [integer]	mean (sd) : 78.4 (4283.52) min < med < max : 1 < 7 < 1874325 IQR (CV) : 22 (54.64)	1827 distinct values		218312 (100%)	0 (0%)

Figure 3: *Example of authentication data.*

3 Data visualisation

Data visualisation, when used effectively, can be a powerful tool for uncovering patterns and structure in complex data. The standard approach of graphically representing numerical data is useful for identifying trends and seasonalities or for exposing relationships. However, this standard approach has limited applicability when dealing with high-dimensional, temporal data, such as that generated within an enterprise network.

3.1 Self-organising maps

A self-organising map (SOM) [19] is a type of unsupervised neural network that produces a low-dimensional representation of an input domain. Building on Alan Turing's model of morphogenesis [31], SOMs use a neighbourhood function to preserve the topological properties of the input space, providing a useful framework for data visualisation.

Consider a dataset X, containing N p-dimensional samples, $\boldsymbol{x}_i = \{x_{i1}, \ldots, x_{ip}\}$. An SOM is an ordered collection of J neurons, each with an associated reference vector $\boldsymbol{w}_j = \{w_{j1}, \ldots, w_{jp}\}$. To train an SOM, each $\boldsymbol{w}_j$ is initially assigned a random vector from the domain of X. When a sample $\boldsymbol{x}_i \in X$ is presented to the SOM for training, the neuron whose reference vector has the smallest distance from $\boldsymbol{x}_i$ is identified as the best matching unit (BMU) for that input:

$$b_i = \underset{j=1,\ldots,J}{\arg\min} \left\| \boldsymbol{x}_i - \boldsymbol{w}_j \right\| .$$

The reference vector of the BMU and those of the neurons close to it in the SOM grid are adjusted towards the input vector. The magnitude of the change decreases with time and with the grid-distance from the BMU. The update formula at iteration t for neuron j with reference vector $\boldsymbol{w}_j(t)$ is:

$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + \alpha(t)\phi(b_i, j, t) \left\| \boldsymbol{x}_i(t) - \boldsymbol{m}_j(t) \right\| ,$$

where $\alpha(t)$ is a monotonically decreasing learning rate and $\phi(b_i, j, t)$ is a neighbourhood function centred at the BMU b_i.

The neurons in a trained SOM are not equally distributed among the entire input space; rather, more neurons are designated for regions with more

Table 1: *NetFlow features constructed for each time bin.*

Name	Description
TotalIn	Total no. of inward communications
TotalOut	Total no. of outward communications
InDegree	No. of unique inward communications
OutDegree	No. of unique outward communications
BytesIn	Total no. of bytes received
BytesOut	Total no. of bytes sent

samples in X (high density) and fewer ones for lower density regions in X. This may be visualised using a 3-dimensional representation of the *unified distance matrix* (U-matrix) – a matrix of Euclidean distances between reference vectors of neighbouring cells of the SOM. Cluster borders are then indicated as 'mountains' of high distances separating 'valleys' of similar neurons.

3.2 Experiments

Enterprise network data is temporal in nature and tends to follow a daily seasonal pattern. In this section we describe an approach to visualise the overall state of the network at a given time, in order to highlight deviations from normal operating state. [24] use SOMs to depict multistate process operations in an industrial chemical plant. An enterprise network can similarly be considered to operate in different states, based on user activity at a given time of day.

For our experiment, we chose a set of 10 computers, 5 of which were known compromised hosts where the red team had command and control. Each day of NetFlow data was divided into 1-hour bins and a set of 6 features computed for each computer within each bin. These are described in Table 1. Each of these 60 variables were first normalised and an SOM was trained using days 52–55, resulting in 96 samples. The resulting U-matrix is shown in Figure 4.

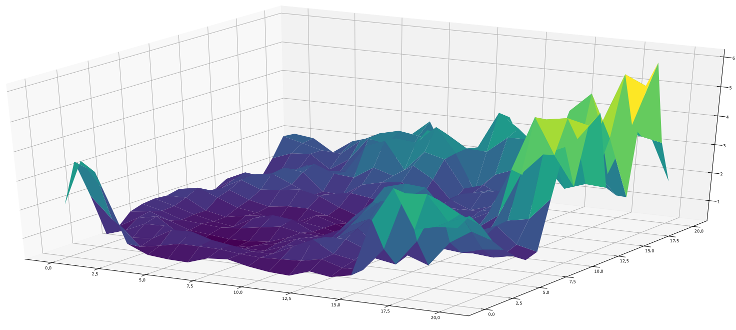


Figure 4: *A 3-dimensional visualisation of the U-matrix.*

We then tested the model using data from days 56 and 58 in order to compare the corresponding mappings. Day 56 provides us with input vectors representing activity prior to the red team's attack, whilst on day 58 the network is known to have been compromised. Visualisation of this mapping transitioning over the course of a day has been given as a 'trajectory' through the 3-dimensional distance space, and is presented in Figures 5 and 6. As is illustrated, when testing the model on data from day 58, over the course of the day the process 'traverses' through more regions that represent low density in feature space than on day 56, prior to the red team's attack. In particular, we see that during hours 4, 5, and 20–23, the BMUs lie within these 'mountainous' regions. In this way, this analysis explores an interesting and viable method for exposing potentially suspicious phases based on the input space of features.

For this simple example, we only considered features derived from NetFlow activity. However, the feature space explored could be constructed using data from a combination of sources. In this way, our approach serves as a very promising basis for visualising high-dimensional multi-source data.

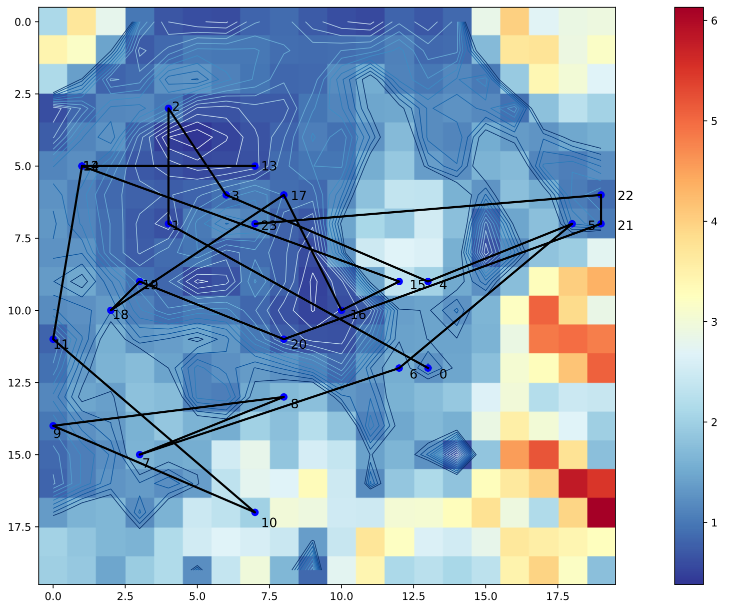


Figure 5: *A contour plot of the U-matrix with the state BMUs for day 56.*

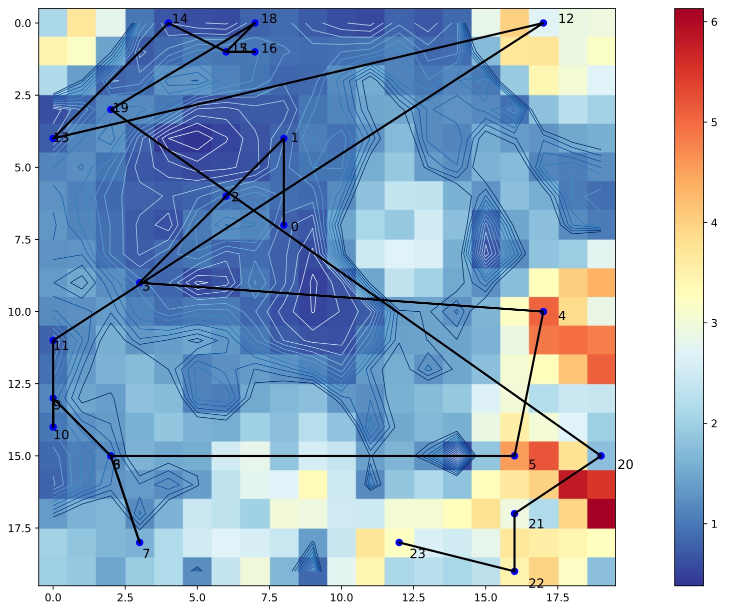


Figure 6: *A contour plot of the U-matrix with the state BMUs for day 58.*

4 Analysis

4.1 Topic models of process data

In this section, we extract structure from Windows event log process data. Each record in the data consists of a tuple:

(date, user, device, process, parent process, count),

where ‘count’ refers to the number of times that a process with the given parent process was started on the given device, user and date. Learning methods often do not work well for high dimensional categorical data and it is therefore worth applying methods to reduce this.

Topic models are a widely used tool for grouping together high dimensional categorical variables [3]. Originally developed for text analysis, they can be adapted to the cyber context. Here, we illustrate this by creating a low dimensional embedding of devices based on the processes ran, and of users based on the devices used.

4.1.1 LDA Model: Device embeddings

Here, we focus only on the tuple of device, process and count. To make the analogy to topic modelling, we treat each device (on a single day) as a ‘document’ and each process as a ‘word’, with the number of times that the process runs as the document word count. We will then fit a Latent Dirichlet Allocation (LDA) model. The outputs of model are the following.

- *Topics.* Each topic is a distribution over processes (words) represented as a P-vector where P is the number of possible processes. Each topic represents a grouping of processes and thus we interpret topics as a proxy for different activities that a device can engage in.
- *Distribution over Topics.* For each device (document), the model decomposes that document into a mixture of topics and provides us with the mixing weights. So, for each day, we obtain a new topic distribution per device reflecting the activity of that device for that day. If the usage/function of the device remains relatively stationary,

then the topic distributions associated with a device should also be relatively constant and in fact this is what we observe in the data. Most machines are peaked around a few topics that captures their daily activity as shown in Figure 7).

4.1.2 Details on Model Training

We trained our topic model on process data from days 52 to 90. We made the following choices regarding data processing and model tuning:

- For each process, we calculated its frequency defined as:

$$F = \frac{\sum_{d=1}^{90} \#\{\text{distinct devices running process on day } d\}}{\sum_{d=1}^{90}\{\text{distinct devices with activity on day } d\}}$$

  . We excluded processes with frequency greater than 0.4 as we are primarily interested in those processes that help us to distinguish between devices.

- As standard, we performed term frequency inverse document frequency (TFIDF) transformation:

$$C_j \cdot \log F_j^{-1},$$

  where C_j is the count for the jth process and F_j is the process frequency as defined in the previous bullet. Again, our motivation is to up-weight those processes that help us to distinguish between devices.

- We trained the model in Python using the `gensim` package. We chose $K = 20$ topics and default settings were used for all other parameters in gensim's implementation.

4.1.3 Results

The 15 panels in Figures 7 and 8 plot the topic distributions for 15 randomly sampled devices, from the set of devices which ran at least one malicious process during red team activity. Each blue line represents the topic distribution associated with process activity on a normal day.

We summarise the conclusions that can be drawn from Figures 7, 8.

- Most devices have topic distributions for daily activity that concentrates around a few topics (these represent the ‘canonical’ activities for which the device is used).
- Malicious processes are compactly captured by a few topics. This was accomplished in an unsupervised manner without labels for malicious activity. This shows that unsupervised learning can potentially be a powerful for creating features/representations that are then used in anomaly detection.

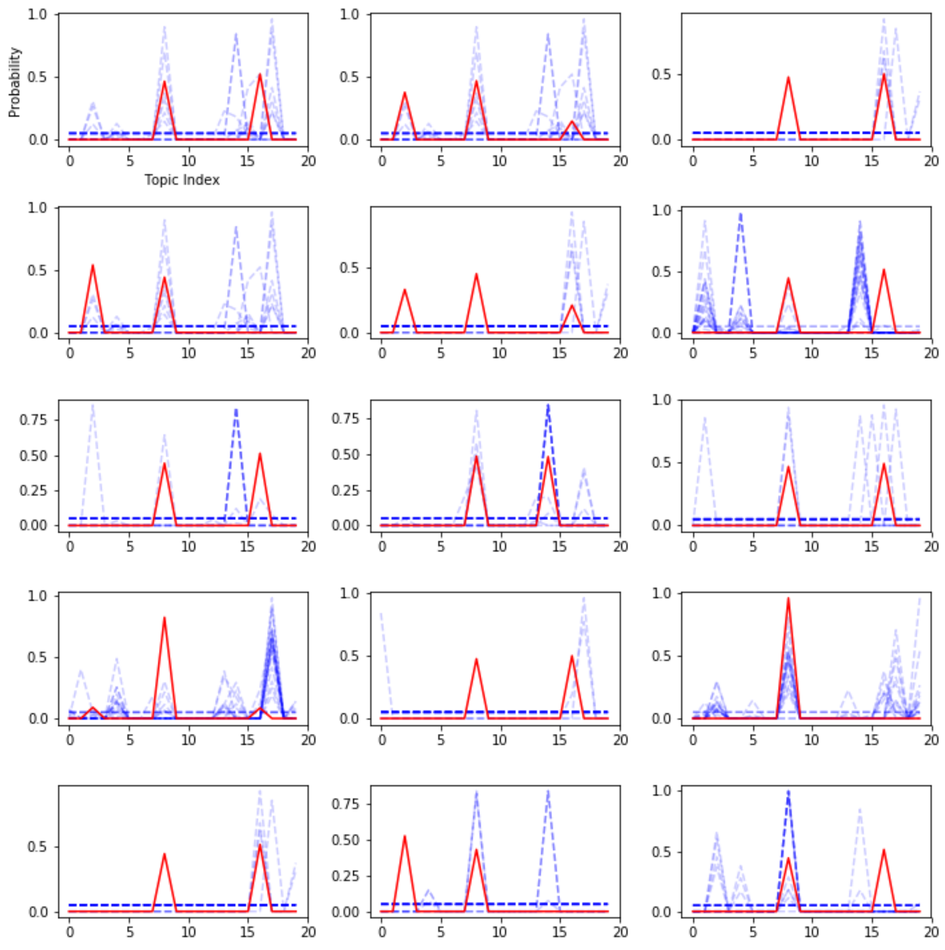


Figure 7: *Topic distributions associated with 15 randomly sampled devices. Each blue line represents the topic distribution for a single day. Most devices concentrate around only a few topics reflecting the fact that devices have fixed usage. The red line represents the topic distribution associated with the malicious processes running on the device (with all other non-malicious background processes removed). The figure shows that topics 2, 8 and 16 capture malicious processes.*

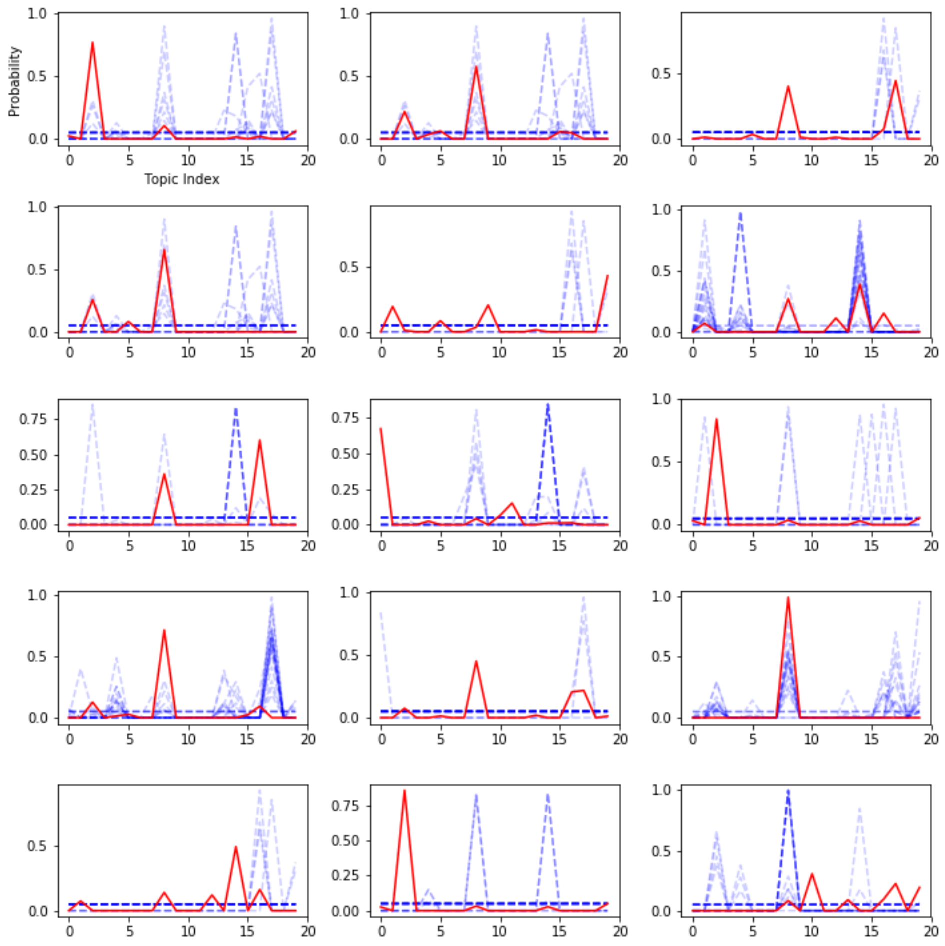
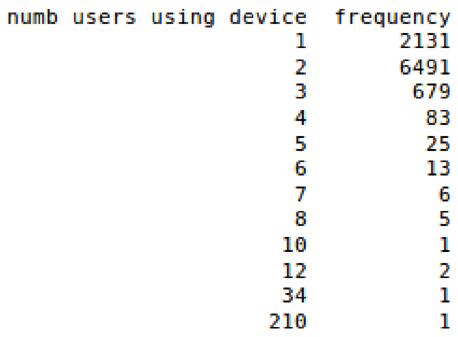


Figure 8: *Topic distributions associated with 15 randomly sampled devices. Blue lines are for activity during normal days. Red line is activity during a day with malicious process.*

4.1.4 LDA Model: User embeddings

Using the same approach as described above, we separated users based on device usage. In this case, users represent the documents and devices the words. When looking at the number of users that use a particular device on a particular day, we find the following:



```
numb users using device  frequency
                      1       2131
                      2       6491
                      3        679
                      4         83
                      5         25
                      6         13
                      7          6
                      8          5
                     10          1
                     12          2
                     34          1
                    210          1
```

Figure 9: *For each device, we find how many users used that particular device on a particular day (a random day was chosen from the summary data). The table shown here tells us how many devices had a particular number of users. For example, there were 2131 devices on this particular day which were used by exactly 1 user, and 6491 devices were used by exactly 2 users.*

We observe that, for the particular day investigated, about 23% of all devices are only used by a single user and almost 70% of devices were used by either one or two users. Similar results were obtained for other days. We may want to collapse all devices used by a single user into one 'super device' which is likely to represent a personal computer. Doing this will significantly reduce the number of words used in our documents by grouping together words of similar type. Using this reduced device space, we can use an LDA model to find topic distributions (where each topic is now a distribution over devices) for each user (Figure 10). We observe from Figure 10 that most users show very little change in their topic distribution as the days pass. This may not be surprising, as many users will use their own personal computer every day and may not start processes on other devices. As before, we also visualised red team activity, however there was little difference between compromised and non-compromised distributions.

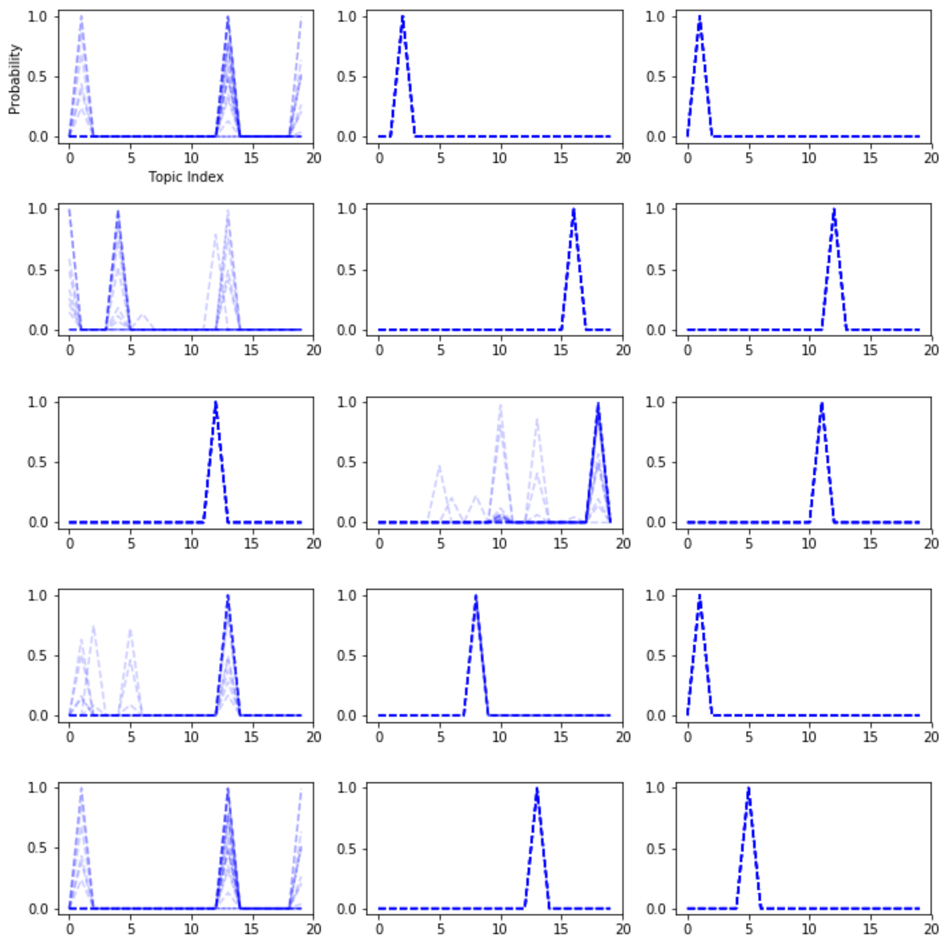


Figure 10: *Similar to Figures 7 and 8, each subplot represents a single randomly sampled 'document' (in this case a user) and each blue dotted line within the subplot represents the topic distribution for a single day without red team activity. Most users are very concentrated around a single topic and their topic distribution is very similar over different days.*

4.1.5 Conclusions

Topic models are a useful tool for performing dimension reduction when faced with discrete variables with a large number of categories. They are trained very quickly using stochastic approximation variational inference. The work above is a preliminary investigation and we expect improved results by adopting the following ideas:

- *Implement device or user specific topic models.* For example, each device may have its own pattern of variation in regards to process activity. However, there may not be sufficient sample size per device for doing so which makes the use of hierarchical LDA models potentially useful.

- *Tuning model performance.* Key parameters include frequency TFIDF weighting, and number of topics. In particular, the current model used only 20 topics. This means that malicious processes will with high probability need to share a topic with non-malicious processes. With a larger number of topics, we may be able to more cleanly separate malicious and non-malicious processes. We have some preliminary evidence that malicious processes can indeed be identified more accurately using 30 topics rather than 20.

- *Quantitative measures.* Up until now, we have used visual inspection to check for anomalies in process activity (as represented by topic distribution). The development of test statistics and rigorous inference would be of value.

4.2 Clustering based cross-domain data fusion for anomalous activity detection

In this approach, we look into other clustering venues that we intend to use for data fusion. Due to limited time we did not have results for data fusion so we describe the logistic briefly here. We assume that each user belongs to a group with certain set of behaviour patterns. We can consider the group that each user belongs to as a latent variable and use the clustering of user activities within each domain to model the hidden group belonging index. Assuming cross-domain consistency, we consider a user an anomaly if it exhibit changes in behaviour that is dissimilar from its group or behave like another group that it does not belong to. A similar method is used by [8] for anomaly detection on practice data and successfully detected all (artificially injected) anomaly by scanning only 50% of the data.

4.2.1 Event mixing/pairing of processes-ports/destinations

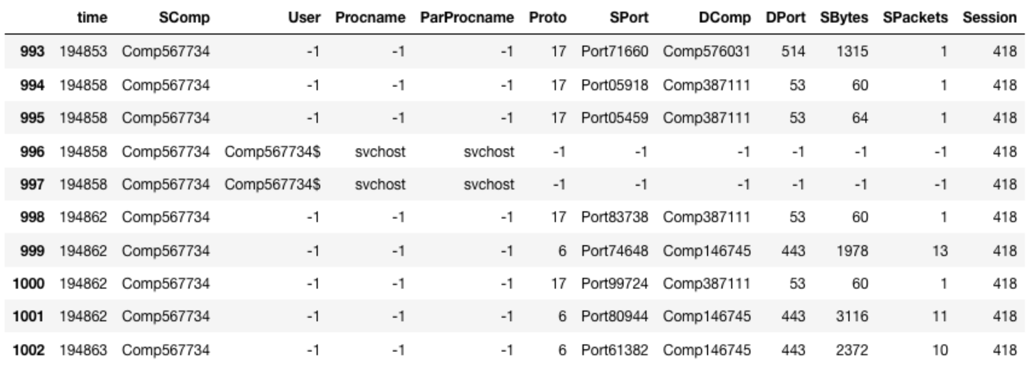
The dataset we are looking at contains three different types event-streams: network traffic summaries (given as events in the Netflow format), summaries of authentication events, and process start activity on Windows machines. All three types of events can be associated with the machine they originated or were directed to. Furthermore, all events are discrete in time and contain a corresponding time-stamp. Despite describing different things, different types of events associated with one machine are not necessarily independent in nature. Instead, it is quite likely that individual events in one stream can trigger events in another, such as the arrival of network information being responsible for the start of a particular process or vice versa. In this analysis, we focus on a method that could mine strong associations of events in different data streams.

As we are looking at events on individual machines, the first thing to do is gather all events associated with that particular machine into on stream. For that, we select a subset of 13 Windows personal computers being active in the network (identified as `Comp866344, Comp531955, Comp032933, Comp567734, Comp912319, Comp762316, Comp185550, Comp653812, Comp882990, Comp709308, Comp564857, Comp152636`, and `Comp643355`). We then filtered the data different data streams for events associated with these machines (by being the LogHost for the authentication and process events, or by being the source or destination machine for Netflow events), and merged them into 13 unified data streams containing all three types of events. In order to process the data in a meaningful way, the events have to be sorted according to their respective time-stamp.

Previous work, [26], has shown that network flow events are not uniformly spread in time, but instead occur in small groups of events. It is not unreasonable to assume the same about process events and authentication events. In this work, we will define an event group, called a *session*, as set of events where each member is separated in time from at least one other member by less c seconds, where c is a chosen threshold. In other words, if the time between event A and its predecessor B is less than c, A is added to the existing session. If not, A marks the start of a new session.

Previous analysis has shown that 95% of network flow events are separated by their nearest neighbour in time by less than 9 seconds. We therefore chose $c = 9$ seconds as a reasonable threshold.

We then proceed to create sessions in the described fashion for each machine's unified data stream, and label all events with the corresponding session they belong to. Figure 11 depicts a sample session of events with their respective properties.



	time	SComp	User	Procname	ParProcname	Proto	SPort	DComp	DPort	SBytes	SPackets	Session
993	194853	Comp567734	-1	-1	-1	17	Port71660	Comp576031	514	1315	1	418
994	194858	Comp567734	-1	-1	-1	17	Port05918	Comp387111	53	60	1	418
995	194858	Comp567734	-1	-1	-1	17	Port05459	Comp387111	53	64	1	418
996	194858	Comp567734	Comp567734$	svchost	svchost	-1	-1	-1	-1	-1	-1	418
997	194858	Comp567734	Comp567734$	svchost	svchost	-1	-1	-1	-1	-1	-1	418
998	194862	Comp567734	-1	-1	-1	17	Port83738	Comp387111	53	60	1	418
999	194862	Comp567734	-1	-1	-1	6	Port74648	Comp146745	443	1978	13	418
1000	194862	Comp567734	-1	-1	-1	17	Port99724	Comp387111	53	60	1	418
1001	194862	Comp567734	-1	-1	-1	6	Port80944	Comp146745	443	3116	11	418
1002	194863	Comp567734	-1	-1	-1	6	Port61382	Comp146745	443	2372	10	418

Figure 11: *Sample session, group of traffic events on machine* `Comp567734` *with corresponding event properties. As visible, no event is separated from its neighbours by more than 9 seconds.*

Session lengths can vary, stretching from just one event to more than a thousand. Figure 12 is a depiction of the session length distribution for the examined machines. The distribution appears to be more or less stable across the machines, with shorter sessions of just a few events dominating.

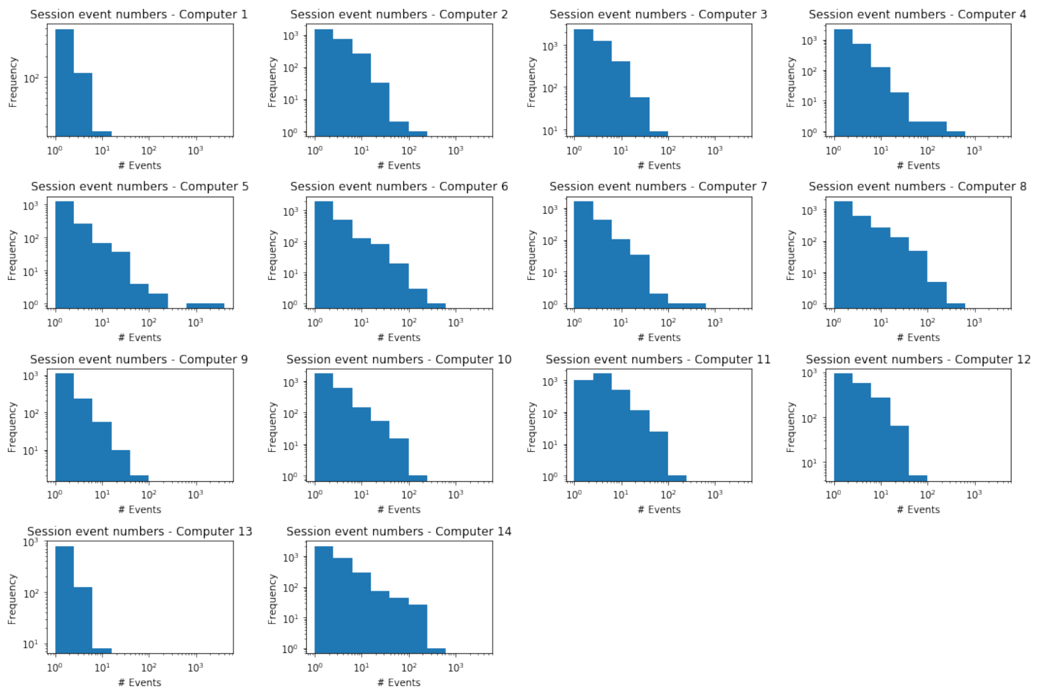


Figure 12: *Distribution of session lengths for the examined computers.*

We have now grouped the data temporally into narrow intervals. As we want to examine the relationship between different types of events, we will in particular look at sessions that contain more than one type of events. The temporal spread of individual sessions is small compared to eventless periods on a typical machine. Therefore, independent events are unlikely to fall into the same session. If events from different event streams are strictly independent from another, we would expect very little mixing inside individual sessions, i.e. the overwhelming majority of sessions would consist exclusively of Netflow, authentication, or process events.

While creating the session labels, we recorded whether more than one type (and if so which) of event was present in each session. Figure 13 compares the number of single event-type sessions to those with more than one type of events.

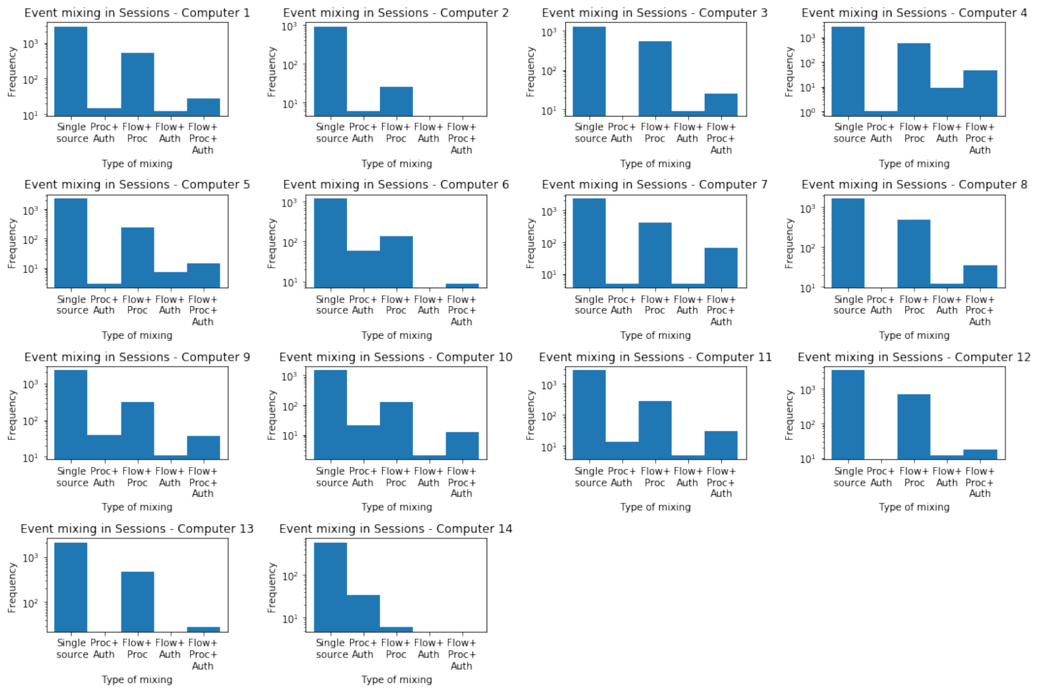


Figure 13: *Mixing distribution of sessions for individual machines.*

As expected, most sessions only contain one type of events. However, we see that sessions that contain both Netflow and process events are occurring very frequently as well, with a stable distribution across almost all computers. Furthermore, events mixing all types of events also occur regularly on most machines. Due to time constraints, we will concentrate in this work on the relationship between network flow events and process events.

After examination of a few mixed Netflow/process sessions, there seems to be some consistency in the specific processes and the corresponding destination computers and network ports that are occurring together. In other words, we see a lot of mixed sessions that contain both the start of a specific process and the subsequent contact of one or more specific computers on a specific network port. We are therefore interested if we can examine the frequency with these specific events are happening and compare them with the number of sessions where these events are not mixed.

In particular, we are looking at three different sets of event features that occur in mixed sessions:

- The name of the started process.
- The destination computer contacted.
- The destination port used for a connection.

For each machine, we gather the set of all specific process names, destination computers, and destination ports that can be observed within sessions containing both network flows and process starts. We will in the future call these sessions ProcFlow-sessions. We will call the set of observed process names P, the set of destination computers DC, and the set of destination ports DP.

For each member of the three sets, we calculate the overall occurrence number, and the percentage α of these events occurring in ProcFlow-sessions. We then proceed to calculate the pairwise percentage with which members of P are occurring together with members of DC and with members of DP. To be precise, for $p \in P$ and $b \in DC$ or DP, we calculate the number of times in which p can be observed in a session with b, and divide it by the total number of times p is observed in a machines data stream. This can indicate whether p is only or mostly occurring together with b, and consequently if there exists a generative association between these two events. As appears more likely that a process is responsible for traffic generation compared to network events generating a process start, we looked at this relationship. However, it is definitely worth investigating the opposite relationship, however time constraints prevented us from doing so.

In principle, the number of pairs can grow quadratically as we observe more and more members in P and DC/DP the longer if we observe a machine. We therefore need to discard some less informative members. Here, we discard members that see less than 10 events in total.

Figures 14 to 15 depict the occurrence percentages for a number of pairs machine `Comp866344`.

```
'svchost with 53 : 0.4506024096385542% (187.0 times)',
'taskeng with 53 : 0.6% (15.0 times)',
'services with 53 : 0.6206896551724138% (54.0 times)',
'services.exe with 53 : 0.7037037037037037% (19.0 times)',
'services.exe with 514 : 0.9629629629629629% (26.0 times)',
'services with 514 : 0.9885057471264368% (86.0 times)',
'Proc247259 with 514 : 1.0% (25.0 times)',
'csrss with 514 : 1.0% (54.0 times)',
'taskeng with 514 : 1.0% (25.0 times)',
'svchost with 514 : 0.9975903614457832% (414.0 times)']
```

Figure 14: *Percentages of processes occurring together with specific ports for machine* `Comp866344`.

```
'taskeng with Comp275646 : 0.6% (15.0)',
'services with Comp275646 : 0.6206896551724138% (54.0)',
'services.exe with Comp387111 : 0.6666666666666666% (18.0)',
'services.exe with Comp275646 : 0.7037037037037037% (19.0)',
'services.exe with Comp576031 : 0.9629629629629629% (26.0)',
'services with Comp576031 : 0.9885057471264368% (86.0)',
'svchost with Comp576031 : 0.9975903614457832% (414.0)',
'taskeng with Comp576031 : 1.0% (25.0)',
'csrss with Comp576031 : 1.0% (54.0)',
'Proc247259 with Comp576031 : 1.0% (25.0)']
```

Figure 15: *Percentages of processes occurring together with flows specific destination computers for machine* `Comp866344`.

We can observe very high association between a number of services and port/computers, for instance we observed 414 sessions where the *svhost*-process occurs together with a flow using port 514 and contacting `Comp576031`, while this process occurs virtually never (only 1 time) in other sessions. Specifically, the association between processes and destination ports can be observed over multiple machines while the contacted computers are changing. This (and other combinations) can also be observed on other computers. However, some computers (for instance `Comp567734`) show a lot less clear associations, with the strongest ones only occurring 60% of the time together.

4.2.2 Application to anomaly detection

Mining event associations between between different data instances help to understand the structure of traffic generated by one machine. Very tight associations can tie a specific type of network traffic to specific processes (and possibly authentication events), and indicate anomalous and potentially malicious behaviour if we observe traffic going from or to that machine without the associated process. For example, if we can associate traffic to a specific destination computer in the network completely with a set of processes that generate this type of traffic, a malicious software or exploit that generates this traffic independently from these processes, we have a tool to observe this as anomalous. As we were mining association of events from different data streams, it is very hard for an attacker to fake these associations for fly under the radar as he would need the ability to generate spoof events on multiple channels. Future approaches can merge several pairwise association together in a similar manner as proposed above to create even stronger associations. Furthermore, associations that are not 100% tight can be used to give an probabilistic estimate how frequently events appear without their corresponding partner event.

4.2.3 K-means clustering of NetFlow Data, Authentication and Process Data

Although we started with the intention to explore data fusion, the actual question we were able to address is more *How can we use unsupervised techniques to construct latent variable for identify anomalies in NetFlow and Windows event logs?* In this subsection, we present our analysis that suggests using K-Means clustering on Process and Authentication data.

We try K-means in clustering for the first stage of data fusion motivated by the scalability. We first explore K-means for NetFlow data only and then on Process and Authentication data. We manually scanned the NetFlow summary datasets and observe there are several source device and destination device entries that use a particular protocol number. Leveraging on this observation we develop a process to compress the number of entries in the NetFlow summary datasets. This process is given as follows:

- Split the attributes into two groups. Group 1 contains attributes Source Device, Protocol and Daily Count. Group 2 contains attributes Destination Device, Protocol and Daily Count.

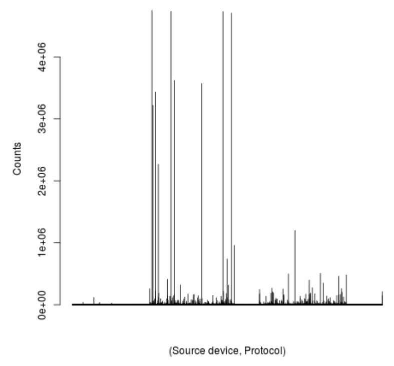
- Aggregate daily count by Source Device and Protocol for group 1.
- Aggregate daily count by Destination Device and Protocol for group 2.

Day 2			**Day 3**		
Total Entries	Group 1 Entries	Group 2 Entries	Total Entries	Group 1 Entries	Group 2 Entries
8,439,505	58,100	37,446	21,435,961	61,757	170,558

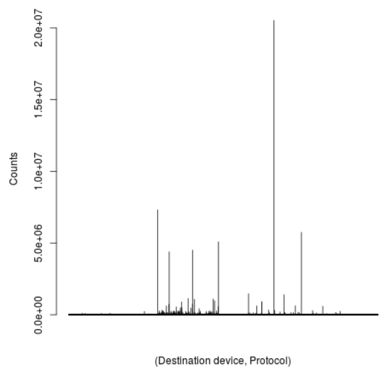
Table 2: *Aggregated NetFlow data.*

From Table 2, we observe that the number of entries in the NetFlow summary datasets is reduced by 99% for Group 1 and Group 2. For day 2, we observe there are more entries in Group 1 and fewer entries in Group 2 whilst for day 3, we observe fewer entries in Group 1 and more entries in Group 2.

Figure 16 shows the aggregated daily counts for Group 1 and Group 2 for day 2. We observe that only a small number of both source device and protocol pairs, and destination device and protocol pairs, have large aggregated daily counts. Looking into the aggregated data we find `EnterpriseAppServer`, `ActiveDirectory`, `VPN` and various devices have large aggregated daily counts and these devices use the TCP protocol (protocol number 6) and UDP protocol (protocol number 17).



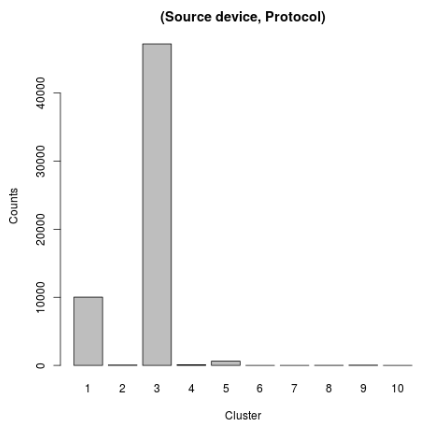
(a) Day 2: Source Device and Protocol.



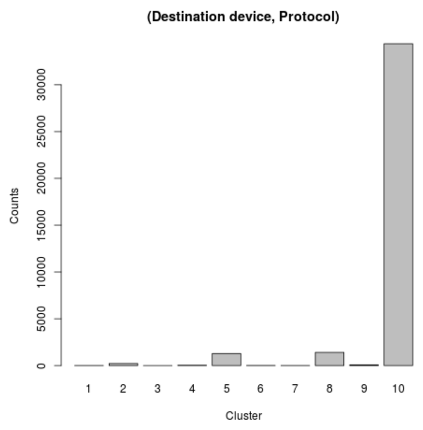
(b) Day 2: Destination Device and Protocol.

Figure 16: *Aggregated daily counts.*

To obtain the clusters, we apply K-means clustering method and set the initial number of clusters to 10. Figure 17 shows the size of each cluster. We adopt the assumption that normal instances belong to big and dense clusters, while anomalies belong to small clusters [4].



(a) Day 2: Source Device and Protocol.



(b) Day 2: Destination Device and Protocol.

Figure 17: *Cluster sizes.*

SrcDevice	DstDevice	Protocol	Port	Daily count
Comp251637	Comp708267	6	80	315
Comp251637	Comp186884	6	443	574
Comp251637	Comp282947	6	80	1
Comp251637	Comp186884	6	80	2
Comp251637	Comp282947	6	443	1
Comp251637	Comp073658	6	Port85926	22
Comp251637	Comp326103	6	Port88837	15
Comp251637	Comp364152	6	445	142
Comp251637	Comp326103	6	Port01352	163

Table 3: *Anomalous communication path.*

We map the source device in the small cluster to the source device in the NetFlow summary dataset to obtain the communication paths from the source device to the destination device. We identified 12 different mappings of anomalous source device to destination devices in three of the smallest clusters on day 2. The communication path between an anomalous source device and destination devices are given in Table 3.

4.2.4 Conclusion

Anomalous communication paths are identified by using the K-Means clustering method due to time constraints. Further anomalous communication paths can be identified by applying other unsupervised techniques. As future work, we plan to apply hierarchical clustering methods to identify new anomalous communication paths. Compared with the analysis of NetFlow data, we do clustering analysis with authentication and process data on summary data without doing aggregation. The intention for this is to understand unusual patterns in user level. More importantly, the Authentication and Process data have certain amount of distinctive user IDs, with which we can implement data fusion through cross domain comparison and anomaly score computing. Due to the time limitation, we adopted a simple algorithm to measure anomalous patterns across domains, replicating the original method outlined in [8] is highly recommended due to the great performance on the Carnegie Mellon data.

4.3 Anomaly detection in user-process bipartite graphs

In this study, we leverage the user–process relationship to find compromised user accounts. We make use of the process log summary data, which contains the set of processes initiated by each user. Our proposed model can be deployed in real time and returns a list of suspicious users for investigation.

4.3.1 Methods

We represent the processes that users run as a bipartite graph, G, such that the nodes are divided into two groups and no edge connects vertices in the same group. More formally, let V_1 be the set of processes and V_2 the set of users. The bipartite graph G is defined as $G = V_1 \cup V_2$, where $V_1 = \{p_i | 1 \leq i \leq k\}$ and $V_2 = \{u_i | 1 \leq i \leq n\}$. An edge e connects a process p and a user u, if the user u ran process p. The relationship between users and processes is represented by a binary adjacency matrix. We expect that users in similar job functions or departments use many of the same processes. For example, a finance account manager will rarely run Android programming frameworks. Our aim is to score processes on a measure of 'similarity', and use this metric to identify users that are anomalous with respect to the diversity of processes they use.

Our approach adapts the framework described in [27] and considers logs over a 7-day moving time window. We set $e_{ij} = 1$ for process i and user j if the user ran process j at any time during the 7-day period. First, we compute the relevance, R. Given a process node $p \in V_1$, R computes the relevance scores of all the nodes in V_1 with respect to p. The nodes with higher relevance are the 'neighbours' of p. For example, given the process WorkBright (an online application for employee onboarding), we compute the relevance scores for all other processes with respect to it, and would expect that an Android development framework, such as the Android Software Development Kit (SDK) (Google's mobile UI framework), would score low.

Given a row node $a \in V_1$, we would like to compute a relevance score for all other row nodes, $b \in V_1$. Intuitively, we conduct multiple random walks starting from a, and count the number of times that we visit each $b \in V_1$. The probability of visiting b from p is the relevance score we want to obtain. The final result is a $1 \times k$ vector consisting of all the relevance scores to a. A process b typically has a high relevance score with respect to a if b has many connections to a or the connections are exclusive between a and b.

Next, we compute the set of anomaly scores, A. Given a process node p in V_1, we compute the anomaly scores for nodes in V_2 that link to p. A node with a low anomaly score is an anomaly to p. Thus, anomalies are the individuals that use processes specific to more than one department or team. Given the natural inter-group connections (between V_1 and V_2), our objective is to discover the outliers within the group.

Based on the relevance scores for V_1, we compute the anomaly scores for the nodes in V_2. We identify nodes with a low score as anomalous. Given a column node $u \in V_2$, we first find the set S_u of row nodes to which u links: $S_t = \{p| \leq p, u \geq \in E\}$. If u is 'normal', then the relevance scores between any pair of elements in S_t should be high. In order to assess the performance of our model, we test whether the anomalous users flagged by our model are among the known compromised accounts. We hence define the label for a user to be 1 if it is compromised during the observation window, and 0 otherwise.

4.3.2 Results and discussion

We present results from four 7-day observation periods, beginning with day 51 in Table 4. On average, we have over 9,000 active users during each period, running over 5,000 unique processes. As we slide the window forward in time, we observe a greater number of compromised days (1 day, 2 days, 5 days, and 7 days). We achieve promising results in terms of overall accuracy of the algorithm and the ROC curve for each of the four moving windows is shown in Figure 18.

Compromised days	All Users	Compromised Users	AUC
1 Day	8897	2	0.87
2 Days	8939	2	0.96
5 Days	8695	4	0.96
7 Days	8654	4	0.90

Table 4: *AUC results for 1-7 compromised days. All users is a count of the total number of users active during the period, and compromised users is the number of malicious users.*

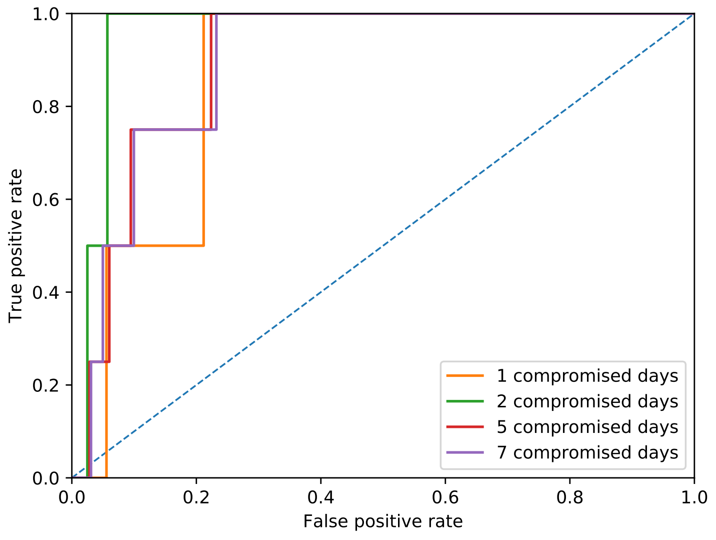


Figure 18: *ROC curve for each window including 1, 2, 5, and 7 compromised days. We consistently achieve high AUC scores.*

The distribution of the normalised anomaly scores for each user is shown in Figure 19. The lower the score, the more anomalous a user is. In a real-world setting, we imagine that all users below a certain threshold of suspicion will be flagged as anomalous, and that list of users can then be further investigated. The threshold selected will likely depend on two factors, being the level of tolerance to false positives, and the capacity of the organisation to manually inspect users. We find that applying a threshold of 3% typically results in under 200 employees and allows us to find at least one compromised user in each time window considered.

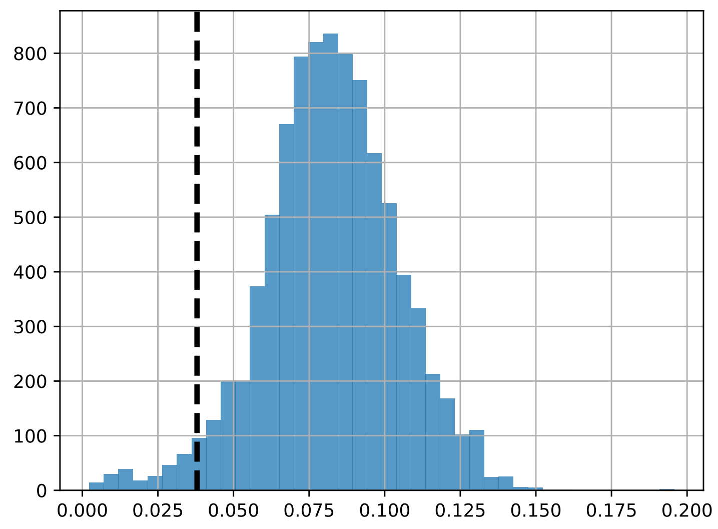


Figure 19: *Distribution of anomaly scores for days 52-58 (two compromised days). The lower the score, the more anomalous a user is. The vertical represents the threshold for identifying anomalous users.*

4.3.3 Future Work

There are a number of promising avenues for future work. First, a richer graph model can be constructed by using weighted edges in order to further improve accuracy. For example, edge weights can represent the number of days the process was used by each user. Second, we have only explored one observation period (a 7 day window), and further investigation is needed in order to determine the optimal observation window. The results presented only considered users. It would also be interesting to extend the analysis to computer accounts as well, in order to see whether results generalise.

Lastly, in order to improve confidence in the anomalies detected and also decrease the number of suspicious user flagged, we can draw on insights from multiple types of activities and combine predictions. For example, we can re-define the graph in terms of authentication activity (where nodes would represent users and the devices they authenticate to), and in terms of network flow activity (where nodes would represent users and the IP addresses they connect to). The three models models can be integrated such that the resulting set of anomalous users would be a function of the results from each model (e.g., the intersection of anomalous users).

4.4 Anomaly detection on categorical data

4.4.1 Problem statement and motivation

Here we introduce several anomaly detection methods that flag unusual activity for further manual inspection (similar methodology to [2]). We motivate our approach by noting that enterprise environments tend to be relatively homogeneous, with many applications being restricted or centrally provisioned; and with users leveraging similar tools on a day-to-day basis. Thus, we investigate the possibility that, upon account compromise, the user significantly changes behaviour in terms of processes launched. Consider a sequence of length l of matrices L^t of size $u^t \times p^t$, where

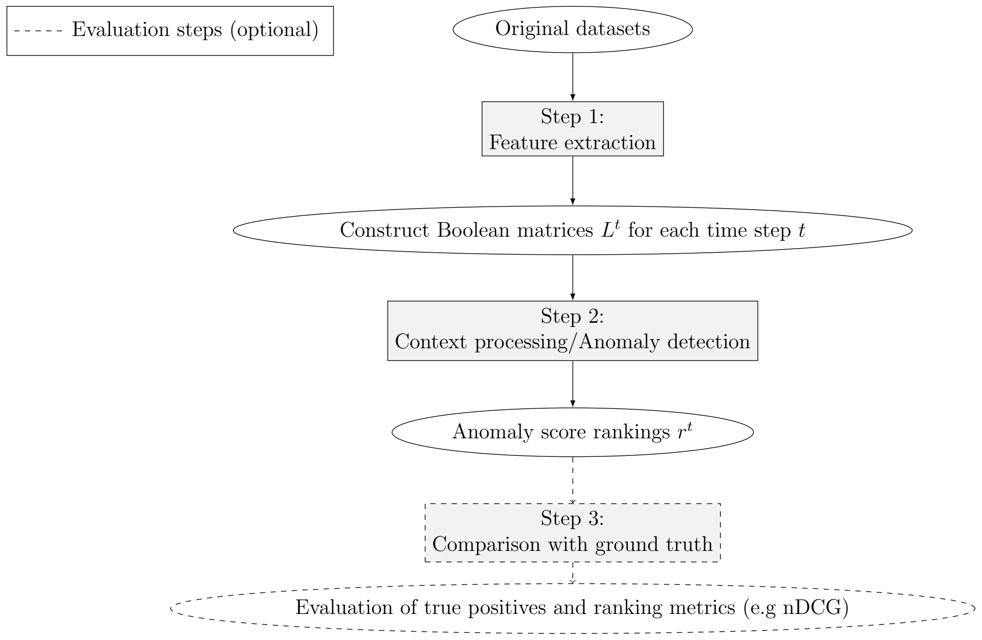
$$L^t_{ij} = \begin{cases} 1, & \text{if user } i \text{ has launched process } j \text{ on day } t, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$\begin{aligned} u^t &= \text{number of users in system at time } t, \\ p^t &= \text{number of unique processes in system at time } t. \end{aligned}$$

Our objective is to produce, for each time step t, a ranking vector r^t of size u^t, ordered by the anomaly score corresponding to each user. A high anomaly score corresponds to unusual activity; thus it may indicate that the user requires closer monitoring. The generation of the anomaly score depends on the particular patterns in the matrices L_t being examined. These approaches summarise user activity using categorical or binary features. We have focused on the identifiers of the processes launched by a particular user. Figure 20 gives a general overview of the data processing and analysis pipeline.

For computing anomaly scores, we consider two methods: Attribute-Value Frequency (AVF) - also described in [2] - and Formal Concept Analysis (FCA). For the following descriptions, we consider each user-process matrix L^t as a dataset D. We also introduce and motivate our evaluation method, normalised discounted cumulative Gain (nDCG). In what follows, we call *attributes* the columns of matrix L^t and *objects/data points* its rows.



Evaluation steps (optional)
Original datasets
Step 1: Feature extraction
Construct Boolean matrices L^t for each time step t
Step 2: Context processing/Anomaly detection
Anomaly score rankings r^t
Step 3: Comparison with ground truth
Evaluation of true positives and ranking metrics (e.g nDCG)

Figure 20: *General anomaly detection approach*

4.4.2 Attribute Value Frequency

Attribute Value Frequency (AVF), introduced in [20], is a fast, scalable and accurate non-parametric outlier detection technique for categorical data. The intuition behind the algorithm is that outliers have attribute values which are infrequent across the dataset. We denote $x^{(i)}$ a typical record/row of matrix L^t at position i and write $x_j^{(i)}$ for the value of attribute j in $x^{(i)}$. If c_j is the number of occurrences of attribute value 1 for attribute j, i.e. $c_j = |\{i \mid x_j^{(i)} = 1\}| = \sum_{i=1}^{n} x_j^{(i)}$ for a dataset D of size n, then the AVF score of a data point x (as described in [20]) is then computed as

$$\mathrm{AVF}(x) = \frac{1}{m} \sum_{j=1}^{m} x_j c_j + (1 - x_j)(n - c_j).$$

That is to say, the contribution to the score for attribute x_j is c_j, the number of occurrences of j-value of 1, when $x_j = 1$, otherwise, the contribution is the number of occurrences of a j-value of 0. The initial multiplication by $1/m$ effectively averages the counts, so $0 \leq AVF(x) \leq n$, but such scaling has no effect on the relative ordering among scores. To make this score potentially suitable for a streaming setting (where the original score would monotonically increase and be meaningless), we modify the AVF score to use the probability of occurrence of attribute values (with $p_j = c_j/n$) instead raw counts such that:

$$\text{AVF'}(x) = \frac{1}{m}\sum_{j=1}^{m} x_j p_j + (1-x_j)(1-p_j).$$

In our setting, this modification only changes the scale of the scores (now comprised between 0 and 1) but not the relative ordering among scores. For more details, see [20].

4.4.3 Formal Concept Analysis

Formal concept analysis (FCA) is a mathematical model for data analysis and knowledge discovery. Given a dataset, structured as a set of objects O, a set of attributes A and the binary relation $R \subseteq O \times A$ between these two sets, the aim of FCA is to derive the implicit relationships between objects and attributes i.e group together objects (or observations) by shared attributes. Typically, if the dataset is matrix L^t, objects would be the rows of the matrix, user identifiers here, and attributes would be its columns, that is process names. FCA outputs concepts formed by a set of objects and a set of attributes such that:

- All objects in the set of objects have all the attributes in the set of attributes.
- There are no other objects in the formal context that have all the attributes in the set of attributes (of the formal concept).
- There are no other attributes in the set of attributes (of the formal concept) that all the objects (in the set of objects of the formal concept) have.

FCA can therefore be considered to be a conceptual clustering method since the attribute set of each formal concept provides an explanation for the grouping of the concept's objects together. Association rules (i.e. dependencies between attributes) can also be derived based on the concepts generated by FCA ([33, 34, 1]). Further details on FCA and association rule mining are provided in [32, 12, 13, 21, 33, 34, 1].

The idea behind using FCA for anomaly detection is as follows:

1. FCA is used to extract frequently co-occurring sets of attributes from the data, thereby inferring highly likely rules for system behaviour.
2. Each user is then scored according to the number of rules it violates and an associated confidence.
3. The users are then ranked in order of anomaly score (the highest score being the most anomalous in this setting).

4.4.4 Normalised Discounted Cumulative Gain (nDCG)

The output of the anomaly detection methods we used is a ranking of users according to their anomaly scores. Here, we suggest using the normalised discounted cumulative gain metric or nDCG for short, which is a metric frequently used in information retrieval to assess the quality of a ranking. [17] considered a typical document search application and argued that relevant documents found further down a list of returned results are highly unlikely to be looked at by a user as they would need more time and effort to be found, thus they become less valuable. This concept motivated the introduction of the nDCG measure [17] and here we apply this to lists of potentially malicious or anomalous behaviour.

The nDCG metric is computed in two steps. First, we compute a score called discounted cumulative gain or DCG. The DCG score relies on the fact that each document/entity in the ranking is assigned a relevance score and is penalised by a value logarithmically proportional to its position/rank in the list of results. We have

$$\mathrm{DCG}_N = \sum_{i=1}^{N} \frac{\mathrm{rel}_i}{\log_2(i+1)},$$

where N is the number of entities/documents in the list, rel_i the relevance score of the ith entity/document in the list. We then normalise the DCG score by the ideal DCG score (iDCG), which is simply the best achievable DCG score, i.e. the score that would be achieved if all relevant entities were at the top of the list. If we suppose we have p relevant entities in the list, the scores can be computed as follows:

$$\text{iDCG}_N = \sum_{i=1}^{p} \frac{\text{rel}_i}{\log_2(i+1)}, \text{ and} \qquad \text{nDCG}_N = \frac{\text{DCG}_N}{\text{iDCG}_N}$$

Here we only consider entities to be either relevant (user accounts that are part of an attack) or irrelevant (user accounts with normal behaviour) and assign a relevance score rel_i of 1 to compromised user accounts and of 0 to user accounts with benign activity, and the idealised score results from ranking all k compromised user accounts at positions $1, \ldots, k$. The closer the nDCG score is to 1, the better the ranking.

4.4.5 Results and brief discussion

We have considered $l = 21$ time steps, between days 54 and 74 of the provided dataset, a period in which the red team was active. For each day, we produced anomaly scores by the AVF and FCA methods, as well as a linear weighted combinations of their scores. We then ranked these anomaly scores in order, obtaining a ranking of users with the highest anomaly scores. Average nDCG scores for the entire period are displayed in Table 5. The nDCG scores were generally relatively low (in the 0.11-0.15 range). The strongest performing anomaly scoring method was AVF, with an average performance of 0.147 over the days when red team data is available. However it is worth noting that the maximum number of compromised accounts, defined a successful login attempt by the red team, for a single day is 4.

Method	Average *nDCG*
FCA	0.115
AVF	**0.147**
FCA + AVF	0.137

Table 5: *Average nDCG scores obtained in the experiments.*

4.4.6 Limitations

It is important to note that the nDCG metric may not provide the full picture in this context and for this type of problem, one should be careful trying to summarise results with only one single metric. However, we have identified the following possible limitations, which if addressed should increase the feasibility of the presented approach for this problem.

1. *Insufficient validation data*: By analysing the percentiles in which the compromised users were found, we empirically obtained promising results. However, it is difficult to assess the reliability of these approaches using standard metrics with such little ground truth information.

2. *Summary data insufficient*: The summary data, whilst very useful given our time and computational constraints, may not capture the entire complexity of the user-process relationship. For better performance, we would choose to work with the full process data.

3. *Parameter tuning*: While AVF doesn't require any parameters, the FCA-based anomaly detection does. Here, given time constraints, we only selected two parameter settings that seemed likely to provide a good trade-off between speed of execution and informativeness of the rules extracted (and therefore anomalies detected). Selecting the right parameter setting would require further exploration.

4.4.7 Future work

We have identified the following possible items as future work:

1. *Explainable anomaly detection* As explained earlier, the basis for the FCA-based anomaly detection method is the scoring of entities of interest depending on the number and severity of system rules' violated. So we know, for each anomaly detected, which were the rules that were violated and thus contributed to the anomaly score. So we should be able to not only provide a listing of anomalies ranked in order of 'suspiciousness' but also provide an 'explanation' for the anomaly, or reasoning for its high score.

2. *Understanding the temporal trends of anomaly scoring.* So far, we scored anomalies on a day to day basis. Future work would include exploring how anomaly scores relate to each other across time.

3. *Using richer feature representations.* Better anomaly score rankings may be produced by using richer feature representations, such as larger categories or numerical data (e.g using features from the Netflows dataset that was not used here or making better use of the authentication data).

4. *Combining outputs of anomaly techniques on numerical data to our techniques outputs.* We would like to explore applying a variety of anomaly detection techniques suitable for numerical data on features such as these and combining the outputs of such techniques with the outputs of the anomaly detection techniques we proposed here. One possible candidate numerical anomaly detection technique would be the Histogram-based Outlier Score (HBOS) introduced in [16] because of its apparent scalability and the fact that it doesn't seem to require too much parameter tuning.

5. *Building probabilistic models.* The features presented here may be combined with other hand-crafted features to build a probabilistic model for anomaly detection, the simplest example of which is a multivariate Gaussian. This would allow per-user alerts when combinations of features are particularly unusual.

6. *Using GPU-accelerated versions of algorithms.* The matrix-based algorithms we have presented are the ideal scenario for GPU acceleration. Indeed, such implementations already exist for FCA, [22, 21].

4.5 Quantile additive model for NetFlow anomaly detection

In [9] it is proposed to use quantile regression forests (QRFs) for NetFlow anomaly detection, [23]. The idea consists in observing the number of connections initiated in a fixed time bin and try to detect if the traffic is unusual. However, as the network traffic varies over the day, it is not possible to use a fixed limit that would determine if the traffic is unusual. QRFs allow us to model a given quantile of distribution of the traffic conditional on some covariates, say, the time of the day or the number of packets sent. Once the quantile regression has been fitted, we can then test if a traffic is unusual compared to previously observed traffic. Although QRFs proved very useful, a major issue with their use is the difficulty to interpret them. If the regression has multiple covariates and we use QRFs, we cannot easily identify why a flag has been raised.

Quantile additive models (QAMs) are semi-parametric models that combine flexibility with interpretability. Thanks to recent advances in QAMs, [10], [14], QAMs could now be used instead of QRFs to obtain more interpretable quantile regressions for anomaly detection. Linked to these articles, two R packages are currently available (‘qgam’: [11], ‘aqmm’: [15]).

4.5.1 One Dimensional Example

As an example, in Figure 21, we fit quantile regression curves (here are 5% and 95%) over a single day with a single covariate (epoch time).

4.5.2 Covariate Analysis

In this section, our observed variable is the number of connections in a 30 second time bin (referred to as time t). A list of potential covariates for the quantile regression is given below.

1. Time of the day (t itself): This is the covariate used in Section 4.5.1.
2. Autoregressive coefficients: Number of connections during time bins ($t-1$, $t-2$,...,$t-k$), where k is to be determined.

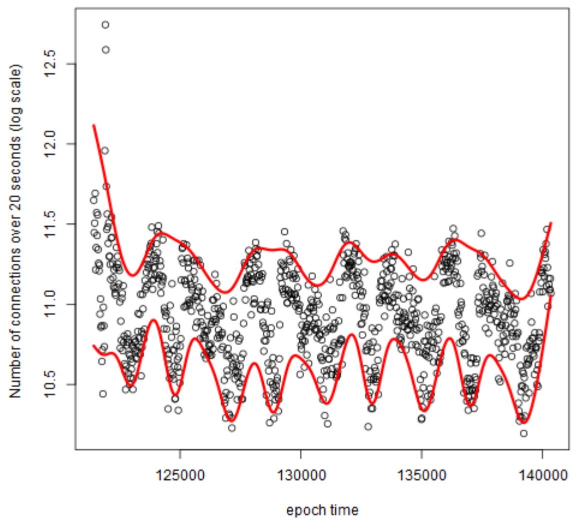


Figure 21: *This example shows a fit using package qgacv of 5% and 95% quantiles using a single covariate (epoch time). The data is NetFlow data from day 2. Epoch time is split into 20 seconds bins and the number of connections per bin is counted.*

3. Number of unique source computers/destination computers during time bin $(t-1)$.

4. Average source/destination packet size during time bin $(t-1)$. To calculate the packet sizes we used NumberOfBytes/NumberOfPackets. If NumberOfPackets $= 0$, we set AveragePacketSize=0. The ratio of these values can also be considered.

5. Average communication duration during time bin $t-1$.

6. Number of communications over time bin $(t-1)$ that use a given protocol. There are three protocols used in the dataset: 1 (ICMP), 6 (TCP) and 17 (UDP).

Using this list, we can begin to explore the relationships between these variables and our output. In Figure 22 we consider each of days 3–6 and show the number of communications for the time bins.

In the following figures, we consider using several covariates over a single day (day 3) for the analyses.

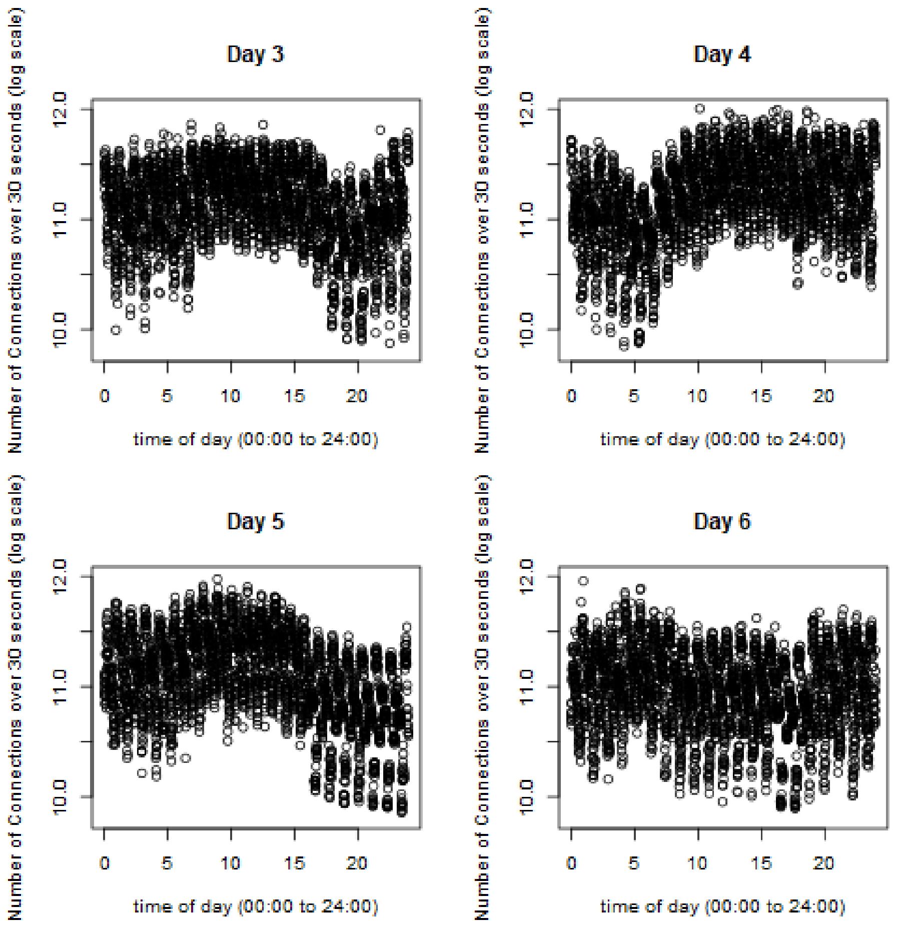


Figure 22: *This figure shows indication of a link between time of day and number of connections.*

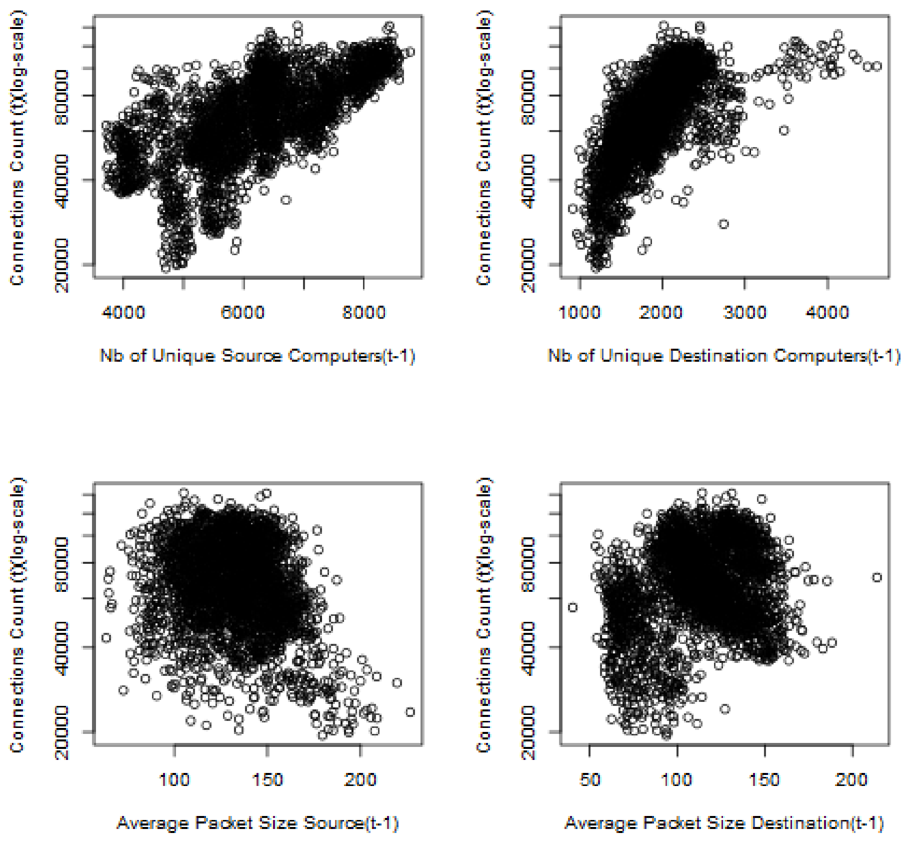


Figure 23: *Plot of log connection count vs 4 potential covariates: number of unique source computers, destination computers, and their average packet sizes during time bin* $t-1$.

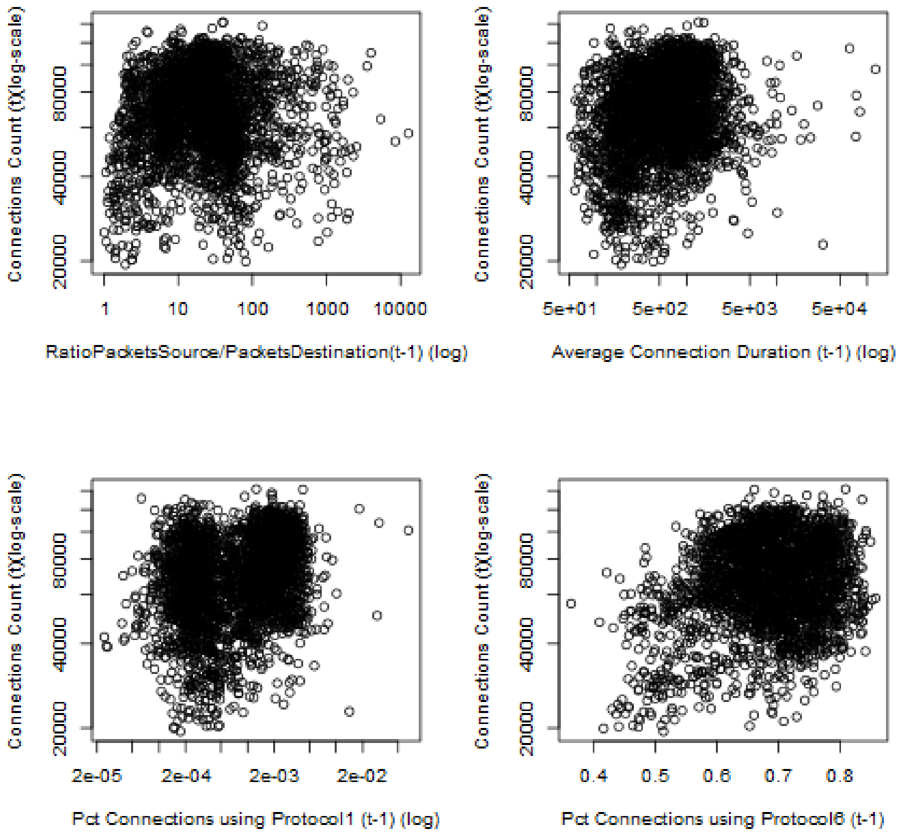


Figure 24: *Plot of log connection count vs 4 potential covariates: ratio of number of source packets to number of destination packets, average communication duration, and the number of communications using each of protocol 1 (ICMP) and 6 (TCP).*

4.5.3 A four covariate fit

Based on our previous analysis, we can now fit a four covariate example. We merge the data, from between midnight and 10am on days 3–6, providing us with 4 observations for each 30 second time bin. In Figure 25 we show the 5% quantile regression curves for each of the 4 covariates, being time, number of unique source computers at $(t-1)$, percentage of previous connections over the previous 30 seconds $(t-1)$ that were using protocol 6 and entropy destination at $t-1$. Note that the entropy destination is simply an empirical probability for each of the unique observed destinations. We see that most covariates seem to have a relation with the response, whilst the entropy destination (t-1) shows little correlation with the response for the 5% quantile. Section 4.5 could be complemented by fitting the regression over more days, or different parts of the day, analysing other potential covariate combinations and comparing the results to QRFs, [23].

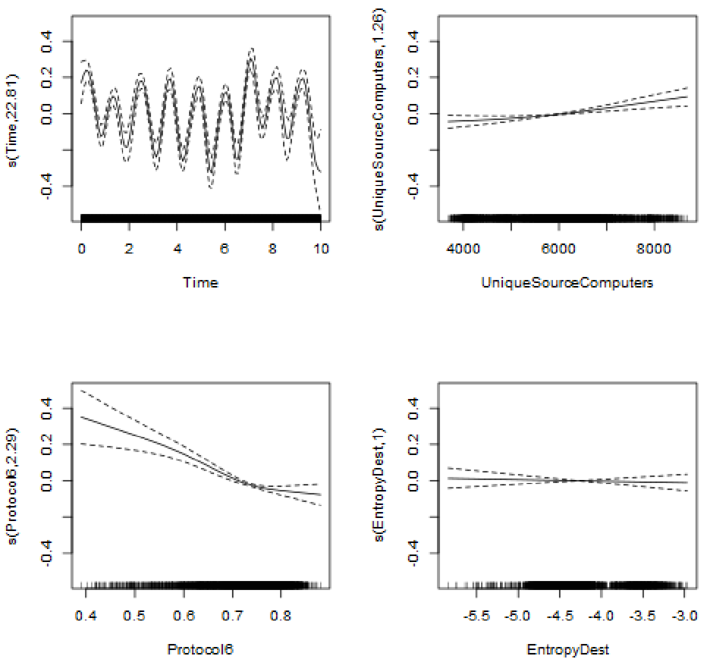


Figure 25: *QAM fit for 5% quantile together with 5% confidence interval.*

4.6 Procrustes analysis on adjacency embeddings for NetFlow data

In this section, the graph adjacency matrices obtained from the daily summaries of NetFlow connections are considered. For each day $t = 2, \dots, 90$, a graph $\mathbb{G}_t = (V_S, V_D, E_t)$ is obtained. The sets V_S and V_D are two distinct node sets: V_S represents the set of source nodes observed in the entire dataset, and similarly V_D is the set of destination nodes. The set E_t is a time dependent edge set, where $(i, j) \in E_t,\ i \in V_S,\ j \in V_D$ if $i \to j$ on day t. Note that, in general, $V_S \cap V_D \neq \varnothing$ since each node might simultaneously act as a source or a destination. Usually, the behaviour of a node when acting as a source is different from its activity patterns as a destination, and therefore the graphs can be roughly interpreted as bipartite, yielding binary adjacency matrices $\mathbf{A}_t \in \{0,1\}^{|V_S|\times|V_D|}$, $\{A_t\}_{ij} = \mathbb{1}_{E_t}\{(i,j)\}$, where $\mathbb{1}.\{\cdot\}$ represents the indicator function. Potentially, it is also possible to consider weighted adjacency matrices based on the daily counts of the number of observed events on each edge, but this approach is not advisable since the counts are extremely heterogeneous. A possible alternative could be to set weights $W_{ijt} = \log(1 + N_{ijt})$, where N_{ijt} represents the number of observed connections on the edge (i, j) on day t.

4.6.1 Embedding directed graphs

For any matrix $\mathbf{M} \in \mathbb{R}^{m\times n}$ of rank k, there are orthogonal matrices $\mathbf{U} \in \mathbb{R}^{m\times m}$, $\mathbf{V} \in \mathbb{R}^{n\times n}$ and a diagonal matrix $\mathbf{D} = \mathrm{diag}(\sigma_1, \sigma_2, \dots, \sigma_k) \in \mathbb{R}^{k\times k}$, where the σ_i's are called *singular values* of $\mathbf{M}$, and the columns of $\mathbf{U}$ and $\mathbf{V}$ are the *left* and *right singular vectors* of $\mathbf{A}$, such that:

$$\mathbf{A} = \mathbf{U} \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{V}^\top \text{ with } \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0.$$

The above decomposition is called *singular value decomposition* (SVD) and can be efficiently used to obtain low-dimensional embeddings for an adjacency matrix of a directed graph.

The SVD is a standard method for embedding directed or bipartite graphs [5, 7]. Given an graph with adjacency matrix $\mathbf{A}_t \in \{0,1\}^{U\times V}$, $U = |V_S|$ and

$V = |V_D|$, for one of the daily graphs introduced in the previous section, and a positive integer $r \geq 1$, consider the singular value decomposition

$$\mathbf{A}_t = \begin{bmatrix}\mathbf{U}_t & \mathbf{U}_t^\perp\end{bmatrix} \begin{bmatrix}\mathbf{D}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_t^\perp\end{bmatrix} \begin{bmatrix}\mathbf{V}_t^\top \\ \mathbf{V}_t^{\top\perp}\end{bmatrix} = \mathbf{U}_t\mathbf{D}_t\mathbf{V}_t^\top + \mathbf{U}_t^\perp\mathbf{D}_t^\perp\mathbf{V}_t^{\top\perp},$$

where $\mathbf{U}_t \in \mathbb{R}^{U\times r}$, $\mathbf{D}_t \in \mathbb{R}_+^{r\times r}$ diagonal, $\mathbf{V}_t \in \mathbb{R}^{V\times r}$, and the other matrices have consequently appropriate dimension. The r-dimensional adjacency embedding of $\mathbf{A}_t$ in $\mathbb{R}^r$ are defined as:

$$\hat{\mathbf{U}}_t = \mathbf{U}_t\mathbf{D}_t^{1/2}, \qquad \hat{\mathbf{V}}_t = \mathbf{V}_t\mathbf{D}_t^{1/2}.$$

Similarly, it is possible to construct an embedding of the graph based on the normalised Laplacian matrix for the directed graph and its singular value decomposition:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}_{\text{out}}^{-1/2}\mathbf{A}\mathbf{D}_{\text{in}}^{-1/2},$$

where $\mathbf{D}_{\text{out}}$ and $\mathbf{D}_{\text{out}}$ are the out-degree and in-degree matrices

$$\mathbf{D}_{\text{out}} = \text{diag}\left(\sum_{j=1}^{n} A_{ij}\right), \qquad \mathbf{D}_{\text{in}} = \text{diag}\left(\sum_{i=1}^{n} A_{ij}\right).$$

4.6.2 Procrustes analysis of shapes

Procrustes analysis [see e.g. 6] is a common technique used in statistical shape analysis. Given two shapes C_1 and C_2, Procrustes analysis aims to find the optimal superimposition $\mathcal{S}(C_2)$ of C_2 on C_1 using three types of operations: translation, scaling and rotation. In some cases, reflection is also used.

Assume that the two shapes C_1 and C_2 are represented by two matrices of equal size, say $n \times m$, and that C_1 is used as reference shape. Procrustes analysis proceeds as follows:

1. *translate* the shapes by removing the centroid,
2. *scale* the shapes to have variance 1 on each dimension,
3. *rotate* the shapes resulting from the previous step by minimising the sum of the squared distances between the points.

A 2-dimensional toy example of the steps involved in the Procrustes alignment is presented in Figure 26.

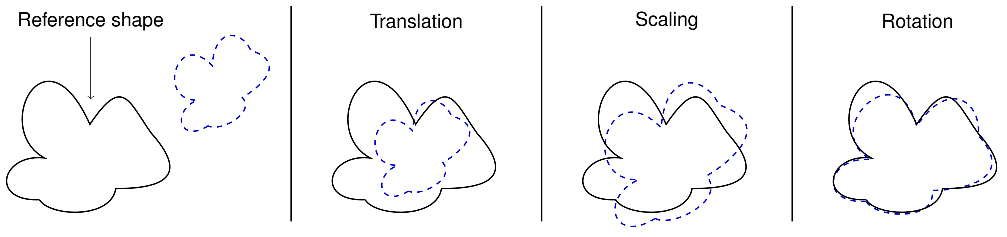


Figure 26: *Pictorial representation of Procrustes alignment of two clouds.*

Potentially, it is possible to use the same method to superimpose a set of q shapes $C_j, j = 1, \dots, q$ to a reference shape C_0, but improved results are usually obtained by *generalised Procrustes analysis* (GPA). The GPA algorithm iterates standard Procrustes analysis, updating the reference shape:

1. choose an arbitrary reference shape (e.g. select it from the available shapes),
2. repeat (for a small number of iterations):
   (a) apply Procrustes superimposition of the shapes to the current reference shape,
   (b) set the updated reference shape to the mean shape of the current set of superimposed shapes.

4.6.3 Scoring source and destination nodes using Procrustes analysis of the adjacency embedding

Assume that the adjacency embedding for the source nodes, $\hat{\mathbf{U}}_t \in \mathbb{R}^{U \times r}$, for a suitably chosen r, are calculated for $t = 2, \dots, 90$. The embedding can be interpreted as a r-dimensional shape, and the normal activity of the network can be roughly interpreted as the *reference shape* observed over a training period of no attacks. Anomalous nodes will correspond to latent positions on the shapes which largely deviate from the learned normal, or reference behaviour. More formally, comparable re-aligned shapes $\hat{\mathbf{U}}_t^\star \in \mathbb{R}^{U \times r}$ for the entire dataset can be obtained from Procrustes analysis, generalised Procrustes analysis, or a combination of the two. Moreover, a reference shape $\hat{\mathcal{U}} \in \mathbb{R}^{U\times r}$ must be obtained for comparisons among different graphs. Comparison between the aligned shapes and the reference allows to understand which of the graphs deviate from the normal behaviour of the network. A simple score of overall dissimilarity is, for example:

$$\Delta_t = \|\hat{\mathbf{U}}_t^\star - \hat{\mathcal{U}}\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Similarly, standardised scores $\Delta_{it},\ i = 1,\dots,U,\ t = 2,\dots,90$ for detection of anomalous nodes can be obtained as follows:

$$\Delta_{it} = \sqrt{\frac{1}{r}\sum_{j=1}^{r}\left(\frac{\hat{U}_{tij}^\star - \hat{\mathcal{U}}_{ij}}{\hat{\mathcal{U}}_{ij}}\right)^2}.$$

Reference and realigned shapes, used for anomaly detection purposes, can be calculated in different ways. Three methods are discussed here:

(1) calculate the generalised Procrustes alignments on a training set of shapes, and take the mean of the resulting shapes as reference for anomaly detection. Then align the shapes in the test set to the reference shapes and calculate the anomaly scores,

(2) calculate generalised Procrustes aligments on the entire dataset, and take the mean of the aligned shapes as reference. Then calculate the anomaly scores from the distances between the realigned shapes and the reference,

(3) calculate the mean shape on a training set, use it as reference, and calculate standard Procrustes alignments with respect to the reference for all the embeddings in the dataset.

Preliminary analysis of the results seems to suggest that method (1) does not provide reliable scores since the test set shapes tend to be largely mismatched compared to the training set shapes. More encouraging results have been obtained using methods (2) and (3).

The scores Δ_{it} can be compared across all the source nodes, on each day. A similar analysis, following the same principles, can be carried out to score anomalous destination nodes.

A sequence of separate graphs separated by port number could also be obtained, and Procrustes analysis could be used to obtain a mean shape at each time point, to detect variations specific to connections on a specific port.

4.6.4 Results

The NetFlow data, summarised per day, have been filtered to consider only the successful mappings to `CompXYZ`. The failed mappings, anonymized as `IPxyz`, have been removed. The number of latent features has been chosen to be $r = 50$, by visual inspection of the scree plot of the singular values [see e.g. 18].

The plot in Figures 27 and 28 report the scores Δ_{it} for the compromised source node `Comp215429`, obtained using reference shape and realigned shapes resulting from methods (2) and (3). Five spikes are clearly present in Figure 27: LANL experts confirmed that the behaviour detected was a scan of the network being performed from the compromised node. Evidence of increased anomalous behaviour is also present at different dates, when red team activity was performed in the authentication dataset. This is more evident in Figure 28.

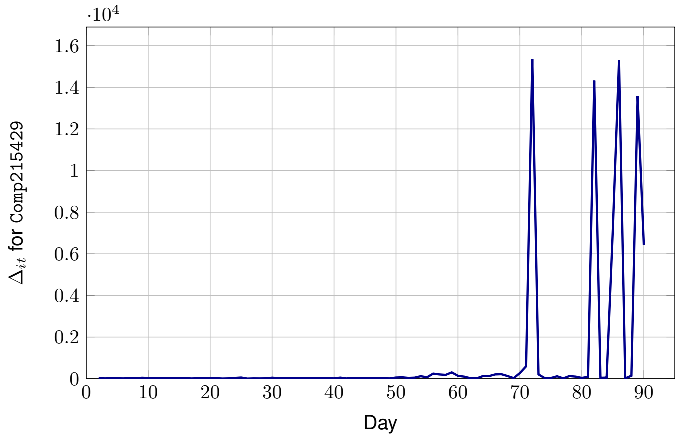


Figure 27: *Time series of anomaly scores Δ_{it} for the compromised source node* `Comp215429`*, using method (2) from Section 4.6.2.*

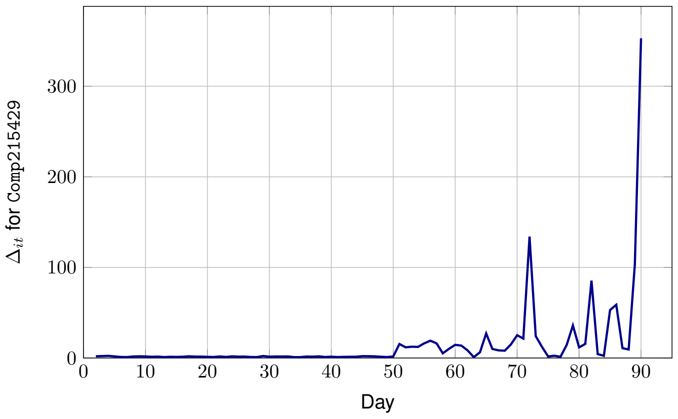


Figure 28: *Time series of anomaly scores Δ_{it} for the compromised source node* `Comp215429`*, using method (3) from Section 4.6.2.*

On the other hand, Figure 29 shows that the interpretation of the scores is not straightforward: in general different scales are observed, and node specific thresholding methods should be used to reliably detect deviations from the normal behaviour of the network. Future work must necessarily address the problem of normalisation of the scores for more direct comparison of abnormal deviations from the learned reference shape.

In Figure 30, the analysis is repeated for the destination nodes, and shows that the scores exhibit increased anomaly levels during the days when the red team is active.

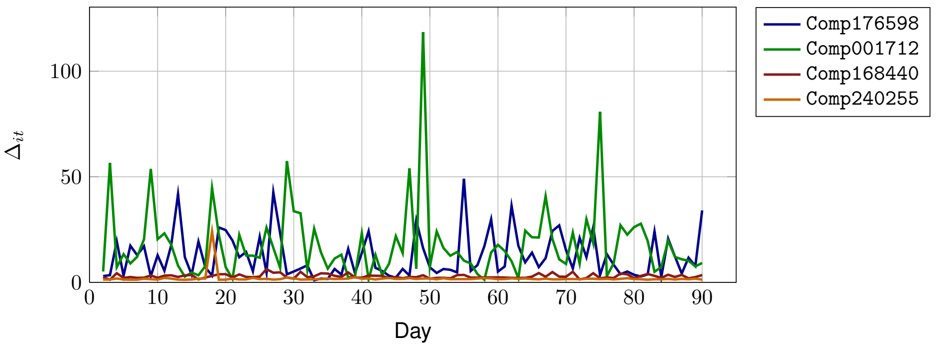


Figure 29: *Time series of anomaly scores Δ_{it} for 4 source nodes chosen at random, obtained using method (2) from Section 4.6.2.*

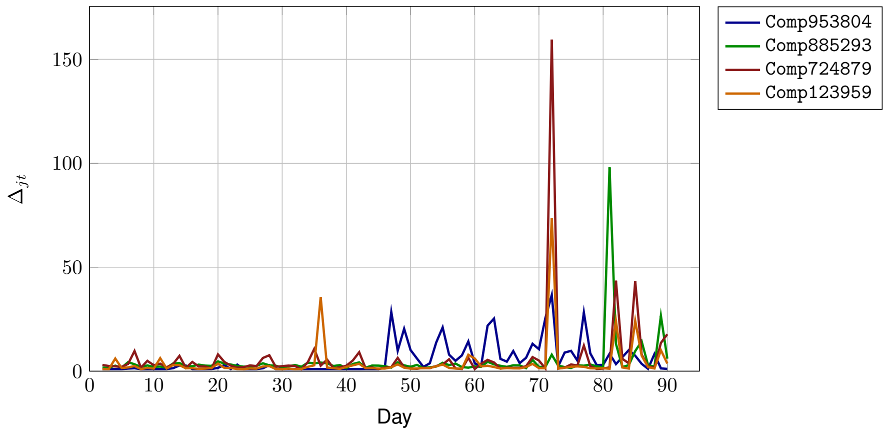


Figure 30: *Time series of anomaly scores Δ_{jt} for 4 compromised destination nodes, obtained using method (2) from Section 4.6.2.*

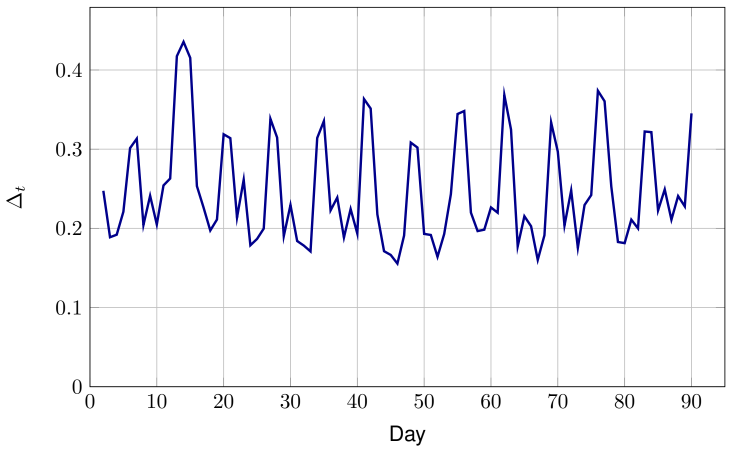


Figure 31: *Time series of Frobenius anomaly scores Δ_t for the entire destination node embedding, using method (2) from Section 4.6.2.*

Finally, Figure 31 shows the overall Frobenius scores Δ_t for the destination node embeddings. It is evident that the structure of the network did not change consistently during the activity of the red team, and the scores are only able to detect seasonal changes.

4.7 Reconstruction of the attack chain in NetFlow data

To understand the behaviour of the attack chain in the LANL network, we consider a known compromised source node. The red team attack can be seen to have started at node `Comp215429` on day 57. For this analysis we consider all network devices that are connected to `Comp215429` over a period of approximately 40 days. Edges where the destination nodes were identified as `IPxyz` were excluded as these were not successfully attributed to any known domain name. In the NetFlow data, we identified 55,025 unique edges emanating from this red team source node and consider the corresponding daily counts for each. This provides us with a matrix, M, of approximate dimension 55025×40, where each row represents a unique edge, and the columns reference the day of interest.

In Figure 32, we plot the aggregated counts (over 40 days) for all the identified unique edges having source node `Comp215429`. The labelled bars in Figure 32 are destination nodes with the highest overall activity across the 40 days. The node with highest sum of counts corresponds to the `EnterpriseAppServer` node, which is expected given it is a core machines in this network.

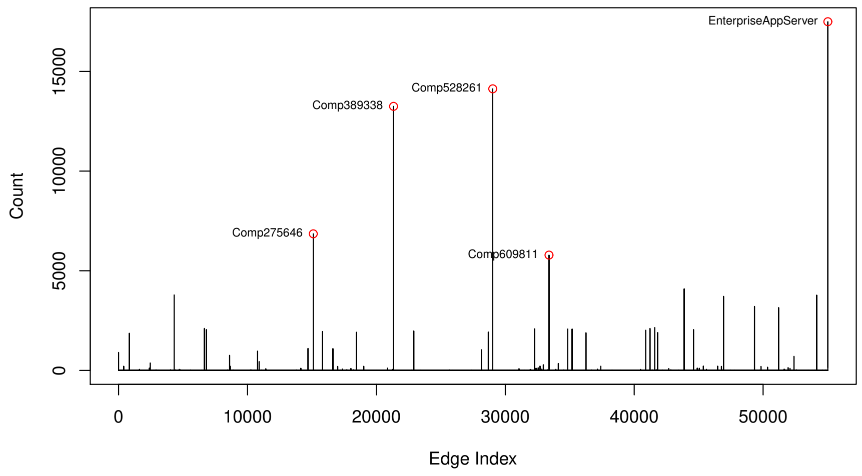


Figure 32: *Edges with corresponding counts of connection.*

In order to reconstruct the attack chain, for each of these nodes we show the nature of the counts per day, as in Figures 33, 34 and 35. Thus we can mark highly irregular activity as more suspicious than consistent behaviour and iteratively apply this idea to identify 'anomalous' nodes. This allows us to illustrate paths which are more likely to exhibit attack behaviour and begin to reconstruct the most likely attack path(s), as illustrated in Figure 36.

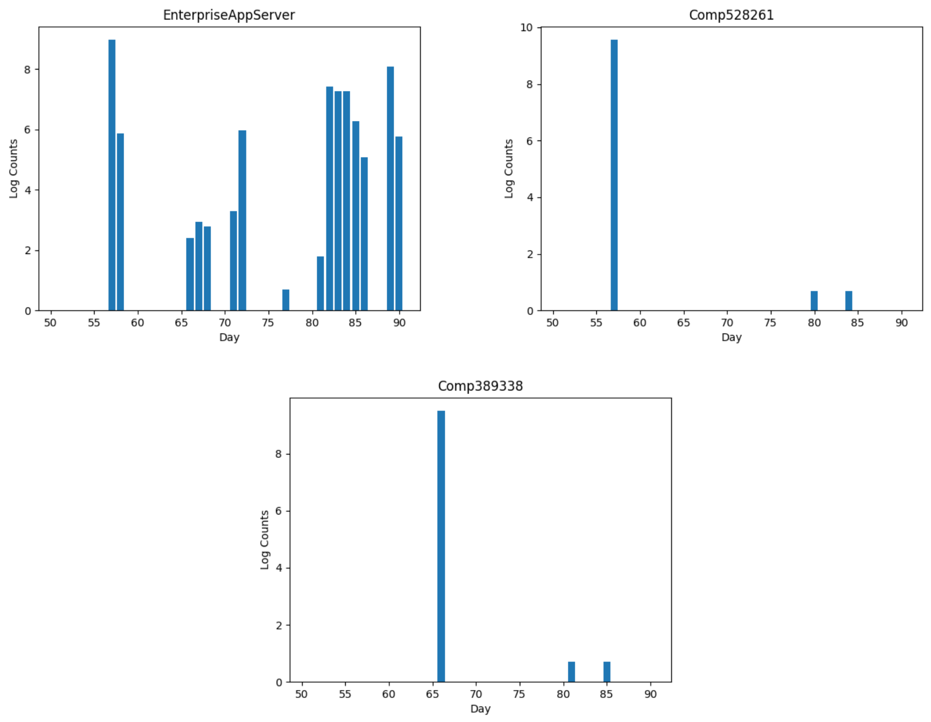


Figure 33: *Figure showing the daily counts for each of the destinations with the highest frequency of connections across the attack period from source node* `Comp215429`*.*

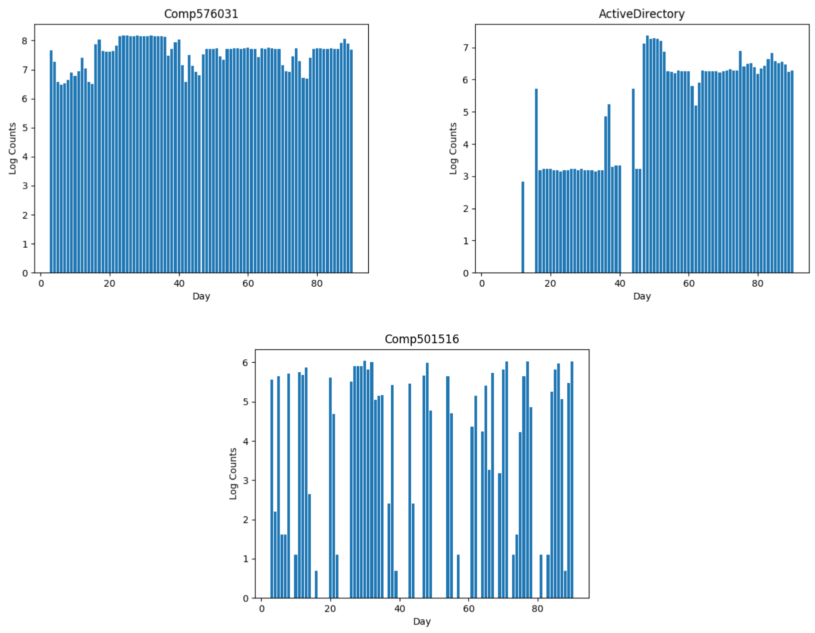


Figure 34: *Daily counts for each of the destinations with the highest frequency of connections across the attack period from source node* `Comp389338`*.*

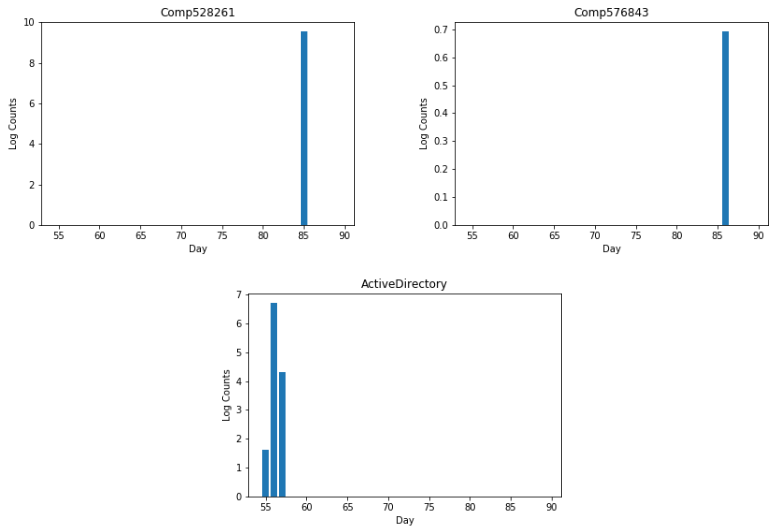


Figure 35: *Daily counts for each of the destinations with the highest frequency of connections across the attack period from source node* `Comp501516`*.*

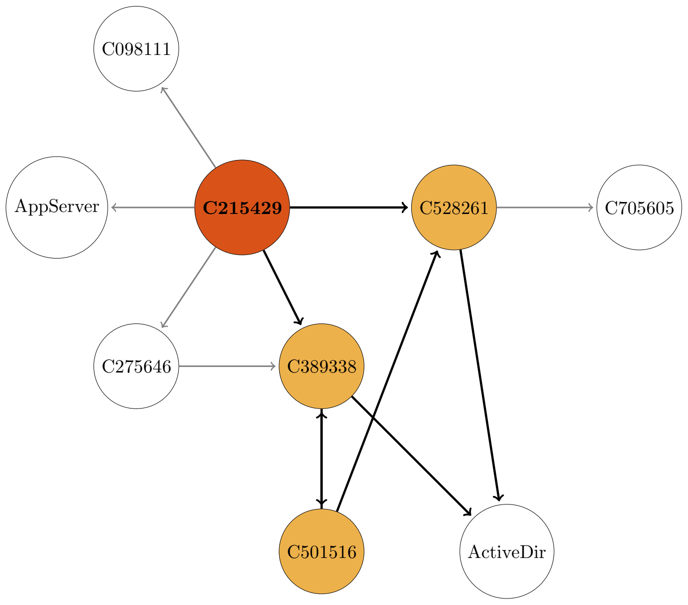


Figure 36: *Graph representation of the reconstructed attack chain, where the red node denotes the known compromised red team client machine, and the orange nodes denote some potentially suspicious devices.*

We can now consider using spectral theory to test for regularities between both suspected anomalous, and normal network edges. As mentioned before, the red team source node is identified as `Comp215429` based on the authentication data. From this we consider the paths to three of the connected nodes on day 57, focusing on the NetFlow data. For each of these edges, we consider the transmission of information over the course of the day as a series of aggregated counts per unit of time (here, per second). In this way we can form a time series of events per bin and apply spectral methods to these streams. In particular, we consider the estimated spectrum, using a method known as multitapering, and from these compute the estimated coherence and partial coherence [25]. Note that the coherence function provides us a measure of correlation in the frequency domain, whilst partial coherence between two processes measures the coherence after removing linear effects from rest of the nodes in the network. In essence, multitaper estimation uses K orthogonal data tapers to reduce bias in the spectral estimation procedure. The resulting multitaper spectral estimate used is an average of the K direct estimates. Here we consider three time series, representing three edges within the LANL network, all originating at the red team source node, `Comp215429`. The figure 37 illustrates the result.

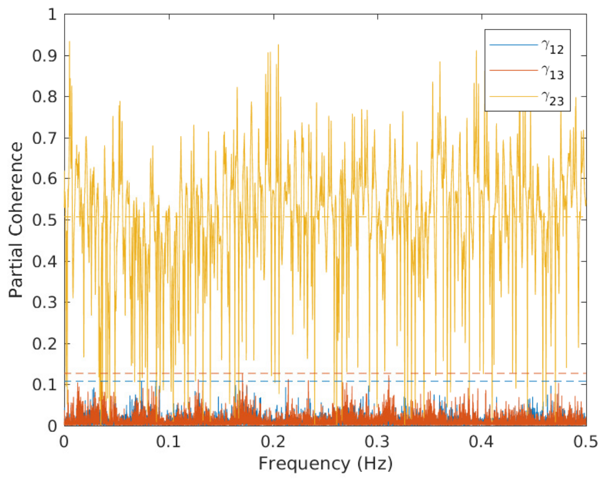


Figure 37: *A figure illustrating the partial coherence between each of the three edges considered. Frequency of oscillation goes from 0 to the Nyqiust frequency, 1/2.*

Here we have used the number of events per second on day 57 for each of the following edges:

1. S1: `Comp215429` $\longrightarrow$ `C275646`.
2. S2: `Comp215429` $\longrightarrow$ `C528261`.
3. S3: `Comp215429` $\longrightarrow$ `EnterpriseAppServer`.

Thus the above plot illustrates the relationship between each of these three edges. The premise for doing this is that in theory, we do not expect coherence between edges of a network and thus any regions of significant coherence may indicate anomalous behaviour. Significance has here been determined by using a multiple hypothesis test on each of the three partial coherence estimates as given by theory in [28]. Whether this anomalous behaviour is in fact malicious would need to be investigated using other methods and further data sources. Nevertheless, it is of interest that the partial coherence between S1 and S2, and S1 and S3 is non-significant whilst between S2 and S3 is.

5 Conclusion and Future Work

Overall, here we have explored a wide range of different ideas and methods for investigating enterprise cyber-security data. Whilst no single method stands as a clear solution, this is to be expected due to the vast and open-ended nature of this challenge. Many interesting directions were explored here and with improvements and combinations of these approaches, there is promise for further developments in this area. Specific directions for future work have been described in detail at the end of each analysis section, along with a discussion identified limitations. We summarise the key areas for future work here.

Self-organising maps, as explored in Section 3, provide a promising method for visualising high-dimensional multi-source data, and further, identifying unusual behaviour with interesting results. Thus, this has potential to be a powerful method for application in cyber-security. As the experiments presented have only considered features derived from NetFlow activity, it would be useful to extend this to research how SOM's might be able to capture the behaviour of the network from multiple sources. For this, we can also potentially consider combining this analysis with other methods explored which aim to handle data fusion and dimensionality reduction.

Topic modelling methods as discussed in Section 4.1, highlighted that from this dataset, malicious processes can be captured compactly by a few topics, an important result. Whilst this methodology would need to be made more robust and rigorous, it could prove interesting to investigate using the output to inform other models tried here. Due to time limitations, we were not able to present results for merging data effectively. This is a key area which we would like to develop further. K-means clustering methods have also been considered for grouping behaviours and identifying anomalous communication paths. Extension of these ideas has been shown to perform well on other datasets, [8] and so with further time, this may also provide insight into how best to merge the data sources. By investing in this area, we believe that many of the analysis methods explored here could achieve better results.

Other areas which illustrated particular potential for anomaly detection were FCA and AVF as they have scope to provide analysts with ranked anomaly scores, and also an associated reasoning, and procrustes analysis.

6 Team members

Bertrand Nortier is a PhD student at the University of Bristol (School of Mathematics) and is currently a PhD enrichment scheme student at the Alan Turing Institute. Bertrand wrote section 4.5 of this report (Quantile Additive Model for NetFlow Anomaly Detection).

Camelia Simoiu is a PhD candidate at Stanford University. She is broadly interested in understanding how user behaviour relates to security outcomes, and draws on methods from machine learning, applied statistics, and online experiments to build quantitative models of behaviour.

Edward Chuah commenced his doctoral studies at the University of Warwick in October 2016. He is very interested in how research could be used to support data centres and high-performance computing systems, with a focus on reliability and system management concerns. He has a strong interest for applied research that is both industry oriented and academically rigorous. He likes to bring his enthusiasm for developing and working on projects that achieve results on time and budget. He enjoys solving real world problems using experimental approaches that are methodologically sound.

Francesco Sanna Passino is a PhD student in statistics in the Department of Mathematics at Imperial College London. He is broadly interested in latent feature models for network link prediction, with applications to statistical cyber-security.

Ghita Berrada, currently a Research Associate in Health Informatics at King's College London, was a Research Associate at the University of Edinburgh (School of Informatics) at the time of writing. Her research mainly centres around decision support systems (especially in healthcare and computer security), with a particular focus on unsupervised learning techniques/anomaly detection (and their explainability) and issues of provenance. She co-wrote section 4.4 with Victor Darvariu (with most of the code used in section experiments to be found at https://gitlab.com/berradag/ad).

Hanne Hoitzing completed her PhD in applied mathematics at Imperial College London and now works as a cyber security data scientist at G-Research. Together with Keli Liu, she wrote Section 4.1 of this report.

Henry Clausen is a PhD student at the University of Edinburgh (School of Informatics) and is focusing on using statistical and ML methods for mining meaningful semantic sequences in network trac for anomaly detection.

John Booth

Karl Hallgren

Kaushik Jana is a research associate in Statistics Section of Imperial College London and is working in the Alan Turing Institute –Lloyds Register Foundation Programme on Data-Centric Engineering. He received a PhD in Statistics from Indian Statistical Institute.

Keli Liu

Leigh Shlomovich is a PhD student in statistics in the Department of Mathematics at Imperial College London. She is predominantly interested in time-frequency analysis of point processes for applications in cyber-security.

Qi(Katherine) He is a PhD student at University College London. Her research interest lies in Bayesian based classification and variable selection and research in econometric studies with high dimensional data. She contributes to this research through doing clustering based cross domain data fusion for anomalous activity detection.

Dr Roberto Jordaney completed his PhD in 2019 in the S2Lab at the ISG department of Royal Holloway, University of London working in detecting concept drift in machine learning model for security. His research was supervised by Prof Lorenzo Cavallaro, Chair in Cybersecurity at Kings College London. Now he is working as Research Engineer at HP Labs.

Silvia Metelli is a post doctoral researcher at Imperial College London and The Alan Turing Institute. She is broadly interested in dynamic network modelling and clustering techniques for cyber-security applications.

Victor-Alexandru Darvariu is a PhD student at University College London and The Alan Turing Institute, interested in the design and application of machine learning techniques to networked systems. He also has an applied background in cyber-security, having worked as a software engineer on identity management platforms in the financial sector.

Zhenzheng (Helen) Hu is a PhD student in Statistics at University College London and Alan Turing Institute. Her current research focus is statistical modelling of binary data that can formally account for and model potential misclassification.

References

[1] Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Computational LogicCL 2000*, pages 972–986. Springer, 2000.

[2] Sidahmed Benabderrahmane James Cheney William Maxwell Himan Mookherjee Alec Theriault Berrada, Ghita and Ryan Wright. A baseline for unsupervised advanced persistent threat detection in system-level provenance. *arXiv preprint arXiv:1906.06940*, 2019.

[3] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[5] I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 269–274, New York, NY, USA, 2001. ACM.

[6] I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis, with Applications in R. Second Edition.* John Wiley and Sons, Chichester, 2016.

[7] D.M. Dunlavy, T.G. Kolda, and E. Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2), 2011.

[8] Hoda Eldardiry, Evgeniy Bart, Juan Liu, John Hanley, Bob Price, and Oliver Brdiczka. Multi-domain information fusion for insider threat detection. In *Security and Privacy Workshops (SPW), 2013 IEEE*, pages 45–51. IEEE, 2013.

[9] Marina Evangelou and Niall Adams. An anomaly detection framework for cyber-security data. unpublished manuscript, 2018.

[10] Matteo Fasiolo, Yannig Goude, Raphael Nedellec, and Simon N Wood. Fast calibrated additive quantile regression. *arXiv preprint arXiv:1707.03307*, 2017.

[11] Matteo Fasiolo, Simon N. Wood, Yannig Goude, and Raphael Nedellec. R package: qgam. `https://cran.r-project.org/web/packages/qgam/index.html`, 2018.

[12] Bernhard Ganter and Gerd Stumme. Formal Concept Analysis: Methods and Applications in Computer Science. Technical report, TU Dresden, 2003.

[13] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1997.

[14] Marco Geraci. Additive Quantile Regression for Clustered Data with an Application to Children's Physical Activity. *arXiv preprint arXiv:1803.05403*, 2018.

[15] Marco Geraci. R package: aqmm. `https://marcogeraci.wordpress.com/software/`, 2018.

[16] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster and Demo Track*, pages 59–63, 2012.

[17] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[18] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

[19] Teuvo Kohonen. Exploration of very large databases by self-organizing maps. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 1, pages PL1–PL6. IEEE, 1997.

[20] Anna Koufakou, Enrique G. Ortiz, Michael Georgiopoulos, Georgios C. Anagnostopoulos, and Kenneth M. Reynolds. A scalable and efficient outlier detection strategy for categorical data. In *ICTAI 2007*, pages 210–217, 2007.

[21] Petr Krajca, Jan Outrata, and Vilem Vychodil. Parallel recursive algorithm for FCA. In *CLA*, volume 2008, pages 71–82, 2008.

[22] William B Langdon, Shin Yoo, and Mark Harman. Formal concept analysis on graphics hardware. In *CLA*, pages 413–416. Citeseer, 2011.

[23] Nicolai Meinshausen. Quantile Regression Forests. *Journal of Machine Learning Research*, 7(Jun):983–999, 2006.

[24] Yew Seng Ng and Rajagopalan Srinivasan. Multivariate temporal data analysis using self-organizing maps. 1. training methodology for effective visualization of multistate operations. *Industrial & Engineering Chemistry Research*, 47(20):7744–7757, 2008.

[25] D. B. Percival and A. T. Walden. *Spectral Analysis for Physical Applications.* Cambridge University Press, 1993.

[26] Patrick Rubin-Delanchy, Daniel J Lawson, Melissa J Turcotte, Nicholas Heard, and Niall M Adams. Three statistical approaches to sessionizing network flow data. In *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, pages 244–247. IEEE, 2014.

[27] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Relevance search and anomaly detection in bipartite graphs. *ACM SIGKDD Explorations Newsletter*, 7(2):48–55, 2005.

[28] A. T. Walden T. Medkour and A. Burgess. Graphical modelling for brain connectivity via partial coherence. *Journal of Neuroscience Methods*, 180:374–383, 2009.

[29] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. *Unified Host and Network Data Set*, chapter Chapter 1, pages 1–22. World Scientific, nov 2018.

[30] Melissa JM Turcotte, Alexander D Kent, and Curtis Hash. Unified host and network data set. *arXiv preprint arXiv:1708.07518*, 2017.

[31] AM Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.

[32] Rudolf Wille. Concept lattices and conceptual knowledge systems. *Computers & Mathematics with Applications*, 23(6):493 – 515, 1992.

[33] Mohammed J Zaki. Generating non-redundant association rules. In

Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 34–43. ACM, 2000.

[34] Mohammed J Zaki. Mining non-redundant association rules. *Data mining and knowledge discovery*, 9(3):223–248, 2004.

turing.ac.uk
@turinginst