INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

ALLAHABAD

5TH SEMESTER MINI PROJECT

Side Channel Attack

Submitted To:
Dr. Bibhas Ghoshal

Submitted By :
Anurag Bhardwaj
Kautish Jaiswal
Nitesh Gupta
Ayan Sheikh
Yash Bhatia

CERTIFICATE FROM SUPERVISOR

I do hereby recommend that the mini project report prepared under my supervision, titled "Side Channel Attack" be accepted in the partial fulfillment of the requirements of the completion of mid-semester of fifth semester of Bachelor of Technology in Information Technology.

Date:
Place: Allahabad

Supervisor :
. . . .
Dr. Bibhas Ghoshal

Abstract

Information technology (IT) is evolving everyday and our day-to-day life is becoming more and more dependent on it. We cannot imagine ourselves without online banking systems, social networking sites, video streaming and World Wide Web. Due to this evolution of Internet our life has become more convenient and technologically advanced. During this course we are exploring the dark side and horrors of Internet regarding hacking and cyber-crimes. Apart from these direct attacks, side channel leaks can be used on search engines like Google, Bing, Amazon to retrieve clients' sensitive information. There is a wrong assumption that encryption restricts data access to only authorized users. Side Channel leaks are the direct results of implementation constraints giving access of clients' data to unauthorized users.

We explore the practicality of one such side-channel attack where the attacker deduces what was typed in the search suggest box (suggestions to partially typed query) by just observing the sizes of the packets exchanged between a user and an access point. In order to uniquely map a search query to a web traffic signature, we make use of Stochastic Algorithms.

Our goal is to find an optimal method to implement the side-channel attack using the packet lengths of data. We also present some methods to mitigate such side-channel attacks.

Contents

1 Introduction

The sensitive information exchanged between two entities is encrypted to ensure that they are not eavesdropped upon by unintended parties. Even with such security mechanisms in place, some information get leaked by the system unintentionally. This leaked information can be catastrophic enough for the system.

Even if the connection between the client and server is encrypted, three parameters of packet flow can be observed for HTTPS protocol:

1. Lengths of individual packets.
2. Directions of packet flow (client to server or server to client).
3. Times of packets' departure and arrival.

It is observed that for every character typed in the search box, several packets are exchanged between the server and the client with list of suggestions:

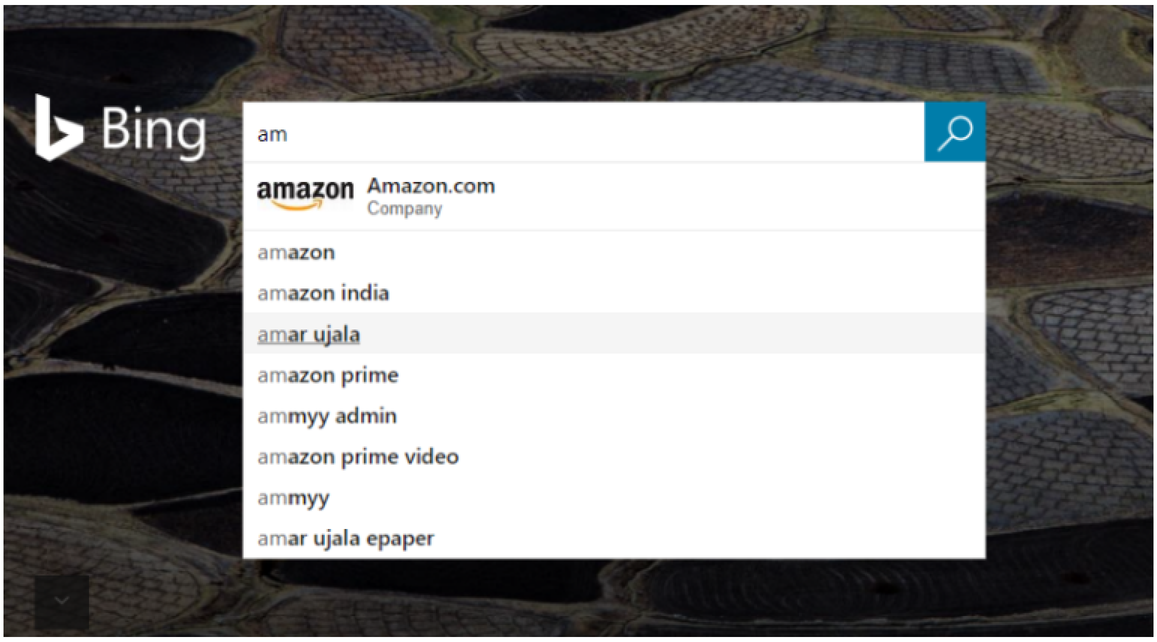


Figure 1: Search suggestions from Bing

The search box operates using AJAX to display suggestions. We can attack by intercepting the exchanged packets in order to infer the users' query.

Exchanged packets between a client and server can be observed using a packet sniffer such as *Wireshark*. Captured packets are read by *Pyshark*. We maintain the records of packet lengths observed when each word in the dictionary is typed as a search query. We then use *stochastic* algorithms to compare the observed packet lengths with the stored values to infer the query most likely typed in by the user.

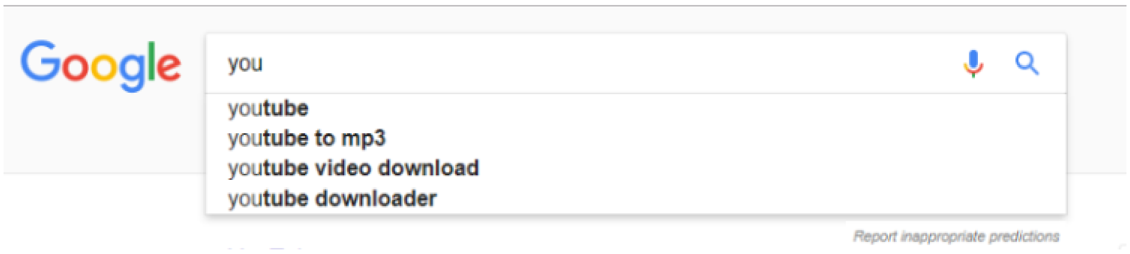


Figure 2: Search suggestions from Google

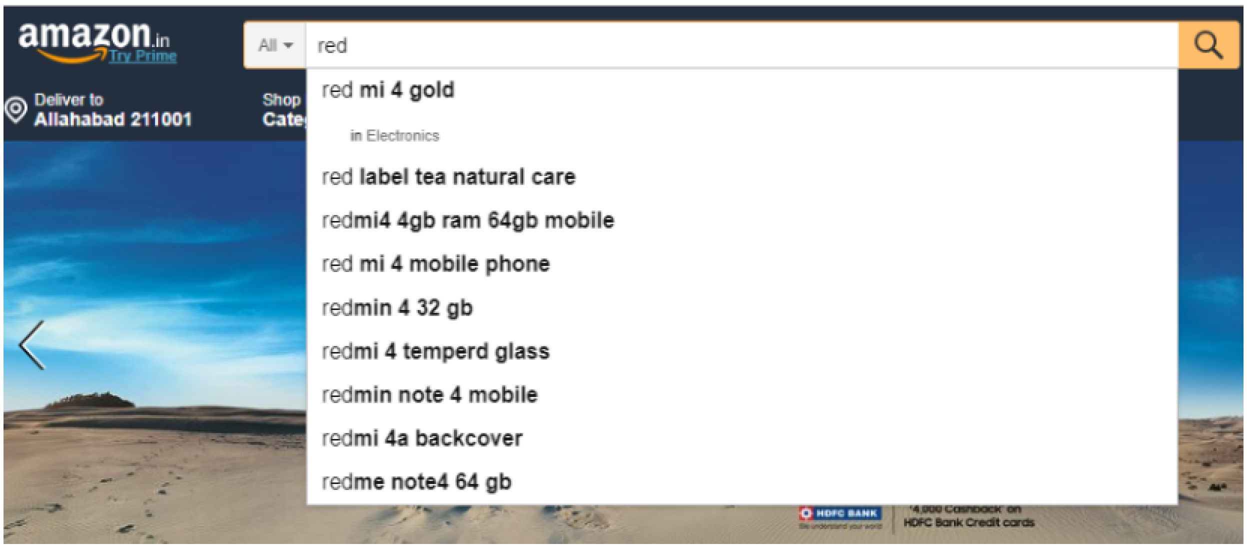


Figure 3: Search suggestions from Amazon

2 Motivation

Online security breaching is a criminal activity that can happen from any part of the world. Most breaches are done by falsifying the personal information or even a breach of security stemming from lack of security. Now that more and more crimes are happening over the internet there are many different aspects to security as a whole. Due to the drastic increase in modern cybercrime the cyber police, information technology specialists, and even the government need to work together to find a solution.

A data breach is a security incident in which sensitive, protected or confidential data is copied, transmitted, viewed, stolen or used by an individual unauthorized to do so[1].

Many times private organisations like Wikileaks have exposed secret military documents and unveiled government policies[2].

Until 2011, Google searches were not encrypted. The search giant enabled encryption for its signed in users[3] in stages and quickly followed it by a complete roll-out[4] of its SSL based version in late 2012.

There are many incidents of data breaches in the history of internet[5]. This motivated us in following ways:
1. Gaining access to a client's search history has huge privacy implications, and privacy is intimately related to computer security.
2. Search history is also of commercial importance, with companies using it for targeted user interest based advertisements.

Recent studies show that such side-channel leaks are both fundamental and realistic. A set of popular web applications are found to disclose highly sensitive user data such as one's family income, health profiles, investment secrets and much more through their side channels. Our study also shows that a significant improvement of the current web-application development practice is necessary to mitigate this threat.

3 Proposed Approach

From the above mentioned three packet parameters, packet length and direction can be used to guess the key stroke.

3.1 Reduction of possibilities

To find a string of size n the victim of our attack has to send 26^n automated search requests to the server. We observed that most of the search requests do not qualify to be a word that one might search. This process can be optimized by eliminating a lot of search requests and responses as they may not be legitimate English words and thus, restricting it to a dictionary, i.e. set of predefined words.

3.2 Data Structure

The packet intercepted from the search suggestions represent prefix of the word. Thus the best data structure suitable to deal with prefix string matching i.e. Trie tree is used. Once the dictionary is chosen, we construct a trie tree which stores all the meaningful words of a dictionary along with probability ranking based on various search engines query dataset. Each node has 256 children each representing an alphabet with the root node representing the *null* character. Likewise subsequently each level l represents valid words of length l sorted lexicographically. The below snippet describes the trie further.

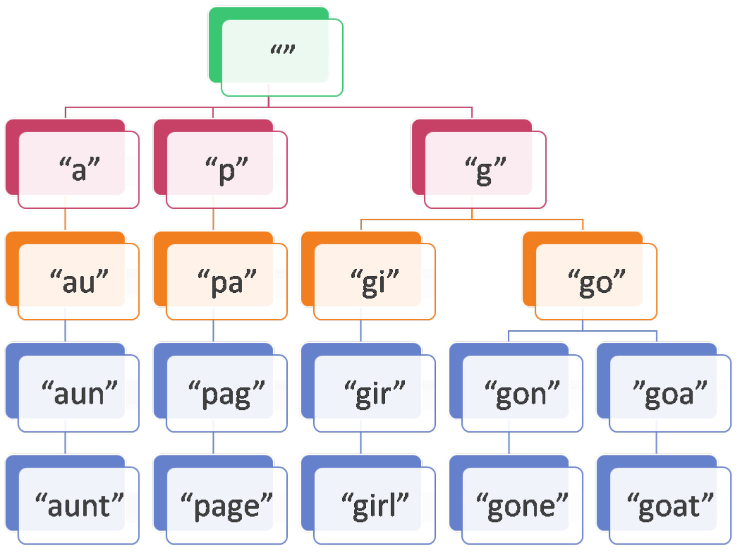


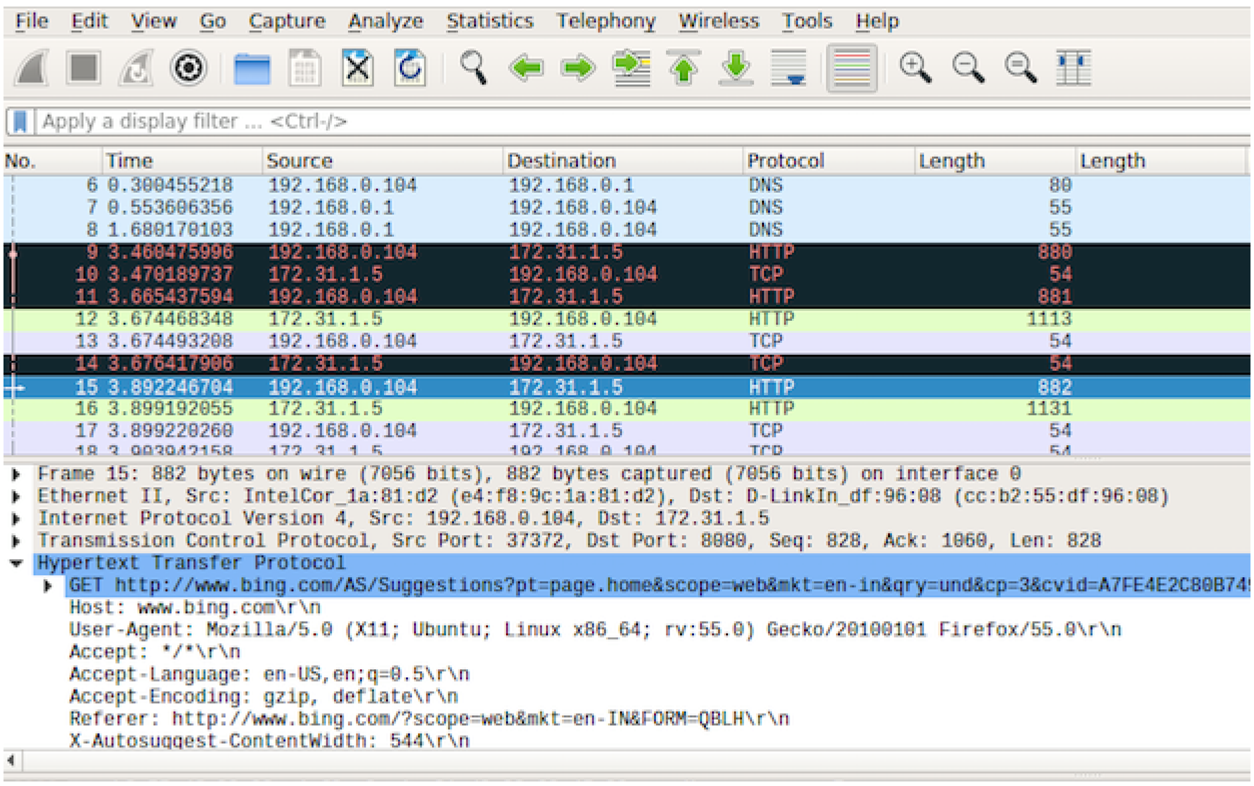
Figure 4: Structure of Trie

3.3 Packet Sniffing

a) **Through Wireshark:** Wireshark is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions[6].

Packets corresponding to search suggestions are captured using packet sniffers such as Wireshark using suitable filters.

Wireshark needs the interface to receive all the packets that it sees whether they are addressed to it or not. Hence, we run wireshark in promiscuous mode.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

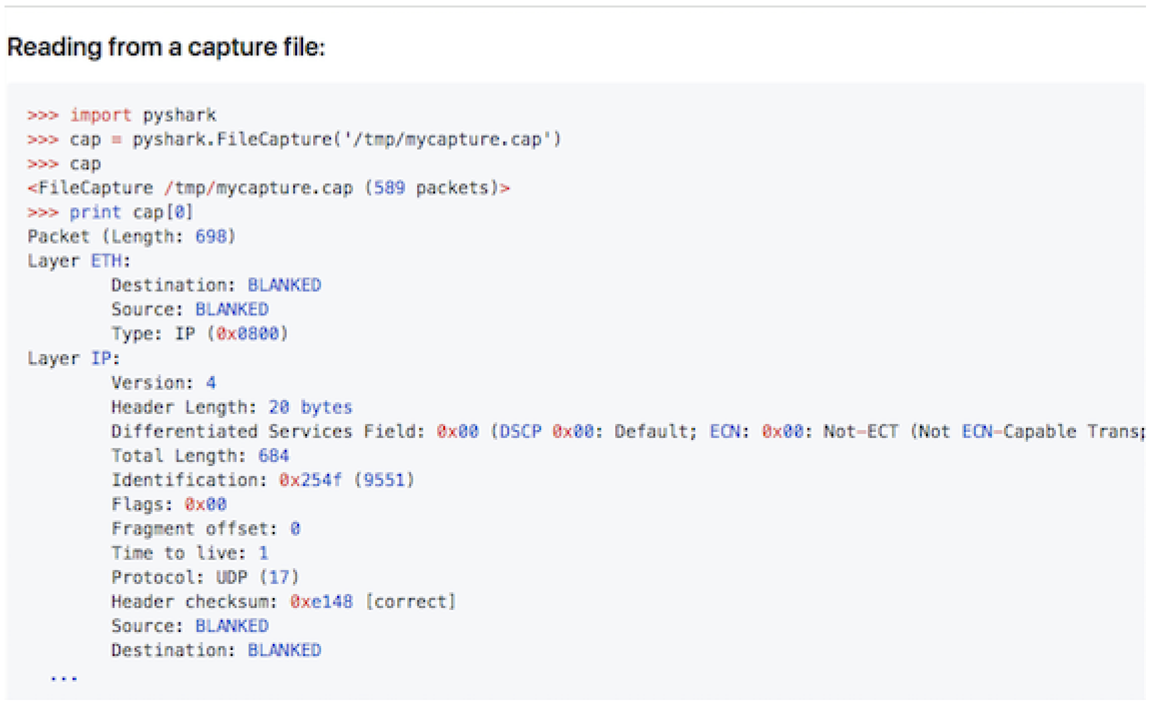
```
No.   Time          Source          Destination     Protocol  Length  Length
    6 0.300455218   192.168.0.104   192.168.0.1     DNS           80
    7 0.553606356   192.168.0.1     192.168.0.104   DNS           55
    8 1.680170103   192.168.0.1     192.168.0.104   DNS           55
    9 3.460475996   192.168.0.104   172.31.1.5      HTTP         880
   10 3.470189737   172.31.1.5      192.168.0.104   TCP           54
   11 3.665437594   192.168.0.104   172.31.1.5      HTTP         881
   12 3.674468348   172.31.1.5      192.168.0.104   HTTP        1113
   13 3.674493208   192.168.0.104   172.31.1.5      TCP           54
   14 3.676417906   172.31.1.5      192.168.0.104   TCP           54
   15 3.892246704   192.168.0.104   172.31.1.5      HTTP         882
   16 3.899192055   172.31.1.5      192.168.0.104   HTTP        1131
   17 3.899220260   192.168.0.104   172.31.1.5      TCP           54
   18 3.903942158   172.31.1.5      192.168.0.104   TCP           54
```

```
▸ Frame 15: 882 bytes on wire (7056 bits), 882 bytes captured (7056 bits) on interface 0
▸ Ethernet II, Src: IntelCor_1a:81:d2 (e4:f8:9c:1a:81:d2), Dst: D-LinkIn_df:96:08 (cc:b2:55:df:96:08)
▸ Internet Protocol Version 4, Src: 192.168.0.104, Dst: 172.31.1.5
▸ Transmission Control Protocol, Src Port: 37372, Dst Port: 8080, Seq: 828, Ack: 1060, Len: 828
▾ Hypertext Transfer Protocol
  ▸ GET http://www.bing.com/AS/Suggestions?pt=page.home&scope=web&mkt=en-in&qry=und&cp=3&cvid=A7FE4E2C80B74
    Host: www.bing.com\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:55.0) Gecko/20100101 Firefox/55.0\r\n
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Referer: http://www.bing.com/?scope=web&mkt=en-IN&FORM=QBLH\r\n
    X-Autosuggest-ContentWidth: 544\r\n
```

Figure 5: Capturing packet using Wireshark

The above snippet shows how wireshark is used to predict the length of the search suggestion packet.

b) **Through Pyshark:** Python wrapper for tshark, allowing python packet parsing using wireshark dissectors. We use pyshark library to capture live packets through a simple python code. Packet lengths are captured for various possible keystrokes and recorded in the dataset. The following snippet shows how to read packets using pyshark:



Reading from a capture file:

```
>>> import pyshark
>>> cap = pyshark.FileCapture('/tmp/mycapture.cap')
>>> cap
<FileCapture /tmp/mycapture.cap (589 packets)>
>>> print cap[0]
Packet (Length: 698)
Layer ETH:
        Destination: BLANKED
        Source: BLANKED
        Type: IP (0x0800)
Layer IP:
        Version: 4
        Header Length: 20 bytes
        Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transp
        Total Length: 684
        Identification: 0x254f (9551)
        Flags: 0x00
        Fragment offset: 0
        Time to live: 1
        Protocol: UDP (17)
        Header checksum: 0xe148 [correct]
        Source: BLANKED
        Destination: BLANKED
  ...
```

Figure 6: Reading packets from pcap file using pyshark

For now we have restricted our attack on our own system. Further the attack can be done any 3rd party system on the same network using MITM(Man In The Middle) and ARP poisoning.

4 Work Done Till Now

Search Engines use AJAX technology to provide search suggestions. Every time a letter is typed in the search box, a request is sent from client browser to the server requesting it to predict what the user might be typing and provide suggestions accordingly. These requests are nothing but packets exchanged between server and the browser.

Observation on Google packets: For any letter typed in the google search box, the packets exchanged were difficult to trace as google sends a large number of irrelevant packets in order to increase the randomness. Further, length of packets varied rapidly for same character tested more than once. For example- when tested for character *'a'* the packet size varied from 67-1250. Thus, no conclusion could be drawn from the packet lengths.

Observations on Bing packets: When the character *'a'* was typed in the search box of the bing, a packet of length 913 was sent. Upon further typing and analysing other characters it can be easily seen that the length of the packets sent were constant, i.e. 913, at least for the same session. Further it was noticed that when second character was typed the size of the second packet being sent incremented by one, i.e. 914, irrespective of the characters. Thus, the sent packets can be used to determine the number of keystrokes made which was correct in approximately **96%** of the cases.

Still the question of guessing the keystrokes from the packet length remain unsolved. The answer lies in the suggestion packets sent by the server as a response to the request made by client. It was observed that the length of the packets received varied from alphabet to alphabet. Further the packets had a fixed part of 716 bytes(for a particular session) which comprise of various http headers irrespective of requests. Thus,

Request length - Header length = Suggestion Length

Moreover the suggestion length was nearly same for a keystroke in **67%** of the cases. The table below shows our observations for some strings after multiple attempts:

String	Packet Length Sent	Packet Length Received(Total)	Suggestion Packet Length	Header Length
a	913	1254	538	716
a	913	1253	537	716
b	913	1260	544	716
b	913	1261	545	716
c	913	1264	548	716
c	913	1263	547	716
ab	914	1238	522	716
ac	914	1234	521	716
ac	914	1238	522	716
ad	914	1265	549	716
ad	914	1265	549	716
ae	914	1230	514	716
ba	914	1283	567	716

Table 1: Packet Analysis Table

Based on above observations we have implemented a python code which uses pyshark library to sniff live packets between browser and server. Again using packet lengths from the server side and mapping them with the observation table we are able to guess the character string that was entered by user to some extent (for strings upto length 3).

We have also implemented a trie data structure consisting of 466544 valid dictionary words. The program takes a string or prefix string as input and returns the lexicographically smallest possible dictionary word.

5 Post Mid-sem goals

1. Deriving an optimised algorithm to correctly predict the strings typed in search box more efficiently.
2. Attacking other AJAX based suggest boxes e.g. amazon.in, ask.com.
3. Giving appropriate ranks to the words in trie based on an efficient ranking algorithm.
4. Implementing attack on a 3rd party system using Man In The Middle and ARP poisoning attack.
5. Other possible ways for the attack apart from the packet length.
6. Suggest preventive measures.

References

[1] wikipedia, “Data breach - a threat.” https://en.wikipedia.org/wiki/Data_breach, 2016. [Online; accessed September 2016].

[2] C. Joseph, “Wikileaks releases secret report on military equipment.” http://www.nysun.com/foreign/wikileaks-releases-secret-report-on-military/62236/, 2007. [Online; accessed September 2017].

[3] E. Kao, “Making search more secure.” https://googleblog.blogspot.in/2011/10/making-search-more-secure.html, 2011. [Online; accessed November 2016].

[4] D. Sullivan, “Post-prism, google confirms quietly moving to make all searches secure, except for ad clicks.” http://searchengineland.com/post-prism-google-secure-searches-172487, 2013. [Online; accessed November 2016].

[5] T. Armerding, “16 biggest data breaches of 21st century.” https://www.csoonline.com/article/2130877/data-breach/the-16-biggest-data-breaches-of-the-21st-century.html, 2017. [Online; accessed September 2017].

[6] “Wireshark - about.” https://www.wireshark.org/, 2017. [Online; accessed September 2017].