Preventing SSL Traffic Analysis with Realistic Cover Traffic (extended abstract)

Nabil Schear* and Nikita Borisov†
*Department of Computer Science †Department of Electrical and Computer Engineering
University of Illinois at Urbana–Champaign
{nschear2,nikita}@illinois.edu

1 Introduction

As more sensitive information is transmitted over computer networks, there has been a steady increase in the deployment of encryption to protect data in-flight. Myriad encrypted network protocols have emerged [8, 2, 1] that enable various applications like encrypted browsing, VPNs, secure shells, and VoIP. Since the data payload of an encrypted protocol is protected by strong encryption, attackers can do *traffic analysis attacks*, which use the information leaked by side channels (e.g., packet size and timing) to try to recover the contents or intent of the plaintext traffic.

Existing attacks can recover a wide range of information from encrypted communications, e.g., Web page visits [3, 6], typed passwords [7, 9], speech data [14, 13], and embedded protocols [15, 4]. Given this threat, a growing number of applications (e.g., low-latency anonymity systems and VPNs) and users (e.g., privacy advocates and whistle-blowers), need better protection from these attacks.

Existing techniques for preventing traffic analysis center on sending data with fixed intervals and/or with fixed payload sizes. Though this type of constant rate defense *is effective* at reducing the information leakage that enables most traffic analysis attacks, it makes it clear that a user is employing countermeasures to evade traffic analysis. This may, in itself, result in unwanted attention and scrutiny. We call this related attack *defense detection*.

To resist both traffic analysis and defense detection attacks, we propose using realistic cover traffic tunnels to mask the observable behavior of the real traffic to be transmitted. The user tunnels his or her real traffic via a proxy service, which embeds the traffic inside fake cover traffic. This tunneling approach incurs overhead in time and in the number of excess bytes transmitted. To trade off overhead and privacy, the user may vary his or her choice of cover traffic model.

In this paper, we introduce the basic design and evaluation of a cover traffic tunneling system called TrafficMimic. We utilize methods for generating realistic cover traffic, borrowing from prior work on traffic generation from the simulation and modeling research community [10]. Using several protocol classification and anomaly detection attacks, we show that TrafficMimic is able to reproduce cover traffic reliably and securely. Lastly, we show that realistic cover traffic provides comparable performance to constant-rate techniques, and in some cases can be very efficient.

2 TrafficMimic Design

To generate realistic traffic models to cover arbitrary encrypted tunnels, TrafficMimic uses two phases: i) learning traffic models, and ii) secure playback. In this section, we briefly summarize each in turn.

To build the learning phase of our secure traffic generator we use structural models. Structural models are a common technique used for simulation and performance testing for network traffic [12]. Structural models mimic the real interactions between the layers that generate network traffic (e.g., network, protocol, application, user). We choose to use Swing, a recent network traffic generator [10]. Swing uses a structural model that includes user, session, connection, packet, and network models to generate realistic traffic for network emulations. It does so by analyzing dumped pcap trace files and extracting empirical cumulative distribution functions (CDFs) of various structural features.

TrafficMimic itself performs secure playback of cover traffic and incorporates a tunnel of real data inside the cover traffic. TrafficMimic accepts connections from other applications and forwards all data sent and received over the input port across an encrypted tunnel using cover traffic. To use TrafficMimic, the user must have TrafficMimic nodes at both end points of the communication.

The remote TrafficMimic node removes the added control and padding information and delivers the decrypted data to the destination; likewise, data from the destination is encrypted and padded at the remote node and decrypted by the local TrafficMimic node.

Both endpoints of a TrafficMimic tunnel share the same design. We refer to the node accepting connections to forward as the *tm-client* and the end point connecting to destination hosts as the *tm-server*. To ensure synchronized cover traffic generation, one node controls the generation of cover traffic for both nodes. We call this control node the *master* and the other node the *slave*. The master may be at the tm-server or the tm-client. Since only the master controls cover traffic generation, the bidirectional cover traffic model can incorporate synchronized actions and predefined sub-sequences that would not be possible with two independent cover traffic generators at either end of the tunnel.

All cover traffic generation is performed in terms of a *traffic request*. A traffic request is a *specification* for what cover traffic TrafficMimic a node will generate. A traffic request contains three elements: a type, a size, and a time. The type dictates whether the request is to send data, send padding, or open/close/reconnect a connection.

Each TrafficMimic node contains an asynchronous thread which handles the generation and processing of cover traffic. To create cover traffic, the master model thread creates *model request*s using the appropriate Swing features, which contain a traffic request to be sent to the slave. Each model request also contains a *model response*, which contains a variable number of responses that the slave should send back. The master model thread sends the model request to the main network event-handling loop of TrafficMimic. TrafficMimic creates an appropriately sized output buffer to send to the slave according to the model request. TrafficMimic then embeds the traffic requests that make up the model response into the request buffer. It then places any real user data to be sent across the tunnel into the buffer. If additional space still remains, TrafficMimic pads it to the specified length. This buffer is then sent to the slave over an SSL encrypted channel. The attacker is unable to see which parts of the transmitted data are padding, control messages, or real user data. The slave uses the same process for merging control traffic, padding, and real user data as the master to create a bidirectional channel.

3 Evaluation

We evaluate TrafficMimic in two aspects: its security properties and the performance of using it for tunneling real traffic. We use the CAIDA 2009 passive data set [11] to train our supervised-learning protocol identification attack. Specifically, we use a K-nearest neighbor classifier-based attack. We found that it was able to detect protocols with 80-95% accuracy in cross validation against other *real* network traffic. We also extend it to support a simple anomaly detection algorithm as follows. During training, we find the maximum distance of any training point in each class from the centroid of the class (e.g., m_k where k enumerates the classes in the training set). During the application phase, we consider any test example whose distance, d_i, to its nearest neighbor is greater than the maximum distance observed during training to be an anomaly. We use the threshold t to control the degree to which test examples are labeled unknown. Thus a test example is considered an unknown protocol if: $d_i > t * m_k$.

We use Swing to generate traffic parameters for HTTPS, SMTP, and SSH protocols from the Feb CAIDA 2009 trace. For comparison to these realistic protocols models, we use TrafficMimic to generate constant rate cover traffic. TrafficMimic sends a full size packet (1448 bytes) every 500ms. The slave TrafficMimic client echoes back the same size packet upon receiving a packet from the master. Ten such exchanges occur per connection before a new connection starts. This constant rate model is a trade-off between interactivity, bandwidth, and overhead.

We found that we were able to fool our classifier approximately 73% of the time for the three realistic protocols we generated. Given that the classifier's accuracy on real traffic is around 80% the slight disparity is unlikely to lead to reliable defense detection.

We also found that we were 77.1% accurate in detecting constant rate traffic as anomalous. We found that the distance threshold algorithm was not able to differentiate SMTP and constant-rate traffic as well as it was for HTTPS and SSH. To solve this problem, we introduce another simple unsupervised anomaly detector to be used for SMTP only. It uses the K-means algorithm to cluster the data. It then uses a threshold algorithm to determine which clusters have the lowest mean inter-cluster distance (i.e., the mean of the distance between all pairs of points assigned to the cluster). We found that this technique was 95%+ accurate at differentiating SMTP and constant-rate traffic.

Next we focus on understanding the performance and overhead of our approach when tunneling real traffic. We use Iperf [5] to generate a simple bulk transfer of a 100KB file and connect it to TrafficMimic to evaluate the effects of cover traffic tunneling. Figure 1 shows the bandwidth in kbps for the bulk transfer on a link between hosts in Illinois and California. It also shows the number of times more bytes that need to be transmitted than the 100KB

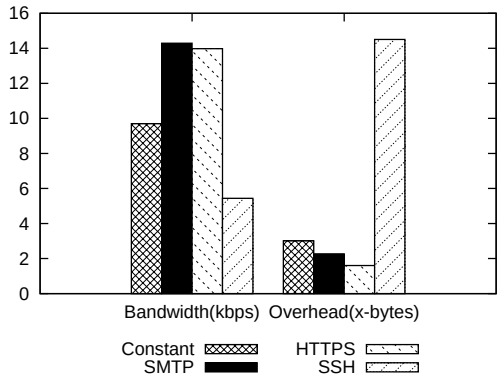

Figure 1: Performance of Generated Protocols Carrying Bulk Transfer Traffic

test file. This overhead factor includes *both directions* of the cover traffic even though the bulk transfer is only one-way.

The results show that we are able to achieve comparable or better bandwidth using realistic protocol models as compared to the constant rate model. The performance of HTTPS is especially good because the return path in the HTTP(S) protocol inherently favors bulk transfer of files. The SSH traffic model showed much higher number of excess bytes because of the asymmetry of SSH connections. Overall, realistic protocol models provide several options for performing bulk transfer efficiently while not succumbing to the security problems noted in the previous section.

4 Conclusion

We have introduced TrafficMimic; a traffic analysis resistance system that utilizes realistic cover traffic models to thwart defense detection attacks. We have shown that TrafficMimic has good performance compared to the existing traffic analysis defenses while also being more secure. We showed that the traffic models we use result in detection rates that are similar to those of real traffic and thus provide a good countermeasure for defense detection. We also evaluated the performance of TrafficMimic using a simple bulk-transfer and compared it with constant rate cover traffic. Overall, we found that TrafficMimic offered reasonable performance; in future work, we plan to investigate how to dynamically influence traffic generation to improve performance without sacrificing security.

References

[1] ATKINSON, R. Security Architecture for the Internet Protocol. RFC 1825 (Proposed Standard), Aug. 1995. Obsoleted by RFC 2401.

[2] BARRETT, D. J., SILVERMAN, R. E., AND BYRNES, R. G. *SSH, the Secure Shell: The Definitive Guide*. O'Reilly Media, Inc., 2005.

[3] BISSIAS, G. D., LIBERATORE, M., JENSEN, D., AND LEVINE, B. N. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Privacy Enhancing Technologies* (2005), pp. 1–11.

[4] DHAMANKAR, R., AND KING, R. PISA: Protocol Identification via Statistical Analysis. Blackhat US, 2007.

[5] GATES, M., AND WARSHAVSKY, A. Iperf. `http://iperf.sourceforge.net`.

[6] LIBERATORE, M., AND LEVINE, B. N. Inferring the Source of Encrypted HTTP Connections. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security* (New York, NY, USA, 2006), ACM, pp. 255–263.

[7] MONROSE, F., AND RUBIN, A. Authentication via Keystroke Dynamics. *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security* (1997), 48–56.

[8] RESCORLA, E. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[9] SONG, D. X., WAGNER, D., AND TIAN, X. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium* (Berkeley, CA, USA, 2001), USENIX Association, pp. 25–25.

[10] VISHWANATH, K. V., AND VAHDAT, A. Realistic and Responsive Network Traffic Generation. *SIGCOMM Comput. Commun. Rev. 36*, 4 (2006), 111–122.

[11] WALSWORTH, C., ABEN, E., KC CLAFFY, AND ANDERSEN, D. The CAIDA Anonymized 2009 Internet Traces. `http://www.caida.org/data/passive/passive_2009_dataset.xml`. Collected Jan 15, and Feb 19, 2009 at equinix-chicago monitor.

[12] WILLINGER, W., PAXSON, V., AND TAQQU, M. S. Self-Similarity and Heavy Tails: Structural Modeling of Network Traffic. In *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*. Birkhauser Boston Inc., Cambridge, MA, USA, 1998, pp. 27–53.

[13] WRIGHT, C. V., BALLARD, L., COULL, S. E., MONROSE, F., AND MASSON, G. M. Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations. In *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 35–49.

[14] WRIGHT, C. V., BALLARD, L., MONROSE, F., AND MASSON, G. M. Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob? In *Proceedings of the 16th Usenix Security Symposium* (2007).

[15] WRIGHT, C. V., MONROSE, F., AND MASSON, G. M. On Inferring Application Protocol Behaviors in Encrypted Network Traffic. *Journal of Machine Learning Research 6* (2006), 2745–2769.