An approximate framework for flexible network flow screening

Niall M. Adams
Department of Mathematics
Imperial College London
London, UK
Heilbronn Institute for Mathematical Research
University of Bristol
Bristol, UK
Email: n.adams@imperial.ac.uk

Daniel Lawson
Heilbronn Institute for Mathematical Research
University of Bristol, UK
Email: d.lawson@bristol.ac.uk

***Abstract*—Network security analysts presently lack tools for routinely screening large collections of network traffic for structures of interest. This is particularly the case when the structures of interest are embodied as summaries of sets of related traffic, essentially behaviour descriptions. This paper sketches a methodology to provide such capability, in the context of flow data. The methodology generates approximate search results, and uses a modular construction to provide the capability to tailor queries for multiple views of the behaviour structure of interest. At core, the methodology involves approximate sequential search procedures. The methodology is framed by a discussion of a large university network.**

I. Introduction

Defending corporate networks from malicious or unauthorised behaviour remains a pressing concern. This is particularly challenging in the context of a large university network for many reasons, including the size and heterogeneity of the network and its users, and constraints on service restrictions arising from freedom in academic research. The former point, essentially *scale*, is particularly challenging since the manpower available for network security is often small in relation to the task.

Packet-based intrusion detection systems (e.g. [1]) are an important part of the network security analyst's toolkit, for blocking known malicious behaviour, for monitoring (or situational awareness) and for exploratory discovery. However, there are a number of shortcomings with packet-level analysis, including: the sheer volume of packet traffic makes routine storage impractical, and respecting privacy concerns around the content of packets. A potentially more critical shortcoming is that most packet-based analysis matches against precise signatures. These characteristics combine to suggest that packet-based analysis is inadequate, in isolation, for the needs of network security analysts.

Flow data, such as CISCO's NETFLOW[1], is another important data source in the network analysts' toolkit. Such data has been successfully used in large-scale network monitoring. For example, [2] provides an example of the detection of a sophisticated intrusion in the context of a corporate network. A different NETFLOW monitoring example is provided by [3]. Typically, analytic use of flow data is off-router, and hence dictates a requirement for more sophisticated storage and processing infrastructure. This paper is concerned with exploiting flow data for exploration and discovery, the structure of which is discussed in Section II-A.

Discussion of packets and flow data demonstrates different levels of abstraction and summary. A flow data record is a higher level description of *behaviour* than a packet. Still higher-levels can be reasoned about. A first example is considering all flow events emitted by a single IP address in an arbitrary period. A second example is all events *of a specific type*, such as `ssh`, in an arbitrary period associated with a specified set of addresses. Figure 2 presents a schematic description of levels of behaviour, and the issue is described in more detail in Section II-B. Such collections of flow events are complicated, and best summarised with a statistical description. The precise statistical description should be determined according to the particular behavioural structure of interest.

Monitoring is not the only concern of network analysts. Having identified a compromised computer, the analyst may wish to find computers which have similar behavioural characteristics. This could be for forensics purposes – to better understand the compromise – or for *discovery* purposes – to reveal other potentially compromised machines. It is this latter activity which is the primary focus of the proposed framework. As noted above, there are numerous ways to characterise behaviour, and for discovery purposes the analyst should have flexibility to make this choice. For example, having identified one corporate computer that is participating as a node in a botnet, the analyst may seek to find other computers with similar behavioural characteristics. Such computers may also be participating in the botnet. This type of context is discussed in more detail in Section II-B.

Finding matches for behaviour associated with a compromise – *screening* – is the objective of the proposed framework. However, both the characterisation of the activity, and the behavioural context for the screening process should be analyst-defined and thus the framework is modular to support *flexible* screening. Given the large size of a corporate flow data corpus, routine searches at this scale are computationally impractical.

[1]http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.HTML

To address this, the problem is posed as an approximate and sequential search task.

There is a natural trade-off here: should the analyst wish to complete a single search with complete accuracy, or numerous approximate searches? Given the complexity of the context, and the propensity for false positives in screening scenarios, the latter approach may be preferable – particularly for discovery purposes.

This paper is concerned with sketching a computational framework for screening in a corporate network. This framework is the "cyber" instantiation and simplification of the more general big-data analysis approach described in [4]. Related ideas, with a focus on processing infrastructure are explored in [5].

II. Data and Context

This section describes the context of the proposed framework in more detail and provides a discussion on the structure of NETFLOW data.

A. NETFLOW data

As discussed in the introduction, this paper is concerned with the use of flow records. Using the example of the network at Imperial College London, the focus will be on NETFLOW. NETFLOW data is a summary description of a coherent flow of packets between two network devices communicating via a router. An (anonymised) sample of NETFLOW data is given in Figure 1. NETFLOW records include the source and destination IP addresses, source and destination ports, start and end times, flow size and other information. The table shows four NETFLOW events, from a common source to four different addresses on two different subnets. Some of these may be web requests and active directory requests.

Some caution is needed in handling raw NETFLOW records, as not all fields are reliably meaningful. For example, the direction (implied by source and destination) of DNS requests, conducted through UDP, can be unreliable.

A large corporate network will generate huge amounts of flow data. For example, a month of Imperial's NETFLOW data is typically in excess of 10 terabytes. This comprises events on more than 40K active corporate devices, and their first hop outside the corporate network boundary. A complication here is the presence of multiple routers and the possible use of virtual routing, which can lead to significant data duplication. By construction, the size of a flow data corpus will be tiny compared to the corresponding packet data. Still, specialist storage and processing infrastructure is required if flow data is to be retained for analysis. The Imperial research team uses HADOOP [6], while the authors of [7] use a bespoke in-house system. For the purpose of this paper the choice of storage and processing infrastructure is not a major issue, provided that it can deliver access to a large amount of stored data in an efficient manner.

A simple mathematical representation of flow data is a graph, with IP addresses as nodes and NETFLOW events as edges. Statistical methods for modelling such representations are reviewed in [8]. Such a crude representation may be inadequate, since it critically ignores both event timing and information that precisely characterises the flow data event. For the type of monitoring and matching purposes that are relevant here, this information is paramount. A more useful formalisation would be to consider marked event processes on edges, with nodes that carry extra feature information.

The abstract question of interest is, given a part of the network (node, edge, neighbourhood, time interval, etc.), how do we find similar structures in retained historic flow data? The abstraction of this question is discussed further in the next section.

Assume that a collection of historic flow data is available, in raw form. The choice of raw form is deliberate, to allow the analyst to pose diverse multiple questions of the data. Denote this collection of M flow records as $\mathcal{D}$, observed over a fixed period. This data refers to events on N internal corporate devices. We would recommend that at least one month's data is sufficient for the task at hand.

B. Behaviour abstraction

There are numerous types of attack style deployed against corporate networks, ranging from opportunistic vandalism to sophisticated state-sponsored intrusion. From the point of view of network forensics, [9] gives a good review. Different types of organisation may be subject to different kinds of attack, according to the character of the organisation. Imperial College London shares many of the cyber-security concerns of any large organisation. Two in particular are important. The first, very common, is preventing the spread of malware. The second is the use of peer-to-peer networking (eg. [10]) for distributing copyright material, such as movies.

If one device on a network is compromised it is likely that there will be others. For example, [7] develop a methodology which successfully finds an intrusion *path*, a connected sequence of compromised machines. Given that a compromise can be found, one way or another, the discovery challenge for the analyst is to find other machines that may be compromised. Since packets cannot typically be stored at scale, an appeal to flow data and characteristics of a behavioural character is required.

The diversity of attack techniques means that the analyst must have flexibility in asking questions of flow data. Figure 2 provides a schematic representation of various levels of abstraction. Time is implicit in the diagram, though at each level the time-scale can be different. *Sessions* are an abstraction which attempt to extract human behaviour from the lower granularity flow data. For example, an SSH session is notionally a single activity that is represented by numerous flow data records. Detecting and exploiting session structure is discussed at length in [11], [12].

Packets, flows and sessions refer to an edge – a connection between two IP addresses. The word clique is used informally, to describe a subset of node or edges (and hence sessions and flows) of interest. These levels can interact with, and inform, graph-based abstractions such as cliques and indeed the whole network.

Should the analyst know precisely which abstract features are of interest, conventional approaches are satisfactory. However, for the purpose of discovery focused-querying, the

```
 Date flow start          DurationProto Src IP Addr    Dst IP Addr      Src Pt Dst Pt Packets Bytes
2013-02-26 17:11:57.289  33.997   TCP    126.253.7.174  100.253.192.9    50314  445    2000    237400
2013-02-26 17:11:57.289  42.997   TCP    126.253.7.174  100.253.192.93   50314  445    1700    364500
2013-02-26 17:12:01.288  0.000    TCP    126.253.7.174  211.202.89.207   51106  80     100     4600
2013-02-26 17:11:58.290  0.000    TCP    126.253.7.173  100.253.192.170  49458  50110  100     7600
```

Fig. 1: Example NETFLOW data (anonymised)

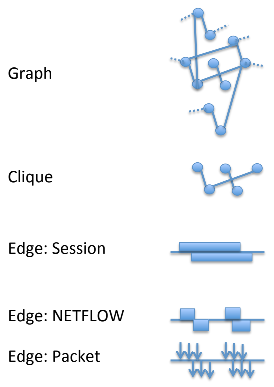


Fig. 2: Schematic representation of some abstraction levels for network traffic data.

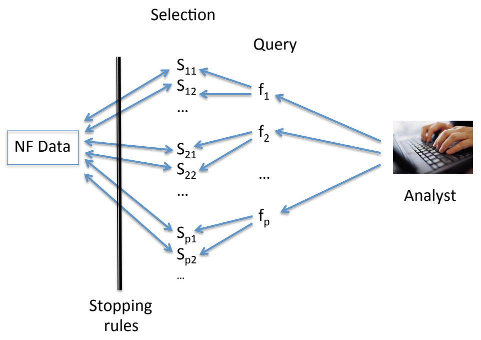


Fig. 3: Schematic representation of the NETFLOW screening framework.

proposed framework retains historic flow data, and constructs appropriate abstractions as required.

III. Methodology

Given the discussion above, the concern of the methodology is to find *behaviours* similar to a target. This behaviour could refer to a computer, an edge, a clique, etc. To keep things concrete, the discussion will be focused on finding computers that exhibit similar behaviour to a target computer. The methodology is not concerned with simple matching such as finding a flow event matched on ports and source IP. This latter objective is best managed, if resources are available, by storing the historic data in a relational database.

Suppose the target machine is of interest to a network analyst because of a particular profile of website requests. Examining the historic flow data associated with this facilitates construction of the empirical distribution of website visits. If the target machine is of interest, a machine with a similar empirical distribution may also be of interest. Note that this requires two things: evaluation of a summary, f, of the target machine's flow data, and the construction of a suitable similarly measure. Of course, in applying the similarity measure the summary, f, must be computed for other machines.

Denote the target computer as A. The usual statistical approach would be to score the similarity of A with some other computer B, using a similarity function $S(A, B)$. Continuing with the website visit example, this function could be, for example, the cross-entropy between the two empirical distributions.

Each computer's NETFLOW data first needs some processing to extract the empirical distribution. A more precise description is to first define A_{NF} as the NETFLOW data associated with computer A. Then the similarity function between A and B is more precisely described as $S(f(A_{NF}), f(B_{NF}))$. Figure 3 give a schematic representation of the framework, stressing the central role of the analyst.

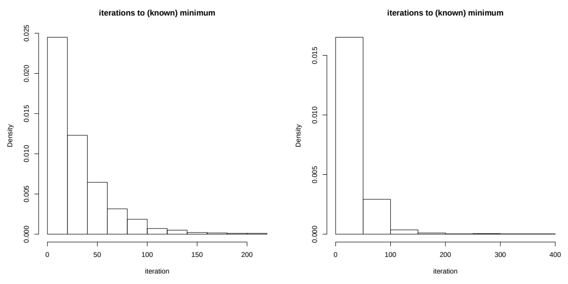
The proposed framework is intended to be able to quickly (though approximately) find the most similar objects, rather than brute force evaluation of all such similarities. At the scale of a corporate network, such routine computation is infeasible. For the data set $\mathcal{D}$, this would require the evaluation of at least $(M-1)\times(M-1)$ similarities, and at best N evaluations of f. The scale of corporate flow data makes this impractical, especially if the objective is to provide the analyst discovery capabilities by providing the means to do multiple queries involving multiple f and S.

Moving to an abstract scenario, consider the *notional* distance matrix

$$D = \begin{bmatrix} 0 & d_{1,2} & \dots & d_{1,N} \\ d_{2,1} & 0 & \dots & d_{2,N} \\ \vdots & \vdots & \ddots & \dots \\ d_{N,1} & \dots & d_{N,N-1} & 0 \end{bmatrix},$$

where $d_{i,j} = S(f(i_{NF}), f(j_{NF}))$, and the similarity function is restricted to a distance metric. This data structure provides all the information that is needed, given f. Since computing this routinely is intractable, the framework uses sequential methods to determine which distances to evaluate. Naturally, this will provide an approximate answer, but it expands the discovery capability of the analyst. This is always a trade-off.

Sequential search procedures have two components, a *selection procedure* and a *stopping rule*. As with other parts of the framework, specifically S and f, choice of these is modular



(a) Unstructured data (b) Structured data

Fig. 4: Distribution of times to secure a minimum

and different choices can be plugged in. Making these modular provides the framework maximum flexibility.

First, consider the selection procedure. To find the nearest object to a target by random selection would require $(N-1)/2$ evaluations, on average, to find the minimum. This is inadequate if numerous queries involving different f and S are required. It is convenient to distinguish two fundamental operations in the context of selection. Evaluating a full row of the notional distance matrix, and evaluating a single distance. The former yields more information, but increases the computation accordingly.

To demonstrate the utility of a simple selection procedure, consider the following toy example. Given a 500×500 distance matrix (a simple realisation of a bivariate Gaussian), consider the average time required to locate the nearest neighbour of a target object.

To begin, evaluate two rows of the matrix (which is about 1% of the total) - requiring 2N evaluations of S and N evaluations of f. Now using a simple, once-only scoring algorithm based on triangulation requires $O(N)$ computations plus sort. Embedding this in a Monte Carlo procedure provides a way to estimate the average number of evaluations for this simple algorithm to secure the (known) nearest neighbour. Figure 4 (a) shows the histogram of these Monte Carlo replicates, while (b) shows corresponding results for an example involving data with more cluster structure. The estimate of average number of evaluations, in both cases, is less than 35, though a significant improvement over the random selection average of 249. Of course, the method exhibits high variance. Refined selection algorithms are part of our on-going research agenda.

In these experiments, the closest object is known. In a real situation, there would be a need to stop the algorithm at some point. There are at least two ways to reason about this. First, The number of evaluations may be fixed in advance, perhaps by computational constraints. Second, an estimate of the probability that the selection algorithm has acquired the closest object can be determined. This estimate, compared to a threshold, can provide a stopping rule. This latter, more flexible approach, is the focus of our current research.

IV. Conclusion

Exploring large amounts of flow data is challenging. For discovery purposes, network analysts may often want to make numerous different type of query to find similar behaviours. The scale of such data challenges this approach. We have proposed an approximate framework that provides the capability for analysts to conduct extensive querying. In addition to enhancing the toolkit, the framework also makes more use of an under-used data source. The framework is not restricted to finding nearest neighbours, and extends readily to distributed storage and processing infrastructure.

The approach is naturally enhanced with pre-screening procedures. This would reduce the size of the search from the whole population of corporate devices to a smaller set.

There are numerous perspectives an analyst can adopt in reasoning about the character of a compromise. This can include devices, edges, port information, local graph structure. Given the changing nature of the attack tactics, it is important that the analyst is able to make imaginative use of flow data. This freedom is embodied in the choice of functions f and S.

At present, we are implementing a version of this framework. There are two statistical research agendas: refined selection algorithms and effective stopping rules.

Acknowledgment

Andy Thomas, research computing manager of Imperial's Mathematics department provided many useful insights. We are grateful to Dr William Oxbury for extensive input.

References

[1] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proc. of 2004 IEEE Symposium on Security and Privacy*, 2004, pp. 211–225.

[2] J. Neil, C. Storlie, C. Hash, and A. Brugh, "Statistical detection of intruders within computer networks using scan statistics," in *Data Analysis for Network Cyber-Security, Adams & Heard, 2014*, 2014.

[3] D. Bodenham and N. Adams, "Continuous monitoring of a computer network using multivariate adaptive estimation," in *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, Dec 2013, pp. 311–318.

[4] D. J. Lawson and N. M. Adams, "A general decision framework for structuring computation using data directional scaling to process massive similarity matrices," *arXiv preprint arXiv:1403.4054*, 2014.

[5] Y. Wang, A. Metwally, and S. Parthasarathy, "Scalable all-pairs similarity search in metric spaces," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 829–837. [Online]. Available: http://doi.acm.org/10.1145/2487575.2487625

[6] T. White, *Hadoop: The Definitive Guide*, 3rd ed. Yahoo Press, 2012.

[7] J. Neil, C. Hash, A. Brugh, M. Fisk, and C. B. Storlie, "Scan statistics for the online detection of locally anomalous subgraphs," *Technometrics*, vol. 55, no. 4, pp. 403–414, 2013.

[8] B. Olding and P. Wolfe, "Inference for graphs and networks," in *Data Analysis for Network Cyber-Security, Adams & Heard, 2014*, 2014.

[9] S. Davidoff and J. Ham, *Network Forensics: Tracking Hackers Through Cyberspace*. Prentice Hall, 2012.

[10] Y.-K. Kwok, *Peer-to-Peer Computing: Applications, Architecture, Protocols, and Challenges*. Chapman Hall, 2011.

[11] P. Rubin-Delanchy, D. Lawson, M. Turcotte, N. Heard, and N. Adams, "Three statistical approaches to sessioninzing Netflow," Department of Mathematics, Univeristy of Bristol, Tech. Rep., 2014.

[12] D. Lawson, P.Rubin-Delanchy, N. Heard, and N. Adams, "Statistical frameworks for detecting tunnelling in cyber defence using big data," Department of Mathematics, Univeristy of Bristol, Tech. Rep., 2014.