Statistical frameworks for detecting tunnelling in cyber defence using big data

Daniel John Lawson†,
Patrick Rubin-Delanchy
Heilbronn Institute
School of Mathematics
University of Bristol
Bristol, UK
†Email: dan.lawson@bristol.ac.uk

Nicholas Heard
and Niall Adams
Department of Mathematics
Imperial College London
London, UK
and
Heilbronn Institute, University of Bristol

***Abstract*—How can we effectively use costly statistical models in the defence of large computer networks? Statistical modelling and machine learning are potentially powerful ways to detect threats as they do not require a human level understanding of the attack. However, they are rarely applied in practice as the computational cost of deploying all but the most simple algorithms can become implausibly large. Here we describe a multilevel approach to statistical modelling in which descriptions of the normal running of the network are built up from the lower netflow level to higher-level sessions and graph-level descriptions. Statistical models at low levels are most capable of detecting the unusual activity that might be a result of malicious software or hackers, but are too costly to run over the whole network. We develop a fast algorithm to identify tunnelling behaviour at the session level using 'telescoping' of sessions containing other sessions, and demonstrate that this allows a statistical model to be run at scale on netflow timings. The method is applied to a toy dataset using an artificial 'attack'.**

I. Introduction

It is frequently the case that malicious activity on a computer network cannot be identified with certainty. For example, this can be because we do not have reliable signature for malware or because we cannot perform packet inspection so as not to intrude on the privacy of users. In the face of such uncertainty, we need to build up statistical evidence for malicious activity from a variety of sources. Fine-grained statistical models for how the network should be behaving would be helpful, but these are computationally expensive and cannot be run on all edges of the network simultaneously. Here we consider a 'big data' modelling strategy that allows these tools to be used, by deciding whether to run them based on simpler models. We give an illustration on artificial attack data showing that combining models across levels can detect a malicious attack.

To illustrate the value of statistical modelling, we consider the problem of detecting malicious activity on a network using network 'flows' [1] (netflow). This is a hard and important computational problem in cyber defence [2]. Specifically, we consider detecting tunnelling [3], [4] in the case where the attacker has obtained *legitimate* credentials. Because an attacker may be detectable if they do not follow existing connections, they must create tunnels from their entry point to the target. Legitimate users are unlikely to need to do this, so the attack can still be detected. This is a very difficult statistical problem, requiring careful models for timing information. The key observation is that network activity is 'telescoped' [5], that is, activity on outgoing edges of a node is entirely contained within incoming activity. Further, the timing of such netflow events are correlated. Finding these correlations in principle requires examining all edges with a complex statistical model. This computation is, in general, infeasible.

To address the issue of computational complexity, we propose a framework in which high level data summaries are used to identify a limited number of candidate edges. This 'multilevel modelling' approach (e.g. [6]) makes several levels of modelling approximation, with each level promoting the most promising candidates for further analysis to the level below. A full computation at the netflow level is only performed for a heavily filtered dataset.

A key aspect to the use of multilevel modelling is how information between levels interacts. If levels are strongly correlated then there is no need to perform detailed modelling as summarised high level data is sufficient. Conversely, if the correlation between levels is weak, the high level summaries are not useful. To overcome this erosion of statistical power, transitions between levels are constructed to be 'conservative', that is, we only filter [7] information that is certainly not of interest. We move beyond filtering by exploiting an 'interestingness ordering' of the remaining data.

We demonstrate our approach on computer traffic generated by the Imperial College network. Imperial College London is a leading university in the UK and its associated computer network contains 345098 IPs and hundreds of millions of edges. To demonstrate the value of the approach, we have created a telescoping session by making a sequence of successive SSH sessions, chosen to resemble an infiltration path ([8] describes a real-world case). We have demonstrated that this tunnel can be detected with our approach. Further, the described methodology can be used with other models.

II. Modelling

A. Models for cyber security

Figure 1a describes the relationship between the modelling levels we use for different scales of abstraction.

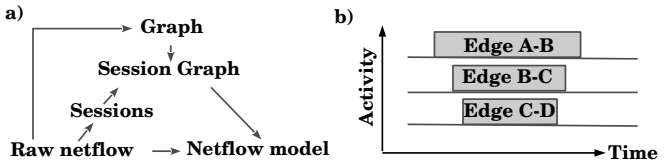

Fig. 1. a) Conceptual relationship between the conceptual levels in our framework. b) Schematic of telescoping sesssions in a 4-chain consisting of the four nodes A-B-C-D, and hence three edges.

1) Graph level: We conceptualise the graph of which nodes communicate in the network. As is routine in network analysis, we use the network structure to limit the space of computations we consider by only performing computations on edges that share a node. The graph level can be modelled for interesting behaviour (e.g. [9]) but we do not pursue this here.

2) Session level: Communications in practice are highly bursty, i.e. an edge is much more likely to see traffic when it has recently experienced traffic. There are many reasons for this, with the most important being the presence of a user. Users create netflow by use of their system, and create changes in the automated behaviour of their computers by running processes in the background. User-generated activity manifests differently to automated computer activity, which often occurs regularly without intervention. Examples are DNS (Domain Name System) traffic, communication between servers for idle services, refreshing website content, etc. For effective correlation analysis, it is extremely important to model whether traffic is directly user generated. This view of sessions extracts out a) sets of flows that will be modelled together (each session), and b) flows that are not of modelling interest.

An additional complication is that router and protocol features split raw netflow events in an unpredictable manner. See [10] for more details of how this can be modelled. To address this simply, we use 'time-out' sessionization, defining netflow records on the same edge as part of the same session if they occur within $T = 125$ seconds of each other. This is arbitrary and should be modelled in real applications.

3) Telescoping Session Graph level: Tunnelling directly causes telescoping of sessions. Telescoping here means that an 'incoming' session to a node B entirely contains an 'outgoing' session (Figure 1b). Finding telescoping sessions has a simple algorithmic solution. However, we must also account for the possibility that sessions contain each other by chance.

We can form a graph of all telescoping [5] and hence interesting sessions. This will be a massive simplification of the original graph. Whilst there will be some legitimate reasons for users to tunnel through a single machine, these are typically constructed by the network administration. Any other chains in the telescoping session graph are therefore of interest.

Let X be a session, described by a source node $N_1(X)$, a destination node $N_2(X)$, a start time $T_s(X)$ and an end time $T_f(X)$. The duration of a session is $D(X) = T_f(X) - T_s(X)$.

Definition 1: Telescoping Sessions. Session i is 'telescoping' session j if:

1) The start time of j is 'shortly after' the start time of i: $T_s(X_i) - a_1 < T_s(X_j) < T_s(X_i) + a_2$.
2) The end time of j is 'shortly before' the end time of i: $T_f(X_i) - a_3 < T_f(X_j) < T_f(X_i) + a_4$.
3) Node B is the destination of session i, and the source of session j: $N_2(X_i) = B$ and $N_1(X_j) = B$.

The tolerances a can depend on X. We set $a_1 = 0$, $a_2 = a_3 = \infty$ to retain all telescoping sessions however different in size, and $a_4 = 4.5$ seconds based on the empirical reliability of timings between our routers.

Algorithm 1: Find all telescoping sessions for node B. Sort all events by 'effective start time', $T_s(X) + a_1(X)$ for incoming sessions ($N_2(X_i) = B$) and $T_s(X)$ for outgoing sessions ($N_1(X_i) = B$). Then iterate over sessions i:

1) If $N_2(X_i) = B$ (incoming session), store X_i with an expiry time $E_i = T_s(X_i) + a_2$.
2) Else $N_1(X_i) = B$ (outgoing session):
   a) Expire all incoming sessions j for which $E_j > T_s(X_i)$, i.e. which have passed their expiry time.
   b) For all remaining sessions j check if $T_f(X_j) - a_3 < T_f(X_i) < T_f(X_j) + a_4$. If so, record the session pair.

This algorithm is linear in the number of sessions and quadratic in the number of unexpired incoming sessions active at a given time. This is limited to the number of IP addresses that are making overlapping incoming connections, and is in practice much smaller than the indegree of a node.

Further, we can easily compute all L-*chains* (consecutively telescoping sessions involving L nodes; Figure 1b shows a 4-chain) and L-*paths* (nodes involved in L-chains) in the dataset. Firstly, assign each session an ID and represent a telescoping session as a pair of session IDs. Finding telescoping sessions of length L only requres checking, for all chains of length $L-1$, whether final session k is a penultimate session in another chain. We iterate over L from 3 upwards until there are no chains remaining. Because only chains containing node B can chain with nodes ending in node B, relatively few comparisons are required. Finally, we construct L-paths by considering all unique chains of length L.

We want to focus attention on the sessions most likely to be tunnelling. For this, each 3-path on edges l, m is scored by the probability p_{lm} that the telescoping occurred by chance. To properly calibrate this requires an understanding of the seasonal nature of each edge, and the correlation between events. However, we can obtain a cost efficient estimate by a) resampling the session durations with replacement, and b) assuming a uniform distribution on the start times. This leads to a probability of each session pair telescoping of $f_{ij} = (D(X_i) + a_1 + a_4 - D(X_j))/T$ for incoming sessions i and outgoing sessions j, where T is the observation time window. Because we resample with replacement, a randomly chosen session telescopes with probability

$$f_{lm} = \sum_{i \in l} \sum_{j \in m} f_{ij} / M_l M_j.$$

Finally, we assume independence between each telescoping pair, which is approximately true if sessions are short relative

to the overall time. Under these assumptions the number of observed nesting sessions n_{lm} is a binomial random variable

$$n_{lm} \sim \text{Bin}(M_l M_m, f_{lm}).$$

Therefore $p_{lm} = p(n_{lm} \geq n_{lm}^{obs})$ is a p-value for the probability that the sessions telescope by chance. p_{lm} can only take finitely many values, and so it is a discrete random variable. However, in this case $p(p_{lm} \leq \alpha) \leq \alpha$ meaning that the p-value is conservative.

Although it is possible to compute the probability of an L-chain similarly, we simply use Fisher's method to combine p-values. We note that since long chains are of intrinsic interest, it is not necessary to compare between length classes.

4) Netflow level: Modelling at the netflow level is extremely difficult. Every protocol behaves differently and should be treated accordingly. Statistical analysis will be most powerful for the netflow events directly, but for convenience we follow many authors (e.g. [2]) in binning activity over time. We consider the total number of flows to give the observed number of flows F_{it} per minute t on edge i. We then consider the log-transformed variable $x_{it} = \log(F_{it}+1)$ as a standardized time series. The 'netflow level score' for a pair of edges l and m is

$$s_{lm} = \text{cor}(x_{it}, x_{jt}),$$

over a period identified by the session algorithm as of interest. This illustrative score is sufficient for our example and is a placeholder for a more complex statistical model.

B. Big Data Statistics

The key statistical challenge is to relate the session features to the netflow level. Although each score should, in principle, be positively correlated with 'interestingness' at the netflow level, it is far from clear how to combine the scores in order to make the best prediction about the value to be gained from performing netflow modelling. In [11] we discuss how scores can be mapped across levels, and the benefits this can lead to. In [12], we discuss some general principles applicable to the cyber problem. Here we focus on making a working framework without quantification of how s_{lm} and p_{lm} relate. A negative correlation (present here) implies that the smallest p_{lm} and the highest scores s_{lm} are interesting.

The session level can be constructed in a single pass of the data. The 3-paths from the telescoping session graph can be run efficiently in a Hadoop streaming environment (hadoop.apache.org) using the MapReduce [13] framework: keys are created in the map phase by duplicating each session X to produce a key:value pair $N_1(X) : X$ and $N_2(X) : X$. They are then sorted using a 'secondary sort' and then the reducer performs Algorithm 1. The telescoping p-values p_{lm} can be computed in the same pass with a modest memory overhead. L-paths are then constructed iteratively; a MapReduce algorithm could also be developed for this problem.

In a 'Big Data statistics' framework containing too many telescoping 3-paths, we could find successively more interesting telescoping sessions. The parameters a_i can be set to consider only very 'tightly' telescoping sessions. We can then iteratively repeat the analysis with less restrictive conditions. This approach first examines sessions of similar duration which are more likely to be tunnels. We could additionally make this explicit by defining a 'tightness score' $s(X_i, X_j)$ and run the iterations to capture ranges of this score. This can quickly obtain interesting results for promotion to the netflow level.

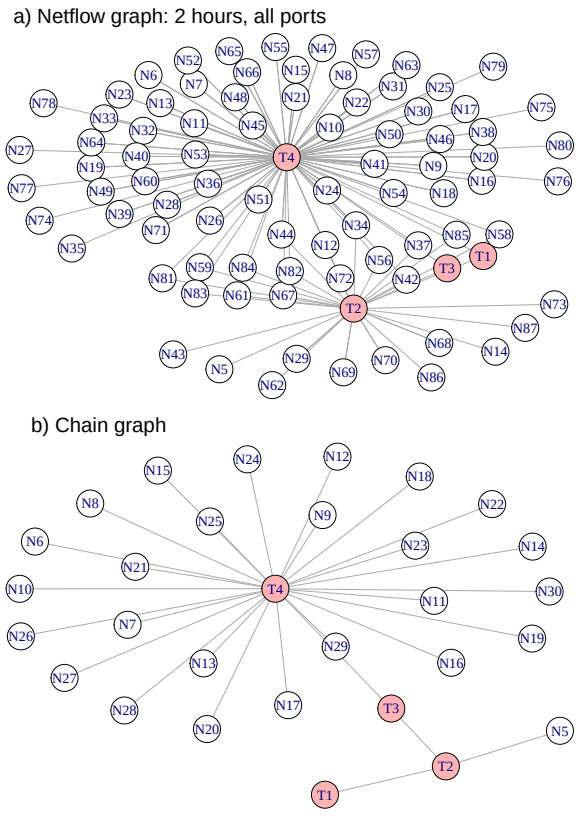

Fig. 2. Graph of netflow connections containing either N2 or N4 in Study 2, showing our inserted tunnel from T1-T2-T3-T4 in red and other nodes NX in white. a) Netflow graph, containing the tunnelling event. b) Telescoping session graph for 5-paths.

III. Results

To test our approach, we generated an artificial 'attack' by creating an unusual SSH tunnel, for which we had credentials. From a home PC (T1) we used SSH to access a mathematics server (T2) at Imperial college, onwards to a Bristol University server (T3), and back to a machine at Imperial (T4). Activity on T4 included installing the Google Chrome browser to simulate events which change its behaviour, and streaming a video to generate a correlated data stream along the tunnel, as might be observed in data exfiltration. The Imperial College netflow records were obtained for T2 and T4 for 48 hours containing the event and we tried to identify a) the path and b) the time of the tunnelling event.

In study 1, we examined the 6569 SSH netflow records over 48 hours. Many of these records are failed port scanning activity. Our telescoping algorithm finds two 4-paths in this dataset. These are N6-T2-T3-T4 (p-value 0.0013) and T1-T2-T3-T4 (p-value 0.00011). Because T2 is a busy server a session overlapped by chance with our procedure, but as N6-T2 was a busy edge it has a downweighted p-value relative to the true path (by a factor of 10).

In study 2, we examine the 4698 netflow events of all protocols and ports for a 2 hour window containing the true

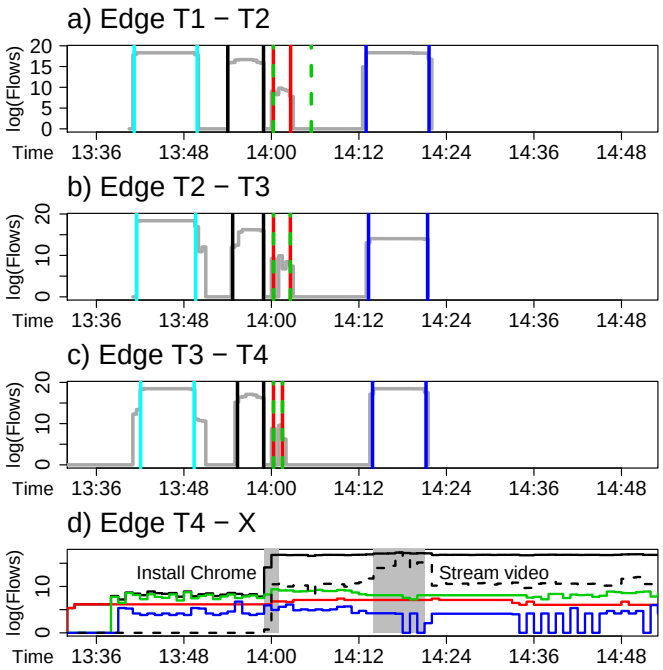

Fig. 3. The path we inserted and was located by our algorithm shown as edge activity (log number of flows per second) over time. a–c) have the inferred times for the 4 telescoping sessions marked as vertical lines. d) shows the top 4 outgoing edges from T4, ordered by correlation with edge T3–T4. The black (highest) curve is Google which is also shown unlogged as a dashed line (normalised). Two important actions were performed through the tunnel and are marked in d); firstly, installing Google Chrome, and secondly, streaming a Youtube video (from Google). Correlations are $(0.91, 0.57, 0.44, 0.39)$ for the black (Google), red, green and blue IPs respectively.

event, as shown in Figure 2a. There are 27 5-paths in this data, giving the Telescoping session graph in Figure 2b. All 5-paths contain the 2 SSH 4-paths found above and end with a variety of servers. Much of this is background desktop activity such as maintaining connections to websites in an active browser, dropbox, and DNS.

Figure 3 shows this path in raw netflow events, as well as telescoping of sessions. Our injected activity on N4 cannot be easily determined from the session level data. However, using the session level results as a prioritisation for the netflow level modelling, we can identify what actions were taken using the tunnel. Figure 3d shows that we have correctly identified a Google IP (black line, correlation 0.96) involved in both the behavioural change and the data streaming. There are no other netflow events moving large volumes of data into T4 during our streaming video action. Stage 1 achieved a data reduction of 100 but more importantly made the tunnelling activity clear. Stage 2 informs about the activity occurring during the tunnelling event.

IV. Discussion

Finding anomalies in netflow data is a very difficult challenge, with modelling still in its infancy (e.g. [14]). This work demonstrates that models do not have to scale intrinsically to be used as part of modelling massive data sets. Further work is required to refine the proposed models - both of session overlap and netflow models - into deployable tools. However, we emphasise that our framework permits the use of complex statistical methodology (such as [15]).

Our algorithm for detecting 'telescoping sessions' has linear cost in both the number of netflow records and nodes. We provided an implementation that can be run on massive data platforms such as MapReduce, and can in principle be run in real time on streaming data. Further, the algorithm is effective at finding SSH tunnelling which is helpful in the defence of large networks.

We have described a two stage algorithm for performing a statistical analysis on large volumes of netflow. This operates by first using a fast $O(N)$ filtering algorithm, designed to discard data conservatively, in order to focus the computation of a costly statistical model to the data of interest. This could a) find the tunnelling event, b) give a probability of a true 'attack' compared to a chance event; and c) make statistical inference about the activity performed during that tunnelling.

Acknowledgment

We gratefully acknowledge Imperial College ICT services and Andy Thomas for assistance gathering the netflow data.

References

[1] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," *Communication Surveys & Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.

[2] 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.

[3] W. Ellens, P. Żuraniewski, A. Sperotto, H. Schotanus, M. Mandjes, and E. Meeuwissen, "Flow-based detection of DNS tunnels," in *Emerging Management Mechanisms for the Future Internet*. Springer, 2013, pp. 124–135.

[4] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Detecting http tunnels with statistical mechanisms," in *ICC'07*. IEEE, 2007, pp. 6162–6168.

[5] H. Djidjev, G. Sandine, C. Storlie, and S. Vander Wiel, "Graph based statistical analysis of network traffic," in *Proceedings of the Ninth Workshop on Mining and Learning with Graphs*, 2011.

[6] N. A. Heard, D. J. Weston, K. Platanioti, and D. J. Hand, "Bayesian anomaly detection methods for social networks," *The Annals of Applied Statistics*, vol. 4, no. 2, pp. 645–662, 2010.

[7] A. Cuzzocrea, I.-Y. Song, and K. C. Davis, "Analytics over large-scale multidimensional data: the big data revolution!" in *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*. ACM, 2011, pp. 101–104.

[8] 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.

[9] C. E. Priebe, J. M. Conroy, D. J. Marchette, and Y. Park, "Scan statistics on enron graphs," *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 229–247, 2005.

[10] P. Rubin-Delanchy, D. J. Lawson, N. Heard, and N. Adams, "Three views on netflow session," 2014, Heilbronn Institute Technical Report.

[11] 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.

[12] N. Adams and D. J. Lawson, "An approximate framework for flexible netflow screening," 2014, Heilbronn Institute Technical Report.

[13] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[14] N. Adams and N. Heard, *Data Analysis for Network Cyber-Security*. Imperial College Press, 2014.

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