Information Security Breach Using Timing Information

M. S. Rosela Steffi[1] , Dr. R. Joshua Samuel Raj[2], P. Sivasamy[3]

[1]PG Scholar, [2]Professor, [3]Assistant Professor,

[#]Department of Computer Science and Engineering, Anna University-Chennai,

Rajas Engineering College, Tirunelveli, Tamilnadu, India.

Abstract* - This document is about a Information Security Breach using the Packet Timestamp information on the up-link and is mostly effective against Traffic streams since it doesn't need the start/end of web fetches. This attack has not been considered in most of the networks such, say VPN (Virtual Private Network), since most of the networks deal with Packet Padding defense. Using this attack there is a 95% chance of network compromisation, hence it lies on the Network Administrators to consider this type of attack while designing the network. Monte Carlo assessment of numerous statistical techniques meant to identify the subsistence of difference in computation time in remote web applications. We then execute a tool that allows infiltration testers to more methodically recognize possible exploit.

***Index Term*— network security, timestamp attacks, traffic analysis, monte carlo**

I. INTRODUCTION

In this paper we consider an attacker of the type illustrated in Figure 1. The assailant can detect the time when packets pass through the encrypted tunnel in the uplink direction, but has no other data about the clients' activity. The attacker's aim is to use this data to guess, with high chance of success, the web sites which the client visits. What is unique about the attack considered here is that the defenders relies solely on packet timestamp information whereas the previously attackers made use of packet size and packet count information.

There are two ways Firstly, packet padding is a relatively straightforward method against attacks that depend primarily on packet size, and implemented in a number of popular virtual private networks (VPN) [2]. Secondly, alternative attacks based on packet counting [2], are tactless to packet padding defences but require slicing of a packet stream into individual web fetches in order for the number of packets associated to be determined, which may be highly challenging in practice on links where there are no clear pauses between web fetches. In similarity, packet timing-based attacks are not affected by packet padding defences but also, as we will show, do not require slicing of the packet stream. Hence, they are potentially a practically vital class of attack against present and future VPNs. While some work has been carried out using inter-arrival time data to categorize the application such as HTTP, IMAP *etc.* [8], to our familiarity there is no previous work exposure use of timing information alone to construct a successful attack against encrypted web traffic.

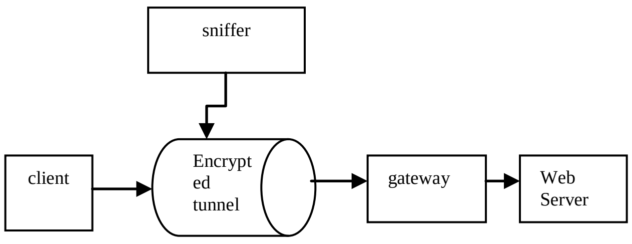


Fig. 1. A client machine is connected to an external network via an encrypted tunnel.

The main assistance of the pervious paper are as follows: (i) it describe an defense against encrypted web traffic that uses packet timestamp information , (ii) it demonstrate that this attack is highly ready to lend a hand against all kind of networks, achieving mean success rates in excess of 90% over ethernet and wireless tunnels and a success rate of 58% against Tor traffic, (iii) we also demonstrate that the attack is helpful against traffic streams In addition to being of interest in its own right, particularly in view of the powerful nature of the attack, this timing-only attack also serves to emphasize deficiency in accessible defences and so to areas where it would be valuable for VPN designers to hub additional concentration We note that, complementary to the previous work, in [3] it is established that when the web fetch borders within a packet stream are known then an NGRAM loom using packet count jointly with uplink/downlink direction information is also enough to construct an valuable attack against encrypted web traffic in spite of packet padding. Hence, we can conclude that (i) uplink direction and downlink direction packet ordering and also web fetch boundaries and (ii) uplink and downlink packet timestamp information are together responsive quantity that have to to be secluded by a secure encrypted tunnel. Packet padding does not guard these quantities.

II. RELATED WORK

Crosby et al. (2009), for example, edge on measure of individual packet round-trip times and evaluate several tests for distinguish timing difference. Their best performing classifier is the "box test," in which lower quantiles of 2 Web Timing Attacks Made Practical Morgan & Morgan each distribution of round-trip times are used in order to recognize difference between these distributions. subsequently, Lawson and Nelson (2010) offered their research at Blackhat 2010 where they too compare several statistical methods for distinguishing independent distributions of timing measurements. They settled on a percentile range approach alike to that described by Crosby et al. (2009).In recent times Mayer and Sandin (2014b) provided an updated analysis of what kinds of timing attacks might be practical in real-world systems. They also released a tool, Time Trial, which can help the typical penetration tester or vulnerability researcher determine if a timing difference exists on a remote system. In their study, the box test was the key method of distinguishing timing distributions.

The common issue of traffic scrutiny has been the subject of much concern and a large body of journalism exists. Some of the most primitive work specifically paying attention on attacks and defences for encrypted web traffic appears to be that of Hintz [5], which considers the SafeWeb encrypting proxy. In this setup (i) web page fetches occur sequentially with the start and end of each web page fetch (ii) the client- side port number, (iii) the direction (incoming/outgoing) and (iv) the size is observed. A web page signature is construct consisting of the amassed bytes received on each port effectively. In [6] it is similarly tacit that the number and size of the objects in a web page can be pragmatic and using this information a organization success rate of 75% is reported.

consequently, Bissias *et al.* [1] measured an encrypted tunnel setup where (i) web page fetches occur in sequence with the start and end of each web page fetch (ii) the size, (iii) the direction (incoming/outgoing) and (iv) the time is observed. The series of packet inter-arrival times and packet sizes from a web page take is used to produce a profile for each web page in a objective lay down and the cross correlation between an pragmatic traffic sequence and store profiles is then used as a gauge of similarity. A classification accurateness of 23% is observed when using a set of 100 web pages, going up to 40% when limited to a smaller set of web pages.

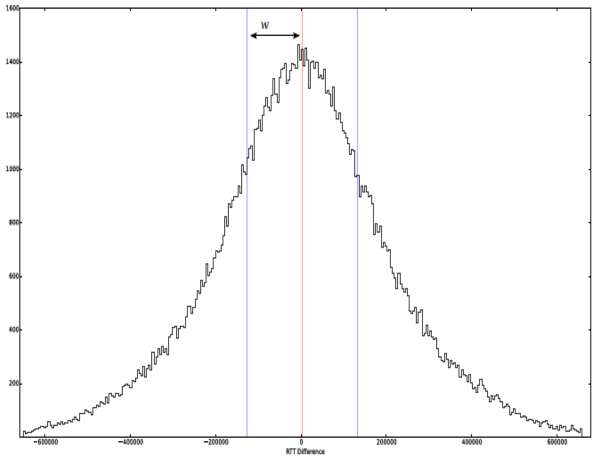
2.1 Timing Side-Channel Scenarios

Security testing community that an attacker can force a database query to pause for a period of time (either through explicit WAITFOR DELAY ... calls or computationally intensive operations). This is often sufficient to prove that a SQL injection flaw exists, but to extract information from the database quickly would require accurate timing measurements of specific requests. A penetration tester could try to speed up an attack by plummeting the holdup they make in the query processing, but how small of a delay can be accurately measured with a minimal number of requests? Without tools to conduct a statistical profile of the application's response time variability, one is likely to spend too much time obtaining only limited results or obtaining inaccurate information due to misinterpret timing delays.

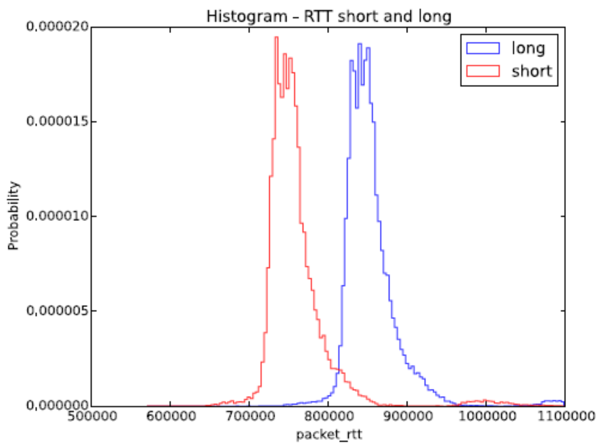
application being tested could abruptly come under heavy load from outside users. In both of these situation, individuality of the RTT distributions can change significantly, complicate the succeeding recognition of any timing differences. We move toward the statistical analysis of RTT distributions from numerous angles. In one move toward, we expand a set of tests, which attempt to accurately approximation the central tendency of the distribution of pair-wise round-trip time differences. If the central tendency of difference is clearly non-zero, in a statistically significant way, then we can say with some self-assurance that a difference in timing exists. In another move toward, we employ a Kalman filter, which is a robust statistical technique commonly used on data sets .

2.2 Central Tendency of Pairwise RTT Differences

The timing capacity for each test case as detach distributions, we gather our samples for each test case as pairs2 .The aim of doing the set in pairs was to decrease the hetero geneity in the distributions of the two test cases, thus making them more similar than if we had composed them in series. For instance, presume we had first determined to gather the set of long-case comments, Following this collection approach, a systematic change in the network circumstances during any of this era would make the two distributions of the RTT data unique. By bring together the data in pairs, however, a ethodical change in the network would affect succeeding pairs of samples; i.e., round trip times within pairs would remain similar With these sample pairs in hand, we then calculate the distinction in round-trip time for each pair of needs. The distribution of these differences turns out to be quite symmetric about the mean and allow us to use more traditional statistical techniques.regard as the histogram of test case data in Figure 2.Each distribution is very similar, but obviously different in their central tendency. Neither is symmetric. owever, if we measure the difference in round-trip time for each pair of RTT measurements, we obtain the distribution in Figure 3. This is clearly much more symmetric than the individual RTT distributions. The fundamental question to be answered in an analysis of the distribution of differences is whether the central tendency of this distribution different than zero.



This is, of course, a common task in statistics and the symmetric nature of the distribution allows us to appeal to standard measures of central tendency; e.g., mean or median. Additionally, working with the distribution of pair-wise differences also makes the analysis more resilient to changes in network and host conditions. As an example of this, during one early test over the Internet, we encountered a data set with a distribution of that shown in Figure 3.



This was initially a confusing result, as it wasn't clear what would generate such an extremely multi-modal data set. However, after reviewing the round trip times as a function of time-of-day, Here, it appears we fell victim to a feature of Comcast's Internet service: PowerBoost™. This feature is designed to allow subscribers to download the first small portion of a file at unlimited speeds (as fast as the infrastructure permits), but once some threshold is reached, Comcast resumes throttling the user to the rate they are subscribed at. It appears this is what had occurred, since the first small portion of the test ran with much less delay. Figure 3: Histogram of Pair-Wise Differences in RTT for Short and Long Computationscentral tendency that describes the difference in application server processing time. This is clearly shown by the purple "difference" series in the above plot. While the variance of data in each case may change, at least all data is usable to approximate the critical computation delta.

2.3 TCP Timestamp Measurement Approach

One advance for leveraging TCP timestamps to measure round-trip times, as mentioned by Lawson and Nelson (2010) in their 2010 Blackhat presentation, involves trying to synchronize the local system clock with the clock of the target system. Samples would then be collected in some small amount of time before the target host's TCP timestamp clock . If the resulting response did contain an increment in the timestamp clock, then one can conclude the response time was greater tb, otherwise it was smaller. Under perfect conditions, one could use a binary search to quickly determine the server processing time. However, to successfully perform this measurement, one would need to have extremely accurate measurements of both the target clock rate, the target clock offset (when, in terms of local time, the clock changes), and be able to submit requests at very accurate times. Lawson and Nelson (2010) rejected this approach due to the perceived implementation difficulties and we currently agree that this approach is likely impractical.

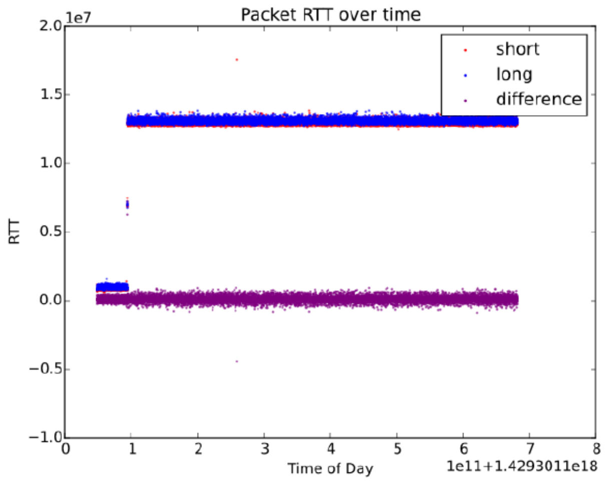


Figure 4: Histogram of RTT Values for each Test Case of them showed no growth in the 1ms clock,

However, an significant examination about low-precision clocks is that they can still be used to compute very small time differences statistically with no association, so long as one can acquire many samples. For example, if a clock measures time in increments of 1ms and we wish to measure the time of some event that takes 0.1ms, then we could gather those samples at random points in time and count the number of samples where the low-precision clock incremented by one. If the sample were calm in an qually-distributed manner over a long period of time, then one would anticipate approximately 1 in 10 samples are taken are not calmly distributed over the clock precision time frame (that is, the sample time-of-day modulo the clock precision is not evenly distributed), then this can rapidly tilt the results. Even with these obvious difficulty, TCP timestamps do offer the swear of potentially negating

Figure 4: Box Test — Indistinguishable Distributions

As discussed earlier, measurements of different test cases captured at nearly the same time could both be affected by the same network disturbances, but the box test cannot take this into account. Finally, the box test is only a classifier, it cannot estimate the observed time difference between test cases, which is useful in some cases.

.Previous researchers have found that there are no "safe" parameters that work on all distributions, so one must employ a training algorithm to identify the parameters most appropriate to a given purpose.

We developed a box test training algorithm that is both relatively efficient and effective. We first observed that the distance between the low and high values (the "box width") is directly linked to the fraction of false positives to false negatives produced by the classifier. A larger box

Figure 6: Midsummary, w = 25

The quadsummary is just like the midsummary, except it has two additional order statistics calculated halfway between L1 and the median and R1 and the median. In other words, the quadsummary procedure is:

```
LET L1 = 50 - w

LET L2 = (L1 + 50) / 2

LET R1 = 50 + w

LET R2 = (R1 + 50) / 2

RETURN mean(P(L1), P(L2), P(R1), P(R2))
```

Finally, the septasummary is the same as quadsummary, except that it adds three more order statistics: the median, one halfway between L1 and the left tail of the distribution, and a third one halfway between R1 and the right tail. The pseudocode is:

```
LET L1 = 50 - w

LET L2 = (L1 + 50) / 2

LET L3 = L1 / 2

LET M = 50

LET R1 = 50 + w

LET R2 = (R1 + 50) / 2

LET R3 = (R1 + 100) / 2

RETURN mean(P(L1), P(L2), P(L3), P(M), P(R1), P(R2), P(R3))
```

Figure 5 for an example of the septasummary with the additional L3 and R3

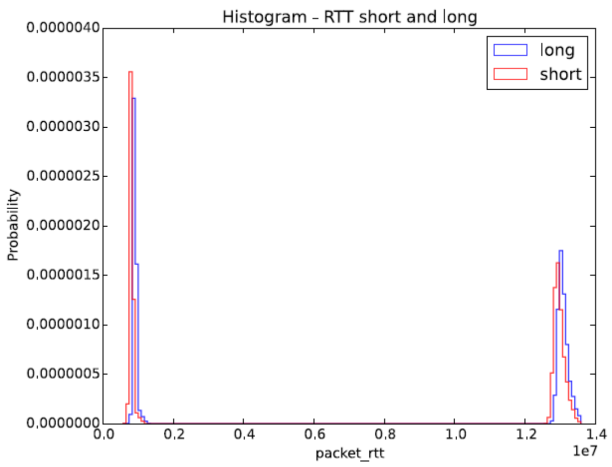
locations in green and the median in dark grey (at nearly the same location as the estimate).

Figure 5:Septasummary, w = 25

$$(p^i{}_k, p^j{}_k) \in \text{ arg}$$

$$t_i \in \arg\min_{t' \in T_i} \sum_{t \in T_i} \phi(t, t')$$

The reason all of the order statistics locations are parameterized from w is that it simplifies the implementation and training algorithm. Having a separate parameter for each pair of statistic locations could improve performance and may be investigated in the future.



If the L-estimator's result is greater than the threshold, then we classify the test cases as different. Otherwise they are classified as being similar.

Each of the L-estimators is trained using the same algorithm to determine the best w and threshold values, which is approximately:

1. The L-estimator is used across the entire training data to obtain a central tendency value. An initial threshold is then

set to 1/2 of this value (halfway between the expected L-estimator output and 0).

2. Next, every w value in the range [1; 49] is tested to identify the one with the lowest overall error.

3. Using the candidate w value from step 2, a series of potentially better threshold values are tested. The values tested are in the range of _20% of the threshold from step

1. A new candidate threshold value which best equalizes the false positive and false negative error rates is selected

4. Using the candidate threshold from step 3, a series of potentially better w values is tested in the range [w 4; w + 4] in increments of 1 percentile. The value which minimizes overall error is selected as the final w parameter.

5. In this final step, step 3 is repeated over the candidate threshold values _10% (in increments of 1%). Once again, the threshold chosen best equalizes the false positive and false negative rates.

III. PROPOSED METHODOLOGY

As part of this research we have developed the new tool suite "Nanown" (Morgan 2015) Nanown is designed around a multi-stage workflow

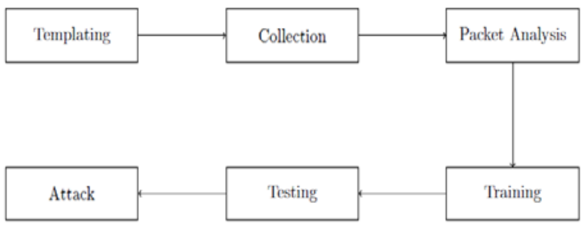


Figure 6: Nanown Workflow

Tinplating Collection Packet Analysis

Attack Testing Training

Classifier is trained using a dedicated subset of the samples collected using bootstrapping techniques. Each classifier is trained on a variety of different sample sizes and the total classification error (mean of false positives and false negatives) is minimized for each of these sample sizes. As a result of this stage, a series of candidate parameters for each classifier are saved.

During the testing stage, each candidate classifier and associated parameters are tested against a separate subset of the samples dedicated to testing. Each classifier is tested to see if it can reach a 5% or lower error rate given the quantity of data available. If this is possible, then the sizes are progressively reduced to isolate the smallest number of observations that still results in a 5% error rate. As an output of this stage, the user is informed of which classifier is most successful and how many samples will be required to reliably arrive at each classification.

At this point the user can easily determine how realistic an attack would be, as they know approximately how many requests would be required for each classification and the rate of sample collection is known from the second stage. If the user chooses to formulate an attack, they would write a script that leverages the classification method and parameters learned during the training and testing stages. For each timing difference classification needed, the attacker simply needs to collect the recommended number of samples and then apply theclassifier to the data. There is always some probability that a classification attempt may return an incorrect classification, so any key result the attacker needs to be sure of can be re-checked with additional round.

IV. EXPERIMENTAL RESULTS

Round Trip Time Measurement

Since TCP software stacks do not expose TCP timestamp information to callers, it was clear that the only way to obtain this information would be to either use a custom TCP implementation, or to employ a packet sniffer to monitor transactions between standard HTTP clients and services. The latter option was chosen as it seemed simpler and less error prone.

EstimatedRTT = 0.9 * PreviousEstimatedRTT + 0.1 * Sample RTT

Our current implementation spawns a long-running network sniffer to monitor all TCP connections between the local system and a designated target service. This sniffer records basic metadata about each packet observed, including port information, observed time stamp, raw TCP timestamp (if available), TCP sequence and ACK numbers, and whether the packet was sent or received. This information is written to a log file for later processing.

TCP Timestamps

Measurement Difficulties

The TCP timestamp option is defined in RFC 1323 and is designed to support algorithms that improve performance. When a host supports this option, every packet is labeled with the timestamp option which contains (amongst other items) a "TSval" field. TSval contains the current time of the sending host. For this reason, it is attractive to try and use this timestamp to estimate the run-time of a server-side operation, since this value would not be affected by most network jitter. However, in practice there are a number of problems with using TCP timestamps.

First and foremost is the fact that the majority of operating systems use a TCP times- tamp clock resolution in the range of 1ms to 5ms, while many of the operations an attacker would want to measure have time differences that are hundreds or thousands of times smaller than that. In addition, the clocks used for TCP timestamps by operating system kernels typically rely on real-time clocks that are not corrected for any hardware clock skew, which means different systems with the exact same software would not have precisely the same

Monte Carlo Procedure

Monte Carlo methods were used to systematically compare the performance of the different classifiers discussed above. To that end, for each of the network scenarios, three data sets were collected for _ 2 f40; 200; 1000; 5000; 25000g, the artificially induced execution time between the short and long messages. For each _, training (50,000 pairs), control (100,000 pairs), and test data sets (100,000 pairs) were collected. Overall, up to 1.25 million pairs of messages were collected for each network scenario.

$$\oint_{\partial\Omega} \hat{n}(\varepsilon\nabla\varphi) = \frac{\varphi_{i+1,j}-\varphi_{i,j}}{h^x_i}\left(\frac{h^y_j}{2}\epsilon^x_{i,j}+\frac{h^y_{j-1}}{2}\varepsilon^x_{i,j-1}\right)$$
$$-\frac{\varphi_{i,j}-\varphi_{i-1,j}}{h^x_{i-1}}\left(\frac{h^y_j}{2}\epsilon^x_{i-1,j}+\frac{h^y_{j-1}}{2}\varepsilon^x_{i-1,j-1}\right)$$
$$+\frac{\varphi_{i,j+1}-\varphi_{i,j}}{h^y_j}\left(\frac{h^x_i}{2}\varepsilon^y_{i,j}+\frac{h^x_{i-1}}{2}\varepsilon^y_{i-1,j}\right)$$
$$-\frac{\varphi_{i,j}-\varphi_{i,j-1}}{h^y_{j-1}}\left(\frac{h^x_i}{2}\varepsilon^y_{i,j-1}+\frac{h^x_{i-1}}{2}\varepsilon^y_{i-1,j-1}\right)$$

Given these data sets, the Monte Carlo procedure consisted of taking the following steps.

1. Train the different classifiers on subsets of the training and control data sets. That is, take a random subset (of different lengths) of the training and control data sets, then calculate the error rates for each of the methods across a range of parameter values. Repeat this many times, saving the error rates for each iteration and classifierparameter combination.

2. From the training data, identify the best (or a set of best) performing classifierparameter combinations.

3. Using the set of best performing classifiers, select a random subset of the test data and calculate error rates. Repeat this many times and save the results.

4. Compare the performance of the classifiers.

V. RESULTS

The overall results of the Monte Carlo experiment comparing the diverse Classifiers discussed in earlier sections. Each cell hearsay the results for a exacting association scenario and delta value. In the cell, when the classifier was able to achieve a < 5% total error rate in less than 20,000 interpretation, the number of explanation used to achieve this is reported. When an error rate below 5%was not achieved using a scrupulous classifier, the lowest error rate is reported (as a %). The best performing arts classifier is tinted in bold for each case. some things are clear from these results. First, no one of these classifiers works well with deltas below 1000ns. It is probable that for deltas this small, the number of observations desirable to achieve a sensible error rate is well above 20,000. The sum of noise in the network records is simply too huge virtual to the signal. Second, the box test doesn't perform all that well. In only one case (5000ns over the VM) did it have the best performance. In fact, in most cases (13 of 17 cases), the box test performed worse than all other classifiers. Third, the best performing classifier was the mid summary, which either required the smallest number of observations to attain an error rate below 5% or, when that level of error was not attainable, had the buck overall in 10 of 17 the cases. In compare the act of the three L-estimators, it seems that when the classification task is easier, due to a high signal to noise ratio, then the estimators with more order statistics are the most efficient. However, as the classification task becomes difficult, the midsummary tends to edge out the other estimators.

VI. CONCLUSION

This paper has looked at the issues associated with identifying timing side-channel vulnerabilities in web applications. The chief impediment to determining whether a timing attack is viable is overcoming the low signal to noise ratio that exists over network connections and under variable host conditions. To address this problem, we have developed and tested novel round-trip time measurement methods which include packet sniffing and a paired request sampling method. We went on to provide a thorough test of several different classifiers, which target the difference in execution times between request types. These tests show that the current state of the art in detecting execution time differences—the box text—does not perform particularly well when compared to other methods, namely the midsummary estimator, which performs significantly better. Finally, we have developed a tool, Nanown, that incorporates our methodology, making it accessible to other researchers and penetration testers.

VII. FUTURE WORK

We aim to widen this effort in a diversity of conduct. that follows is a short list of the improvements and extensions we are now bearing in mind. We guess the time profile to be greatly reliant on the exact course linking the server and client. However, we judge that packet sizes are not sturdily reliant on that pathway, as the Internet has a moderately usual MTU size of 1500 bytes. Thus, it may be achievable to coach a size profile from one spot on the internet, and test touching that profile elsewhere. additional experiments are needed to confirm this speculate.

REFERENCES

[1] Saman Feghhi and Douglas J. Leith" A Web Traffic Analysis Attack Using Only Timing Information" ieee transactions on information forensics and security, vol. 11, no. 8, august 2016 G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted HTTP streams," in Privacy Enhancing Technologies (Lecture Notes in Computer Science), vol. 3856, G. Danezis and D. Martin, Eds. Berlin, Germany: Springer, 2006, pp. 1–11.

[2] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a Distance: Website fingerprinting attacks and defenses," in Proc. ACM Conf. Comput. Commun. Secur. (CCS), New York, NY, USA, 2012, pp. 605–616.

[3] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in Proc. IEEE Symp. Secur. Privacy (SP), May 2012, pp. 332–346.

[4] S. Feghhi. (2015). Timing Only Traffic Analysis Project: Codes and Measurements

[5] [6] X. Gong, N. Borisov, N. Kiyavash, and N. Schear, "Fingerprinting websites using remote traffic analysis," in Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS), New York, NY, USA, 2010,pp. 684–686.

[6] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial Naïve–Bayes classifier," in Proc. ACM Workshop Cloud Comput. Secur. (CCSW), New York, NY, USA, 2009, pp. 31–42.

[7] A. Hintz, "Fingerprinting websites using traffic analysis," in Privacy Enhancing Technologies (Lecture Notes in Computer Science), vol. 2482, R. Dingledine and P. Syverson, Eds. Berlin, Germany: Springer, 2003, pp. 171–178.

[8] M. Jaber, R. G. Cascella, and C. Barakat, "Can we trust the inter-packet time for traffic classification?" in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2011, pp. 1–5.

[9] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in Proc. SIAM Int. Conf. Data Mining, 2001, pp. 1–11.

[10] M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS), New York, NY, USA, 2006, pp. 255–263.

[11] L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in Computer Security (Lecture Notes in Computer Science), vol. 6345, D. Gritzalis, B. Preneel,

[12] and M. Theoharidou, Eds. Heidelberg, Germany: Springer, 2010, pp. 199–214.

[13] X. Luo et al., "HTTPOS: Sealing information leaks with browserside obfuscation of encrypted flows," in Proc. Netw. Distrib. Syst. Symp. (NDSS), 2011, pp. 1–20.

[14] B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar, "I know why you went to the clinic: Risks and realization of HTTPS traffic analysis," in Privacy Enhancing Technologies (Lecture Notes in Computer Science), vol. 8555, E. De Cristofaro and S. J. Murdoch, Eds. Springer, 2014, pp. 143–163.

[15] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in Proc. 10th Annu. ACM Workshop Privacy Electron. Soc. (WPES), New York, NY, USA, 2011, pp. 103–114.

[16] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical dentification of encrypted Web browsing traffic," in Proc. IEEE Symp. Secur. Privacy, 2002, pp. 19–30 articleDetails.jsp?arnumber=1004359& newsearch=true&queryText=Statistical%20identification%20of% 20encrypted %20Web%20browsing%20traffic

[17] T.Wang, X. Cai, R. Nithyanand, R. Johnson, and T. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in Proc.23rd USENIX Secur. Symp. (USENIX Security), San Diego, CA, USA, Aug. 2014, pp. 143–157.

[18] X.Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in Proc. IEEE Symp. Secur. Privacy (SP), May 2007, pp. 116–130.

[19] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis," in Proc. IEEE16th Netw. Distrib. Secur. Symp., Feb. 2010, pp. 237–250 traffic-morphing-efficientdefense-against-statistical-tr Afficanalysis.