CSE508 Network Security
Provably-Secure Traffic Analysis Defense

Feifei Ji (108232745),
ChenYang Liu (108036433),
Chien-Chun Ni (108233498)

May 16, 2012

1 Introduction

With the massive growth of the Internet, the importance of website browsing privacy also raises. While users serving on Internet, especially, via wireless connection, it is not hard for a malicious attacker to eavesdrop transmitting packets, and analysis statics such as packet upload/ download ratio, corresponding packet length and so on. These informations are harmful for *traffic analysis* attack. By digging these information, the attacker can reveal the websites that user are serving.

The most intuitive method to defense traffic analysis is to hide or alter the original packet length by padding. In [1], Liberatore et al. demonstrated by padding packet to network MTU, the accuracy of the traffic analysis can be reduced for 90%. However, in [2], Dyer et al. proposed that hiding packet lengths is not sufficient for traffic analysis since coarse information is unlikely to be hidden efficiency. By their VNG++ classifier with providing overall time, total bandwidth, and size of bursts, they can still make efficient traffic analysis.

In this project, we modified and fix the problem of "BUFLO" to defense against the attack. The rest of paper is organized as follows. In Section 2, we presented our design logic and implementing detail of the given algorithm. In Section 3, we evaluated our result with 200 different websites visiting. In Section 4, the conclusion and future work is proposed.

2 Design

To defense website fingerprint analysis, the most intuitive way is to hide or alter the output traffic load. For attacker, the fingerprint of a website can be analyzed from three kinds of data send between server and client: total transmission time, individual packet length, and total packet length. These coarse features served as the basis of the classifier. Our design goal was to obfuscate information as much as possible and control overhead.

2.1 Total transmission time

Total transmission time for both upload and download session is the most coarse feature for traffic analysis. It has been widely used as a fundamental feature in various classifiers. Even Dyer et al. suggested that with padding, the traffic can still be analyzed, classifiers based on solely total time still achieved good accuracy against random guess. Therefore, it is a good place to get started.

2.1.1 Design

During each session, we controlled the transmission time by sleeping randomly. The sleeping time was dynamically tuned for every fixed interval according to the current average transmission speed.

To hide information from both directions. we set up two proxies both on server side and client side. Both of the proxies measured the current throughput, and updated the sleeping time T as: $T = 2 * (throughput/T_{interval})$. In this way, at different time, the corresponding sleeping time T was randomized. This also reduced the overhead since the sleeping time is related to throughout.

2.1.2 Implementation

OpenSSH provided implementation of both the server and client proxy. For random sleep, we intercepted the main event loop function of running SSH proxy, and forced the loop to sleep randomly for every time it entered. The overhead was controlled by the method described in the Design section.

Additionally, we observed that base on the behavior of user serving the Internet, server usually sent much more packet to client. To adapt this

phenomena, we set the updating interval of the sleeping time T to be longer than the client loop. Also, to reduce performance penalty, we limited the maximum sleeping time of the server.

2.2 Individual packet length

Although Dyer et al. showed that hiding individual packet length failed to lower the accuracy of classifiers, we still decided to hide this feature to some extent. The reason was that the analyzer may still get the chance to intercept a sample user session of a interested website. For example, the censor may find some patterns of people's behavior when they log on Facebook, and intercept the packets of a time period to analyze.

2.2.1 Design

Individual packet length was padded to avoid leaking information. Each packet length was increased to the MTU. According to [2], a new random padding length r was suffice to achieve the same performance with lower overhead and we also experimented that method.

2.2.2 Implementation

We achieved per-packet padding from the packet level function of OpenSSH. To do this, we modify the the fundamental SSH packet assembly function, and force the sending packet to be padding to MTU.

2.3 Total packet length

This last and the most important feature we tried to obfuscate is the affections of per-direction bandwidth and burst rate. As shown by Dyer et al, this feature greatly boosted the classifier and revealed good amount of information of the website.

2.3.1 Design

Total packet length in one session also needed padding because it also provided useful features to the classifier. We kept tracking the total packet length in a session and padding it to the nearest power of two after we detected the session was idle. We defined the idle session as no packet coming

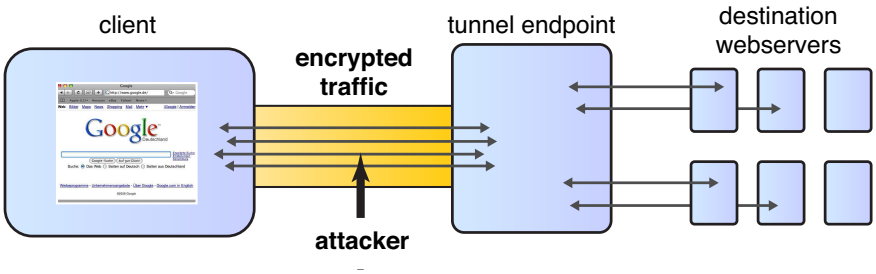

Figure 1: Website fingerprinting scenario and conceivable attackers

in 1.5s (i.e. the minimum time we must send packets). After this idle is detected, we continued transmission and sent junk packets until the total length reached power of 2.

2.3.2 Implementation

We maintained a idle time counter to reach our goal. The idle time was accumulated as long as the output bytes remained the same in each loop. For every time the idle time exceeded the threshold, extra packets must be send. We started to put junk packets into the send queue to sent them. This function was implemented in both sides.

3 Evaluation

In this section, we describe the experimental methodology to evaluate our implementation.

Fig. 1 illustrates the network setup we use throughout this paper. In it, the client connects to a remote proxy over an encrypted transport layer. To simplify the experiment, we use *localhost* as proxy server and port 1080, that is 127.0.0.1:1080. The proxy makes requests on the client's behalf, and returns the results over the encrypted connection. The observer is limited to examining the encrypted traffic and creates logs of packet lengths (and inter-arrival times, if desired).

The proxy we evaluate in this paper is the OpenSSH implementation of a one-hop SOCKS proxy. However, we expect our results hold for VPN proxies and WPA base stations. Neither of these systems significantly alter

Padding method	Overhead(%)	Attacker Information Gain(%)
Linear	39	41.6
Exponential	15	43.9
Mice-Elephant	51	60.8
Pad to MTU	38	40.3

Table 1: overhead and entropy

packet lengths since they perform no buffering and packet aggregation or fragmentation [3].

3.1 Individual Packet

We compared 4 different padding methods [2] and tested their overheads and ability to hide information.

To measure the ability to hide information, we calculated the entropy of the packets length and calculate the information gain attacker can achieve most.

Mice-Elephants padding if packet length is $\leq$ 128, then the packet is increased to 128 bytes; otherwise it is padded to the MTU.

Linear padding All packet lengths are increased to the nearest multiple of 128, or the MTU, whichever is smaller.

Pad to MTU All packet lengths are increased to MTU.

Exponential padding All packet lengths are increased to the nearest power of two, or the MTU, which ever is smaller.

The result is shown in table 1. In the table, exponential padding gets the least overhead. padding to MTU allows attacker least information gain. Linear padding and Exponential padding get almost the same result as padding MTU. Almost 40% information gain shows that padding packets is not very helpful for secure traffic analysis defense.

3.2 Session

Since the total number of bytes transmitted is always a power of two, we can represent it by its log, k_{up} and k_{down}. Likewise, the timing of a session are

also a power of two and can be represented by their log, t_{up} and t_{down}. So the attacker observes 4 values: (k_{up}, k_{down}, t_{up}, t_{down}).

We visited a collection of 200 websites obtained via Yahoo's random URL service and recorded the (k_{up}, k_{down}, t_{up}, t_{down}) value of each session. Then we calculate the residual Shannon entropy of the random variable representing the 4 values of a session. The definition of entropy are as follows.

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

$$R(X) = 1 - H(X)/\log n$$

Here H(X) represents the entropy of random variable X, and $P(X = x_i) = p(x_i)$, R(X) represents the information you can gain comparing to uniform distribution (the entropy of a uniform random variable is $\log n$, where n is the number of all possible tuple). The result of $R(X)$ in our experiment is 0.01617, which means the records of the 4 values are almost uniform distributed. Thus the attacker can get very little information gain.

4 Conclusions

We shown in this report that efficient countermeasure against traffic analysis exists without introducing much overhead. We explored coarse features of SSH-encrypted packets between client and server proxy. The packet profile generated by our scheme smoothed out certain outstanding feature. We also measured and evaluated the overhead of this scheme.

Future Work The ultimate and most convincing evaluation of the security of our scheme should be conducted on the recent classifier proposed by Dyer et al. We will in the future analyse the packet using external tool like WireShark and extracted features to feed the classifier. It will provide deeper insight to our defense.

References

[1] M. Liberatore, "Inferring the source of encrypted HTTP connections," in *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, 2006.

[2] K. Dyer, S. Coull, and T. Ristenpart, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," *kpdyer.com*, 2012.

[3] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *CCSW '09: Proceedings of the 2009 ACM workshop on Cloud computing security.* ACM Request Permissions, Nov. 2009.