PATH TRACING IN TOR NETWORKS

Nick Johnson, Steve McLaughlin, John Thompson

Institute for Digital Communications
Joint Research Institute for Signal & Image Processing
School of Engineering
The University of Edinburgh
Edinburgh, EH9 3JL, UK
{Firstname . Surname}@ed.ac.uk

ABSTRACT

Internet based communications methods use channels which cross many nodes in their routes between source and destination. The user often has little or no control in the routing process and may be concerned that the medium is insecure. To combat this, privacy preserving networks have been developed in an attempt to allow secure, private internet-based communication. Whilst these networks employ high levels of encryption between nodes, it is possible to track a users data by correlating input and output streams. This paper addresses the implementation of the algorithm first proposed by Danezis for determining the exit node of data injected into a privacy preserving network, such as Tor. The algorithm is discussed along with some modifications and assumptions necessary for implementation. Results gained from applying the algorithm to data from a real Tor network are presented and discussed.

1. INTRODUCTION

A privacy-preserving network (alternatively called an anonymising network) such as The Onion Router (Tor) [1] [2] [3] [4] is designed to give some measure of privacy to those using the internet who wish to conceal their activity from observation. Tor is designed to provide anonymity by encrypting data between the end-points of a route. It is used for internet-based communications and operates on a large number of machines, mostly those of individual users.

Tor is an overlay protocol and uses an underlying layer of transmission control protocol (TCP) / internet protocol (IP) to handle data transport, delivery and routing. The small volume of centralised control which exists in any Tor network (including the default internet-based network) comes from the central directory servers. These maintain the *state* of the network and collect and collate data such as which nodes are suitable for use as exit nodes, their uptime and any bandwidth restrictions imposed by the node operators. This information allows Tor to determine a choice of route for a specific connection based upon user requirements. Traffic to and from a directory server uses a different port to that of the payload traffic and can be easily separated.

There are three types of node commonly encountered in a Tor network. Exit nodes - which send traffic un-encrypted to its final destination, entry nodes - which accept un-encrypted traffic, encrypt and forward it into the network and routers - which forward traffic between Tor router nodes. The entry and exit nodes are generally the end points of any Tor communication. There are a large number of possible configurations but it is most common for each user node to be an entry, router and exit node.

In a typical usage scenario, the user configures their browser to route traffic to the destination (i.e., a desired web-server) via Tor. The user's node becomes a node in the default internet Tor network and their traffic is routed to the exit node nearest (i.e. with *best* connection to) the destination. The security of the traffic is maintained between the user's node and the exit node via layered encryption.

Danezis [5] proposed an algorithm for tracking users in a privacy preserving network. His approach allowed him to determine the route and probable end-point of communications in such a network by correlating traffic signatures at entry and exit nodes. Fundamentally, his approach is based upon detecting a known signal from amongst a selection of possible noisy signals. The input signal (known *a-priori*) and an estimate of each possible output signal are correlated to determine which output signal most likely contains the input signal. Each estimated output contains noise (generated by the system) and may be distorted by the estimation process. Results presented in this paper confirm Danezis's work using a physical Tor network with real data.

Work towards a similar goal has been performed by Murdoch and Danezis [6] which assumes a corrupted node in the Tor network is available for use by the attacker. This method is active in its approach but has the advantage of being able to operate without complete knowledge of the network. The correlation (template in their nomenclature) function employed is similar to that used in this paper. Another active method of identifying relay nodes is presented by Chakravarty *et al* in [7]. Other works relating to traffic analysis attacks of privacy preserving networks [8] which are similar in nature have been published by Zhu *et al* [9] and Levine *et al* [10]. Syverson *et al* focus directly on Tor in [11]. In [12], Bai *et al* seek a method to identify Tor network traffic from amongst a mix of traffic.

The remainder of this paper is organised as follows: Section 2 introduces the Danezis algorithm and discusses the modifications and assumptions necessary for implementation; Section 3 introduces the system testbed and describes how the data was generated and gathered; Section 4 gives some notes

This work was funded by Agilent Technologies.

 ISSN 2076-1465

on how the performance was assessed; Section 5 presents some results and interpretation and finally Section 6 presents some conclusions.

2. DANEZIS'S ALGORITHM

In [5], Danezis proposes a method for attacking privacy preserving networks such as Tor based on the correlation of input and output packet streams. He assumes that a Tor network can be modelled as a delay mixing model [13] which adds delay to incoming packets in a predictable manner before ejecting them. He indicates that it is possible to deduce the exit node by calculating the correlation between the input stream and a number of possible output streams and selecting the exit node whose output stream has highest correlation.

More specifically, what Danezis's algorithm states is: two time-series can be estimated, with lengths equal to that of the observed output streams and where each estimate is composed of the input stream, delayed by some function, added to a uniformly distributed background traffic stream (which models network traffic that arises independently of the presence of the input stream). The background stream rate is adjusted such that the combined output rate is equal to the rate of the observed stream. If one was to consider these estimated series only at the instances corresponding to observations of packets in the true output series then a more accurate estimate would be made. Division of the estimated series will indicate which output stream most likely contains the input stream.
The formula given by Danezis to estimate any output traffic distribution is shown in (1) with an explanation of the parameters used given in Table 1. One element which is not generally clear is the uniform distribution $U(t)$ and how it is computed: $U(t)$ is the uniform distribution in the interval $[0,T]$ and as such it conforms to (2). In any practical implementation, t will be discrete so the integral may be replaced with a sum.

$$C_X(t) = \frac{\lambda_f(d * f)(t) + (\lambda_X - \lambda_f)U(t)}{\lambda_X} \tag{1}$$

$$\int_0^T U(t) = 1 \tag{2}$$

$d(x)$ is a function which represents they input/output relationship of the network in terms of packet delay, a temporal transfer function for packets. Given that this function can change over time as the network changes or as Tor changes its routing it is more practical to estimate it empirically using some training data. If access to individual nodes is possible then one can make the assumption that a given number of packets travelling on one link will have a delay distribution similar to packets on any other link, but with a change in scale. The use of a Gaussian mixture model (GMM) as an estimator for the distribution of $d(x)$ is therefore appropriate and is what is used in this work.

Once an estimated distribution (C_X) has been computed for each output stream then they can be compared to determine which is most likely to contain the input stream. If C_Y denotes a second output stream emanating from node Y, $X_{i=1\ldots n}$ denotes the set of times that packets are observed at node X, $Y_{j=1\ldots m}$ denotes the set of times packets are observed at node Y, H_0 denotes the hypothesis that the input stream in contained in the output stream from node X and H_1 denotes the hypothesis the input stream is contained in the output stream from node Y, then it is possible to calculate the likelihood ratio of the two hypothesis as shown in (3).

$$\frac{\mathscr{L}(H_0|X_i,Y_j)}{\mathscr{L}(H_1|X_i,Y_j)} = \frac{\prod_{i=1}^n C_X(X_i)\prod_{j=1}^m u}{\prod_{i=1}^n u\prod_{j=1}^m C_Y(Y_j)} > 1 \tag{3}$$

Equation (3) can be cumbersome to compute numerically and leads to large values which suffer from rounding errors when implemented. However, it can be manipulated into a log-likelihood form as shown in (4) which reduces the scope of the possible values and so this form is used instead.

$$\log \mathscr{L}_{H_0/H_1} = \sum_{i=1}^{n} \log C_X(X_i) - \sum_{j=1}^{m} \log C_Y(Y_i) + (m-n)\log u > 0 \tag{4}$$

In his paper, Danezis gives no source code and little implementation detail so the following assumptions are made:

1. It is assumed that the algorithm is functional when implemented in a discrete form. It is originally presented in a continuous form which does not lend itself to easy implementation. Variables such as C_X are treated as discrete vectors by applying a binning process to the continuous time series data which arises from measurement. This implies that C_X depicts packet counts over the segment of interest with t being the bin index. Similarly, $f(t)$, $d(x)$ and the other estimated series (C_Y etc.) are binned with the same resolution and thus X_i and Y_j are discretized. Perhaps more intuitively C_X, $f(t)$, X_i and Y_j can be thought of as packet counts per unit of time where t indexes the time interval and where the temporal resolution is equal for all variables including $d(x)$.
2. It is assumed that the algorithm is robust to changes in the scaling of t such that it is possible to vary the temporal resolution of t and therefore the bin width of C_X with the caveat that the temporal resolution must be consistent across all variables.
3. It is assumed that Tor traffic can be separated from other traffic at any node. The Tor traffic which flows from one exit node to any other Tor node (to pass acknowledgements (ACK) back upstream, for example) will be highly correlated with the exit stream and may disrupt the algorithm.
4. It is assumed there is enough training data to model the delay function, $d(x)$, using a GMM of three elements; should it be found that a three element mix has redundancy, the order is reduced to two or one. Tor incoming packet delays are measured at all exit nodes for packets originating at other Tor nodes and it is assumed that the statistical distribution of these packets is representative of the delay distributions experienced by packets exiting the Tor network.
5. It is assumed λ_f (and subsequently, λ_X, λ_Y, etc.) is computed as the mean number of packets per unit time (one unit is the elapsed time between time index t and time index $t+1$) in the interval $[0,T]$. Thus, if there are 2000

Parameter	Meaning
λ_X	The rate of packets in the interval in question exiting node X
λ_f	The rate of packets in the interval in question in the input stream $f(t)$
$U(t)$	The discrete uniform distribution in the interval in question
u	The value of $U(t)$ at any t
$d(x)$	A function describing the delay mix of the network
$f(t)$	The input signal (stream)
$C_X(t)$	An estimate of the number of packets in the estimated output stream at t
t	The time index
$(d*f)(t)$	The convolution of the input signal with the delay mix function

Table 1: Parameters used in Danezis's formulae.

packets in an interval of 25 seconds with a unit size of 0.1s then the rate is $2000/(25/0.1) = 8$ packets per unit time.

3. THE TOR TEST NETWORK AND DATASET GENERATION

In this work a small Tor network over which full control and monitoring capabilities are available[1] is used as a testbed with which to generate a dataset to test the algorithm. There are four nodes in the network: three are directory servers, routers and exit nodes (node 94, node 104 & node 110) whilst the remaining one is solely an entry node (node 93). The network is restricted such that the entry node cannot be the exit node and therefore the exit must be one of nodes 94/104/110. The exit node is fixed to be one node (node 94) for the duration of any measurements which allows it to be used as a ground truth against which any estimates the exit node of the network can be evaluated. A topology of the network showing the nodes and data flows is shown in Figure 1.

The dataset was recorded using `tcpdump` [14] [15] on the network described above for a period of twelve hours beginning at approximately 1700 hours to minimise the effect of any user traffic on the dataset. The dataset was split into twenty four contiguous segments of thirty minutes which enables averaging of results across segments in order to reduce the impact of any unusual network circumstances.

Network traffic was generated by directing a web-browser (Konqueror) to access the home-page of the BBC news website (http://news.bbc.co.uk) at timed intervals. The intervals are on the order of seconds and randomly generated as the result of selecting a random number from a Poisson distribution with parameter equal to 30. Using a Poisson distribution removes the periodicity which would be encountered with a uniform process but retains some element of regularisation. Algorithm 1 shows the traffic generation method.
Filtering was performed to isolate packets serving different functions by port number and included removal of any SSH packets (port 22) which were part of the recording or monitoring processes (such as status indicators).

The traces were processed using a range of AWK scripts to extract packet delays and time series. Once processed, traces were loaded into MATLAB for further processing. The data were first scaled by the trace start time i.e., subtracting from all elements in the series the lowest value before using the `histc` function to convert from a series of timestamps to a binned representation of the data. The scaling allows the binned respresentation to be computed with an arbitrary resolution for any segment without having to change the resolution of the whole dataset, i.e. it makes the segments independent of the time they were recorded.

[1]This implies that is it possible to inspect any of the traffic flowing into or out of the nodes and that it is possible to configure Tor in any manner desired.

Algorithm 1 Network traffic generation

1: **loop**
2: Generate *interval* using Poisson-based random number generator with parameter equal to 30.
3: Start web browser and access web page via Tor network.
4: Wait for *interval* seconds.
5: Kill web browser.
6: **end loop**

4. TEST METHODOLOGY

In order to test the algorithm against the dataset, some standardized methodology must be used. In this paper, each of the 30 minute segments are examined individually and the overall accuracy of identification (of the correct exit node) is expressed out of 24, the total number of segments.

In (3) the forumla for computing the most likely output node from two choices is shown, however, a solution is required for the case of $\geq$2 nodes which is scalable. Whilst it would be possible to use a tree-search type of algorithm, it makes sense to use a ratio combining algorithm. Define the ratio A to be the ratio of the estimated distributions for nodes 94 and 104; B to be the ratio for nodes 104 and 110 and C to be the ratio for nodes 94 and 110. Recall the estimated distributions as being the instantiations of (4) with the relevant data. It is then possible to define the following:

$$P94 = A + C \quad (5)$$
$$P104 = -A + B \quad (6)$$
$$P110 = -B + -C \quad (7)$$

By selecting $\max[P94, P104, P110]$ the most likely exit node for the data in question can be estimated; the difference between the selected node and the others gives some measure

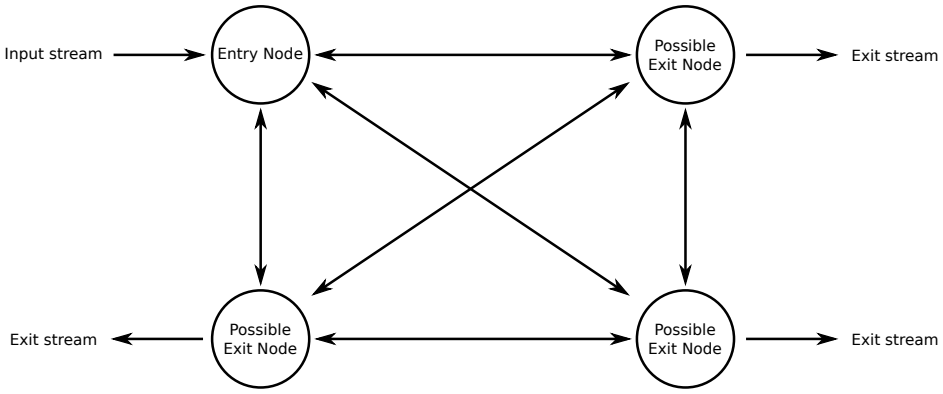


Figure 1: Network topology for Tor test network showing the entry node (with associated input stream) and the three possible exit nodes (with associated output streams).

of confidence in the decision; the larger the difference, the greater the confidence.

5. RESULTS

Before considering the complete dataset, it is informative to observe the operation of the algorithm upon one segment, segment 24, which is plotted below.

To begin, the input stream (Figure 2) is observed in isolation. The traffic appears near-periodic but with some variation which is expected given the Poisson-based generation method used.
Next, the output streams from each of three possible exit nodes are observed (Figure 3 (a)). Any packets travelling towards the target web-server have been removed to show only the background traffic at each exit node with a temporal resolution of one second. Node 104 is observed to send out a periodic stream of packets; as there is no Tor traffic or exit-stream traffic then it can be assumed this is either due to a user process running on this machine or, more likely, a network operation (backup, file handling etc). Node 94 sends out more frequent traffic than Node 104 and the lack of periodicity would indicate a user process (web browsing, email etc). Node 110 has little outgoing traffic perhaps because there were no active users during the time the traffic capture was in operation.
Finally, the outputs from each of the possible exit nodes are observed when the input stream is applied to the network (Figure 3 (b)). There is a large rise in the volume of traffic emanating from Node 94 with a near-periodic pattern which appears to be similar to the input stream giving an indication that this may be the true exit node.

The results for the complete dataset are shown in Table 2. It can be seen that the algorithm has correctly identified node 94 as the exit node in each of the twenty-four segments for three different temporal resolutions. Increasing the temporal resolution (ie, from 1s to 0.1s) increases the computational load but can be of help in situations where the input and output streams are of such a density as the binned representations have a 1 in each bin. This can disrupt the algorithm as the signature becomes masked at that resolution. Using a finer resolution can restore some degree of sparseness to the signature allowing the algorithm to function.

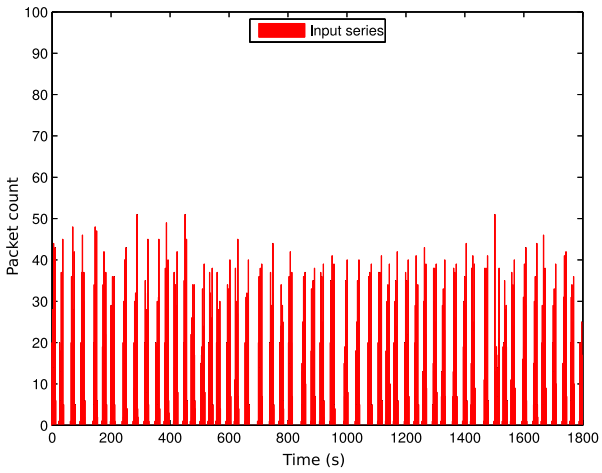
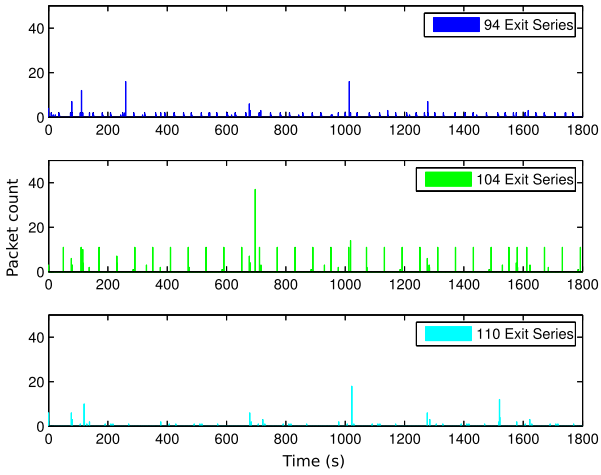


Figure 2: The input stream in isolation.

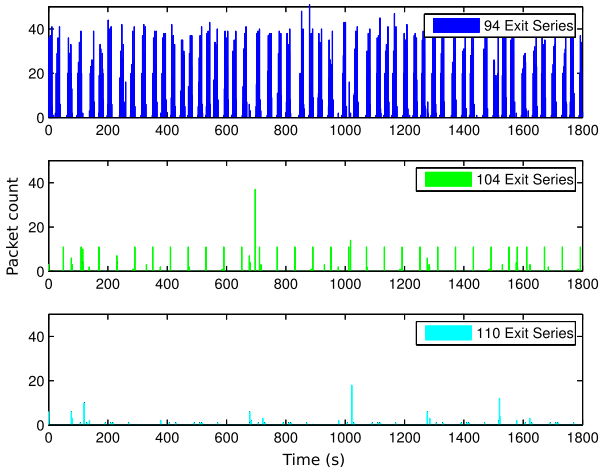
It should be noted that the delay function $d(x)$ was estimated as a vector with 1 in the first bin and 0 elsewhere for each of the temporal resolutions. This can be interpreted as an indication that the model for the delay is at a resolution smaller than the resolution at which the algorithm is being run. Delays on the order of milliseconds are observed but the algorithm is run at a resolution of seconds hence the above result. In the original paper, Danezis places some emphasis on an accurate model for this function but it would appear that in practise, a crude model is sufficient to generate results.

Resolution (s)	Node 94	Node 104	Node 110
1	24	0	0
0.1	24	0	0
0.01	24	0	0

Table 2: Results for the dataset. These indicate the number of times that each node is chosen as most likely exit node and are scored out of 24 - the total number of segments for this dataset.



(a) Network traffic observed at each node when no input is applied.



(b) Network traffic observed at each node when input is applied to the Tor network.

Figure 3: The output streams from the three possible exit nodes showing: (a) - no input being applied to the network, i.e. the background traffic and (b) - the result when the input stream is applied.

6. CONCLUSIONS

It has been shown that it is possible to remove the privacy afforded by the use of a Tor network by correlating time series representations of the input and output streams of the network. The algorithm proposed by Danezis is shown to work - albeit with some minor modifications to make implementation possible. The advantage of this method is that it does not require any attempt to break the encryption of packets performed by Tor, relying on signature matching, which, whilst crude, has been shown to work with real data gathered from an experiment upon a live Tor network.

REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson. TOR: The Onion Router. Tor Project/EFF. [Online]. Available: http://www.torproject.org

[2] ——, "Tor: The Second-Generation Onion Router," in *Proceedings of 13th USENIX Security Symposium*, August 2004.

[3] D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing," *Commun. ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[4] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous Connections and Onion Routing," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, May 1998.

[5] G. Danezis, *The Traffic Analysis of Continuous-time Mixes*, ser. Lecture Notes in Computer Science. Springer, 2004, vol. 3424, pp. 35–50.

[6] S. J. Murdoch and G. Danezis, "Low-cost Traffic Analysis of Tor," in *Proc. IEEE Symposium on Security and Privacy*, 8–11 May 2005, pp. 183–195.

[7] S. Chakravarty, A. Stavrou, and A. Keromytis, "Identifying Proxy Nodes in a Tor Anonymization Circuit," in *Signal Image Technology and Internet Based Systems, 2008. SITIS '08. IEEE International Conference on*, nov. 2008, pp. 633 –639.

[8] J.-F. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, H. Federrath, Ed., vol. 2009. Springer, 2000, pp. 10–29.

[9] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On Flow Correlation Attacks and Countermeasures in Mix Networks," *Lecture Notes in Computer Science*, vol. 3424, pp. 207–225, 2005.

[10] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing Attacks in Low-latency Mix Systems," in *Proceedings of the 8th International Financial Cryptography Conference (FC 2004), Key West, FL, USA, February 2004, volume 3110 of Lecture Notes in Computer Science*. Springer, 2004, pp. 251–265.

[11] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an Analysis of Onion Routing Security," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, H. Federrath, Ed., vol. 2009. Springer, 2000, pp. 96–114.

[12] X. Bai, Y. Zhang, and X. Niu, "Traffic identification of tor and web-mix," in *Intelligent Systems Design and Applications, 2008. ISDA '08. Eighth International Conference on*, vol. 1, 26-28 2008, pp. 548 –551.

[13] A. Serjantov and G. Danezis, *Towards an Information Theoretic Metric for Anonymity*, 2003, pp. 259–263. [Online]. Available: http://dx.doi.org/10.1007/3-540-36467-6_4

[14] LBNL Network Research Group. tcpdump. LBNL. [Online]. Available: http://www.tcpdump.org/

[15] ——. libpcap. LBNL. [Online]. Available: http://ee.lbl.gov