# Quantifying Traffic Analysis of Encrypted Web-Browsing CS261, Fall 1998

Shailen Mistry, Bhaskaran Raman EECS Department, U.C.Berkeley

December 10, 1998

#### Abstract

With web transactions becoming increasingly common, the need was felt for making them secure and authenticated. The Secure Socket Layer (SSL) protocol was designed to provide security and integrity of data over an untrusted network and is used mainly for securing HTTP data. However, SSL does not make any effort to prevent traffic analysis. Hence, it is possible to learn interesting information about what a user is doing even if all transactions are done over SSL (opening bank accounts, money transfers, online purchases).

Our aim in this project was to quantify the extent of traffic analysis possible. We tried to see if we could figure out the URLs visited by a user by just looking at the sizes of the packets being exchanged. We found that for many web sites, the hacker could quite accurately track which pages a user was visiting within the site.

## 1 Introduction

With the arrival of the world-wide-web, electronic commerce started to become increasingly popular. However, the internet consists of users from different trust domains as well as hackers. There is a plethora of ways to act maliciously as an internet host [Bel95], [Bel89], [FV]. Hence, SSL [Net] was designed to secure web transactions [Res98].

SSL provides a mechanism for security and integrity of data being transferred over the internet. But it does nothing to prevent traffic analysis of the encrypted data. In many cases, sensitive information can be learned without even looking at the contents of the packets on the wire. Some examples of the kinds of information that can be found are:

- The transactions that a user is performing at an online banking site: money transfer, ordering traveler's checks, opening a new account.
- The books that someone is buying at an online bookstore.

• Whether a particular person is buying or selling stocks in the share market.

In this project, we tried to see if such information can actually be learned by doing traffic analysis. We quantified the accuracy to which a hacker can learn such information just by looking at the sizes of the transfers. We made the following minimal assumptions.

- The hacker can sniff at the packets being transferred. Anyone with root access on an internet host on the path in between the client and the server can do this easily with tcpdump [McC].
- The hacker can browse through the secure site beforehand. All this requires is an account at the web-site.

We tried our experiments with four different secure sites, two using SSLv2 and the other two using SSLv3. Our results show that the hacker could very easily, and quite accurately track the pages the user was visiting. Even in cases where the pages were dynamically generated at the server (using a CGI script), we found that the size of transfer could fairly accurately tell that the particular page was visited.

The single largest shortcoming of our testing was that we used only one user account at each of the online sites.

#### SSL Background

For the sake of the reader not familiar with the specifics of SSL, we briefly mention the aspects of SSL relevant to the discussion in the subsequent sections.

There are two versions of SSL commonly used: SSLv2 and SSLv3. While SSLv3 is more secure, most online sites still use SSLv2. SSLv3 has been standardized as an IETF draft in [DA98]. Two out of the four secure sites in our tests used SSLv2.

Although the two versions are quite different, for our purposes, they can treated similarly. Both have 2 layers of operation: the handshake layer and the record layer. The handshake layer is responsible for the initial authentication of endpoints and for establishing keys to be used for securing and authenticating communication after the handshake. The record layer provides the framing and packet structure required for encrypting the data of the layer above. The SSL protocol stack is given in figure 1.

## 2 Methodology

Our basic approach was to compare the size of information sent between the server and client against information gathered apriori. The hacker sniffs at the packets exchanged between the two ends. The setup we had is shown in figure 2. The steps we followed are as given below.

- 1. For a particular site, we initially went through the pages of the site recording two types of information about each page.
  - SSL transfer size: We recorded the size of the packets just after stripping off the TCP/IP headers. Hence this size included the SSL header, HTTP header and the HTML/Image download.
  - Locality information: We included HTTP link information between pages. For example, the "confirm-traveler's-check-order" page on the banking site could be reached only from the "order-traveler's-check" page.

We gathered this information for static and dynamic pages, as well as for embedded images.

- 2. As the client browsed through the pages at the particular site, we recorded the size of SSL download again by looking at the TCP sequence numbers. To do this,
  - We first separated out the packet exchanges for each TCP connection using the client port number information the connections are independent of one another.
  - We then recognized and ignored the SSL handshake at the beginning of each TCP connection.
  - On each connection, after the handshake, we treated further packets from the client as a new HTTP request or an SSL session close.
  - Finally, we recorded the aggregate size of the packets returned in response to each client request.

In particular, we did not look at the contents of any of the packets - not even the initial unencrypted SSL handshake packets. This is because we found that for a particular site, the handshake could easily be recognized.

3. Finally, we did a match of the download sizes for the client against the information built earlier. We used locality information where possible. In other cases, we just did a context-insensitive match on size.

We used a threshold of  $\pm 40$  bytes for matching.

#### Choice of web-sites

We chose the following SSL-enabled web-sites for our tests.

• https://www.isaac.cs.berkeley.edu: the homepage of the *Internet Security*, *Applications*, *Authentication and Cryptography (ISAAC)* research group at Berkeley. This was our simplest web-site. Most pages were static and there were very few embedded images. Also, this site did not have any user-accounts or logins.

- https://www.index.berkeley.edu: the homepage of the *Internet Demand Experiment (INDEX)* project at Berkeley. This site had a large number of embedded images, dynamically generated web-pages, as well as a userlogin. It also had more web-pages than the ISAAC site.
- https://banking.wellsfargo.com: Wells Fargo's online banking site. This was a realistic test case containing potentially sensitive information. This site has mostly dynamically generated web-pages since it deals with user-specific banking transactions. As with INDEX, it had a user-login for the account holder.
- https://orders.datek.com: Datek's online stock brokerage site. Like the Wells Fargo site, this also had a user-login and many dynamically generated web-pages for doing online trading.

While ISAAC and Wells Fargo use SSLv2, INDEX and Datek use SSLv3.

We tried to include Amazon's web-site http://www.amazom.com in our tests. Since it was unencrypted, we attempted to go through an SSL-proxy [Cha]. But since the site used http-redirects, the proxy could not be used.

#### Test cases

We performed tests under the following cases. The cases differ from one another based on whether or not the client had enabled auto-download of embedded images, and on whether or not the client cached static (non-dynamic) webpages.

- Case A: No images, no cache. We started with this rather unrealistic, but simple scenario.
- Case B: With images, no cache. We tried this case to see if the size-matching process became less accurate if there were a lot of images. We expected that this could happen if the sizes of the images turned out to be quite close to the sizes of the html-pages we were interested in.
- Case C: With images, with cache. This was a more realistic case. One can imagine that this case would be easier to handle than the previous case. This is because we expect that the images will be cached at the client and hence will not be downloaded more than once.
- Case D: With images, with warm cache. As another realistic case, we tried our tests with a warm client cache after the client had recently visited the site. In this case, we expected that most of the static pages, as well as images would be cached and the only pages transferred would be dynamic ones.

#### 3 Results

Figures 3, 4, 5 and 6 show the results of our tests for the four sites. For each site, we did three sets of tests which had a sequence of five, ten and fifteen distinct

web-pages. In some cases, we had more than this number of downloads. This was because of two reasons: (1) multiple frames in a single page (2) revisiting the same page involved another transfer in cases where there was no caching.

#### ISAAC Site

In the ISAAC test-site, the results indicate an exact match or an approximate match to all pages visited for case A and B (see figure 3). This illustrates that when no caching is involved, our analysis is quite accurate. In cases C and D, when caching was enabled, there was an exact or close match whenever there was a download. Case C exhibits that pages revisited during browsing (like the "nt-lucre" page in TEST 2) are recognized during their initial download, but are in the browser's memory for subsequent visits. Case D magnifies case C with almost all pages not being recognized as they are already in the browser's cache.

Since ISAAC was mostly a static site, traffic analysis identified all pages fairly accurately. We did not have to use any locality information for recognizing any of the pages on this site.

#### INDEX Site

For the INDEX site, tests indicate an exact match for majority of the pages (see figure 4). We did have a few pages, such as the "entry" page after login, that were mismatched consistently. This was because the download size for these pages varied significantly. We expected that we would have more inexact matches in case B (with images) than in case A (without images) due to the large number of images in the site. But we found that this pattern was not clearly shown in our results. This was because the varying size of the pages played a greater role in causing incorrect matches.

#### Wells Fargo Banking Site

Wells Fargo's online banking site proved to be a very interesting site to investigate. All the pages at this site were dynamically created. Although the pages in the site would be expected to vary greatly since it depends on the user, we discovered that most dynamically created pages were almost invariant in size. An exact match was the predominant case, while we did have a few mismatches (see figure 5).

We found that pages such as "accnt-summary" and the "checking" page could not be recognized at times since they varied greatly depending on the user's banking transactions.

We included locality information for some pages in this site. For example, the "fund-transfer-confirmation" page appears only after the "fund-transfer" page.

At this site, everything from a monetary transfer, to creating a new account, to foreign currency exchange, to changes in user information could be recognized by the hacker.

#### Datek Stock Brokerage Site

The Datek Online Stock Brokerage was another great site to explore. The only

mismatches at this site were for pages that varied greatly in size (see figure 6). For example, the "history" page showing the history of user activity, was of a different size each time (as can be expected). Since these pages could be accessed from anywhere on the site, locality information could not be used to detect them.

We included locality information for some pages in this site as well. For example, the "confirm-order" page could be reached only from the "get-quotes" page.

Interesting events that could be determined on the Datek site included: request for a particular number of quotes, trade confirmations, stocks or funds redemptions, and account information modification.

#### Some Interesting Observations

- The observation that simplified our work a great deal was that the SSL handshake pattern could be recognized without looking at their contents.
  - For SSLv2, the handshake consisted of three 35 byte packets sent from the server-to-client, client-to-server, and server-to-client.
  - For SSLv3, the number of bytes sent by the client in the handshake was always one of a few values.

Hence, in both cases, it was quite easy to recognize the handshake.

- Of particular interest was Wells Fargo's use of dynamically created pages for their whole site (after user login). The session arguments in the URL change during each transfer forcing a cgi script to be run which produces the page. Therefore, for this site, cases B, C and D could be considered the same.
- For sites other than Wells Fargo, in case D (warm cache), most pages were cached and thus not transferred.
- While we expected that the inclusion of images may result in more inexact matches, not all of our tests show this clear pattern. In some cases, other reasons (like varying page sizes) caused more inexact matches in the case without embedded images than in the case with images.
- Interestingly, we discovered that even with cache size set to zero, Netscape has a display cache to handle "back" and "forward" commands.

### 4 Conclusions

#### What we can do

The tests illustrate that in most cases traffic analysis can determine the web page browsed. Transfer sizes of web pages are predictable enough to allow the hacker to recognize the client's navigation through a web site.

Traffic analysis of SSL encrypted web browsing allows the attacker to find a great deal of important information. As observed in the four test-sites, events such as money-transfers, trades, creation of new accounts, and modifications to user information can easily be detected.

#### What we cannot do

Traffic analysis will not work in cases where the size of page varies greatly. It will be less effective when the page sizes are close to one another. These are of course, very site specific. However, for the sites that we tested, the pages were different enough to allow recognition based on size to succeed on most cases.

#### **Shortcomings**

Our tests were limited mainly by the use of one user account on the INDEX, Wells Fargo, and Datek web sites. Pages like the "history" page on Datek, "accnt-summary" page on Wells Fargo and the "usage-detail" page on INDEX were highly variable in size and highly user-activity dependent. But, other dynamic pages like the "confirm-order" page on Datek, "money-transfer" page on Wells Fargo or the "change-passwd" page on INDEX have little user-specific information and are almost constant in size.

## 5 Further Work

#### Adding More Functionality

- For the sites tested, the database was created by manually recording the transfer sizes. Further work would include writing a program to automatically dump the contents of an SSL encrypted site. But, this may not be possible for banking or stock brokerage sites. The main concern with such sites would be that a spider could run undesirable transactions (like selling stocks at a price of zero!).
- By integrating the sniffer and the analysis tools, the traffic analysis can be done in real time, while the client is browsing.

#### Further Traffic Analysis

- The results found on the accuracy of traffic analysis of web browsing over SSL are very promising. Yet, further testing with more client variability is important.
- We used page sizes and locality information to match pages. Yet, we do not know how much locality helps with the matching. In most of our tests, just the page size information was enough to find an accurate match.
- Since block ciphering is limited to 64 bits (8 bytes), it probably will not effect traffic analysis. We expect this because our matching threshold was 40 bytes. However, one way to weaken our traffic analysis would be to add large, random padding for each transfer at the SSL record layer.

• IP Security will make traffic analysis much more difficult as the port numbers of the transfers will be encrypted. Simultaneous multiple connections cannot be examined. In addition, we cannot separate out https packets from non-https packets. But if pages were transferred serially and there is little non-https packets, our traffic analysis will probably still work.

## References

- [Bel89] Steven M. Bellovin. Security problems in the tcp/ip protocol suite. Computer Communication Review, 19(2):32-48, April 1989.
- [Bel95] Steven M. Bellovin. Using the domain name system for system breakins. In *Proceedings of the Fifth Usenix UNIX Security Symposium*, June 1995.
- [Cha] Yatin Chawathe. Ssl forwarder. Personal Communications, http://www.cs.berkeley.edu/~yatin.
- [DA98] Tim Dierks and Christopher Allen. The tls protocol version 1.0. Transport Layer Security Working Group, INTERNET-DRAFT, http://www.ietf.org/internet-drafts/draft-ietf-tls-protocol-06.txt, Nov 1998.
- [FV] Dan Farmer and Wietse Venema. Improving the security of your site by breaking into it. http://www.deter.com:80/unix/papers/improve\_by\_breakin.html.
- [McC] Steven McCanne. tcpdump network monitoring tool. ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.
- [Net] Netscape. Secure sockets layer. http://www.netscape.com/products/security/ssl/index.html.
- [Res98] E. Rescorla. Http over tls, http://www.ietf.org/internet-drafts/draft-ietf-tls-https-01.txt. INTERNET-DRAFT, March 1998.