Winking Router: A Side Channel Attack Survey

Divyansh Upreti (SBU ID:112026646)
Rohit Aich (SBU ID: 112126618)
Prabuddha Kumar (SBU ID: 112076553)

***Abstract*— A side-channel attack in network security is any attack based on information gained from the implementation of a computer system, rather than weaknesses in the implemented algorithm itself. Timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited.**

General classes of side channel attack include: cache attack- attacks based on attacker's ability to monitor cache accesses made by the victim in a shared physical system as in virtualized environment or a type of cloud service, timing attack- attacks based on measuring how much time various computations and power-monitoring attack- attacks that make use of varying power consumption by the hardware during computation.

We classify our research work under optical attacks in which secrets and sensitive data can be read by visual recording using a high resolution camera, or other devices that have such capabilities. In this paper, we aim to explore the possibility of exploiting the blinking pattern of a switch to identify any webpage a user visits in future. We share our findings, architecture, setup and methodology and mention the scope of future work that can be done in this domain.

I. Introduction

The rapid growth of connected devices and the sensitive data they generate poses a significant challenge for manufacturers seeking to comprehensively protect their devices from attack. Consumers expect their IoT devices and data to be adequately secured against a wide range of vulnerabilities and exploits. High levels of security must now be implemented as a primary design parameter, rather than a tertiary afterthought.

Historically, solutions that included robust encryption/decryption algorithms with cryptographic keys were considered secure. This is because brute force attacks have ultimately become computationally infeasible due to the increased key length of the cryptographic algorithm.

Nevertheless, there is a category of attacks that simply ignore the mathematic properties of a cryptographic system instead focusing on its physical implementation in hardware. More specifically, cryptographic systems routinely leak information about the internal process of computing. In practical terms, this means attackers can exploit various techniques to extract the key and other secret information from the device.

This vector is known as side-channel attacks, which are commonly referred to as SCA. A side-channel attack is a form of reverse engineering. Essentially, side-channel attacks monitor power consumption and electro-magnetic emissions while a device is performing cryptographic operations. Side-channel attacks conducted against electronic devices and systems are relatively simple and inexpensive to execute. This means attackers can exploit various side-channel techniques to gather data and extract secret cryptographic keys.

In this paper, we study if a side channel attack is possible on a switch by observing the blinks it emits while visiting a website. We analyze if it is at all possible for a an attacker to use the blink pattern to identify the websites a user is trying to visit. If such an attack is possible, we believe that it could have a devastating affect on user's security and his browsing habits as patterns could easily be studied by observing the switch in network. Such attacks can happen on networks ranging from small home networks to larger intelligence agencies networks.

We list our contributions as follows:

- We study the possibility of exploiting side channel attacks on switches using their blinking patterns by conducting a systematic study under controlled network conditions.
- We try to identify a relation between blinks of switches and incoming and outgoing packets.
- We observe and record blink patterns of top Alexa websites on different network speeds and create a dataset of these fingerprints.
- We apply various machine learning techniques on these finger prints and provide our study and analysis of the results obtained.

Further we aim to extend the boundaries of our work by sampling different websites at different speeds and listing down the results, observations and limitations.

The sections of this paper are laid out as follows. Section II provides information about related work in the domain of side channel attacks, section III explains the architecture and methodology of our work. This is followed by a section on results and discussions and a concluding section. We later provide the future scope of our work in section VII followed by team's contribution.

II. Related Work

In this section, we discuss past works and surveys on SCA through different network techniques on variety of

network components.

Xun Gong et al. [1] analyze the web traffic. They have shown that traffic patterns leaked through side channels can be used to gain sensitive information. Attackers can find out which website, or webpage a user is accessing simply by monitoring the packet size distribution. The attacker sends frequent probes and measures the sizes, timings and counts of packets arriving. If the training and test data collected is from the same location, large fraction of sites can be accurately detected. They study on how traffic analysis can be a serious threat to Internet privacy as it can be carried out remotely, without access to the analyzed traffic, thus greatly increasing the scope of attack. There was some accuracy loss due to differences in test and training environments but overall It was found that despite working with noisier information source than previous related works, this website detection attack nevertheless showed that remote traffic analysis is a serious threat to Internet privacy.

The study done by Kadloor et al. [2] shows that a low-rate sequence of probes sent to a DSL router can give out essential information about the traffic timing and volume information, if only the router is connected to a public domain, like the internet. The low rate of probes would not cause a significant delay in the router response, and hence the user would also be oblivious of such nefarious attempts. The attackers would be easily able to measure the round trip times of the packets they sent through the router, and based on the scheduling algorithm, FCFS or round-robin, they could almost accurately predict the user-activity by 90%. Earlier, it was assumed that in these attacks, the attacker should require access to the victim's traffic, i.e., it should be able to access the victim's router. However, new research proves that no such linkage is necessary, and indeed any one who could send messages to the victim's router could pull this off by studying the network packets.

Coull et al. [3], examine the extent of privacy of data in encrypted messaging services by examining Apple iMessage, which is an end to end encrypted service. They find out that it is possible to extract some information from the packets such as the language of the message, the size of the content, some operating system information among others. They also suggest a countermeasure by inserting random padding to the packets. This is relevant to our project in that they provide remedies to mitigate snooping by side channel attacks. Several studies have also been done to use side-channel attacks to create specific fingerprints of different websites, based on the traffic of network packets or round-trip-timing of network packets (RTT).

Research conducted by Ling et al [4] shows that it is possible to predict the websites the victim has visited by measuring the round-trip-times (RTT) from the victims machine to destination websites. They derive a model by fingerprinting the websites based on the RTT, and then conduct a side-channel attack where they try to predict the websites visited by users based on the RTT-s recorded, with a specific probability factor. They also give out possible solutions to counteract the attack, like using the k-means clustering and the K-anonymity algorithm to achieve similar RTT-s for different websites, so that they cannot be distinguished.

Hermann et al. [5] use a Multinomial Nave Bayes Classifier to detect websites with an accuracy of 97 %. They use website fingerprinting to learn the packet distribution of a website, and use the fingerprint of websites for classification. They mention that this technique is extremely effective against single hop techniques under closed world assumptions. They do, however concede that if the closed world assumptions are left out, the accuracy is so far not great, but getting better.

Conducting a different study on profiling attacks, Liberatore, et al. [6] examine the effectiveness of network analysis techniques for identifying encrypted HTTP streams. They use classification algorithms to identify encrypted traffic on the basis of similarities of features when compared to known profiles. They show that these techniques can work at significant scale. The paper also suggests countermeasures to prevent unwarranted snooping. The key highlight is that the authors claim that traces of websites from packets can be identified 66-90% of times, under a set of assumptions.

In specific cases, side channel analyses on network packet lengths have also revealed information about encrypted Voice-over-Internet (VoIP) calls. Wright et al [7] shows that even the encrypted VoIP calls are not safe against side-channel attacks with network packets. As the VoIP calls are encrypted with variable bit rate (VBR) and length-preserving encryption to save bandwidth, the researchers have been able to analyze the length and demographics of the output VoIP packets to identify the phrases spoken within the call with an accuracy of 50%. In certain cases, they have been even able to achieve an accuracy of 90%. They constitute a model by considering the constitution of phonemes, and predict how the length of network packets varied over different phrases. They later fit this model to the output packets to predict the encrypted phrases. This shows that the side-channel attacks involving network packets are so efficient that the encryption efforts to secure the data have become useless.

In general, side-channel attacks to determine users online activity is a novel topic for the researchers. The study done by Qing Yang et al [8] shows that public USB charging stations pose a significant privacy risk to smartphone users even when no data communication is possible between the station and the users mobile device. Their results show that the attack is highly successful and they were able to achieve over 90% webpage identification accuracy. This work is studies side channel attack on smartphones under

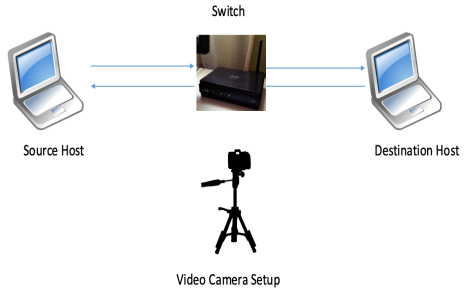


Fig. 1. Research Setup: Phase 1

phone constraints demonstrates that websites can be correctly identified within a short timespan of two to six seconds.

The researchers use machine learning algorithms to identify the web-pages the user visited, and exploits smartphones factors such as user interaction with touchscreen, WiFi and LTE connectivity, training and testing device mismatch, type of connection (HTTP or HTTPS) and relative location of the host serving the webpages to the smartphone. They present a side channel attack allowing a charging station to identify which web-pages are loaded while smartphone is charging. They collect power traces of Alexa top 50 websites on multiple smartphones under several conditions, including battery charger level, browser cache enabled/disabled, taps on the screen, WiFi/LTE, TLS encryption enabled/disabled and time elapsed between collection of training and testing data, website location and were able to achieve identification accuracies as high as 98.8% with 2 second traces. Attacks using other websites to infer the victims online activity from browser caches and history have also been common.

Edward W. Felten et al [9] studied the attacks that allow a malicious web site to determine whether or not the user has recently visited some other, unrelated web page. This malicious page can determine this by measuring the time the users browser requires to perform certain operations. As browsers perform various forms of caching, the time required for operations depends on users browsing history. By measuring the time required to access an element, malicious program can tell if the element is in a nearby cache and thus learn that element has been accessed recently. For example, an insurance-company site could determine whether the user has recently visited Web sites relating to a particular medical condition. This is a dangerous side attack to conceal users browsing history and figure out his general browsing patterns.

III. Architecture & Methodology

As a part of our research, we had to construct an elaborate setup. We used an Ethernet Switch (Model no.: TRENDnet TEG S50G 5-Port Gigabit Ethernet Switch), connected with laptops and LAN internet port by Ethernet cables. We had a smart-phone camera, steadily held with a tripod, with which we recorded the blink patterns of the LED-s of the switch as we accessed different destinations through it. Our data collection constituted of two phases; in the first phase, we tried to access locally hosted sample websites through the switch in a fully controlled environment, while in the second phase, we scaled our research up and tried to access top Alexa websites with varying internet speeds. The below subsections explains the above phases in detail.

A. Phase One: Trying to access a locally hosted sample website

In this phase, we wanted to check the basic behavior of the switch, and understand the granularity of the blink of its LED-s; i.e. we wanted to understand how many times the LED-s blink for each network packet transfer.

1) To achieve this, we set up an Oracle Virtualbox instance inside one of our laptops, and installed a fresh Microsoft windows instance inside it. The purpose of having a fresh OS setup was to nullify the passive data transfers through the router. We established a bridge network connection between the Host OS instance (mac OS) and the virtualbox instance, and made sure that no passive network packets pass through the router.
2) Next, we hosted an Apache Tomcat and a Xampp server instance inside our nested operating system, and deployed sample web-applications on them. We carefully designed and chose our websites, so that they had different constituent technologies (like JSP-s, Servlet-s, PHP, different CSS, HTML and Angular JS contents), and would be able to produce different packet signatures when accessed from a remote host.
3) In parallel, we also had instances of WireShark running in the source host's end. We set up specific static IP-s for both the source and destination machines, and filtered WireShark with those IP-s. With this setup, we intended to find out how many packets were being traced in each time-stamp, so that they could be matched with the number of blinks captured in the camera at the similar time.
4) We used our fixed camera all this while to record videos of the switch blinks. We subjected these videos to our blink collection algorithm written with OpenCV (described in detail in the section 'Data Collection'), and collected the number of times the switch was blinking, while accessing our own websites hosted in the remote servers inside the virtualbox OS instance.

B. Results and inferences from the first phase of experiment

We primarily discerned two critical outcomes from our first experiment explained above. Firstly, we found out that since the two hosts (source and destination) are connected with each other over a high-speed wired and noiseless network, data transfer is extremely fast. As a result, any sample websites that we tried to access would get loaded

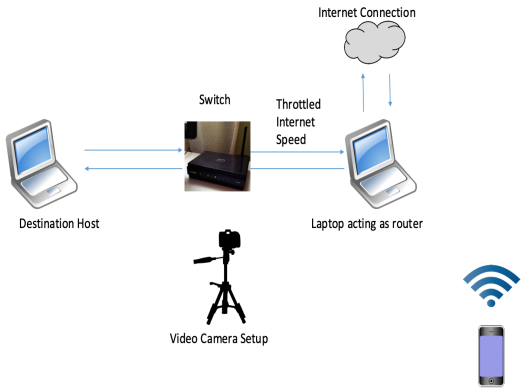


Fig. 2. Research Setup: Phase 2

almost instantaneously. Our samples show that the websites, irrespective of the data-load, would load within 1.6 seconds every time. The WireShark packet traces confirmed this observation. However, in those 1.6 seconds, our switch would blink only once. As a result, we were unable to view blink patterns. We inferred that the speed of incoming data packets had to be throttled down to make the load lengthy, and to allow the switch to blink for a while, so that we can record possible patterns of lights.

Also, we found that there were a lot of TCP packets that the switch sent to both the hosts (SYN-ACK) messages. These packets logged a lot of blinks in the switch, but that was after the main data-loading. However, we found that there is indeed a relation between incoming and outgoing network packets and switch blinks. Of course, this relation would vary depending on parameters like internet speed and switch model, but this observation shows that at least there exists a transitive relation between the blinks and the websites being visited, and that could be exploited to create a side-channel attack model. We decided to make our channel even less noisier as we scale up the experiment to the second phase, i.e., hit the top Alexa Websites.

C. Phase Two: Trying to access top Alexa websites

This was the main part of of our experiment. We decided to change the setup from the previous one a bit. Firstly, connecting to the internet via wired LAN port would give us a very speedy internet connection. We fathomed that this could lead to the same results as the first phase, and then we would not be able to record blink patterns from our switch. We decided to throttle down the internet speed manually. To confirm our approach, we connected the switch to one of the available LAN internet ports in the university campus by an Ethernet cable, and connected one of the laptops to it. We then tried to hit simple website as google.com or reddit.com. We found that not only the data got loaded tremendously fast, the university internet had a lot of monitoring activities going on, and sent out a lot of multicast packets to all the connected devices. As a result, we found out from WireShark that over a minute after accessing a simple website's landing page, 90% of the recorded packets were actually non-HTTP multi-cast packets sent by SBU.

Furthermore, we tried to throttle down the speed by connecting one of the laptops to the SBU WiFi, and use it as a Wifi to Ethernet router. We connected our switch to this laptop, and connected another laptop to the switch. This way, we could control the internet speed by throttling it down by the former laptop (now acting as a router), and access websites from the latter laptop through the switch. Unfortunately, the huge number of multicast IGMP packets amounted to a lot of noise, and we could not deduce a particular pattern for the destination websites.

Under this circumstances, we decided to move away from the SBU network, and use the normal AT&T internet available over common smart-phones and broadband across United States. We describe this experimental setup below, and also show how we collected the data.

1) We connected a smart-phone to AT&T WiFi, turned it into a internet hot-spot, and connected a the router laptop with it. The laptop we used as a router did not have background traffic running in order to block tertiary packet flow. We configured this laptop as a wireless to Ethernet internet router, and throttled down the upload and download speed to 512 KBps.
2) We connected our switch to this router machine, and in turn connected another laptop to the switch. We configured the second laptop to be a client, which would consume the internet via the switch, and would be able to access real websites with it. The background services were stopped here as well to collect perfect samples. We did use incognito mode in our browsers and disable cookies. For each hit, we would close and reopen the browser in incognito mode, so that we can get consistent data samples without any contamination from browser cache and cookies. We also had an instance of Wireshark running on the client machine to monitor exceptional and unusual flow of network packets.
3) We then started the real time data collection, and recorded the blink patterns while we accessed real-time web-pages. We collected blink data for five of the top 10 Alexa websites, google.com, facebook.com, youtube.com, twitter.com and linkedin.com for a 512 KBps internet speed.
4) We throttled our speed up and down from 16 MBps to 256 KBps for different websites, and tried to see how much the blink pattern changes with changing the internet speed.

These above experiments were performed in ideal lighting conditions, in dark rooms, without the present of any external

lighting source other than the router blinks. However, later we have conducted the same experiment in normal conditions, and have found it to be working smoothly. The below section describes the methods of data-collection in detail.

D. Data Collection Procedure

We throttled the internet speed through the switch in order to get consistent samples of reasonable transfer speed. Initially, we throttle down the speed to chose 512 KBps transfer speed to record the samples. We used an Mac OS utility app, 'Network Link Conditioner' to throttle the speed in simple multiples of two. This ensured that we were able to collect feasible data transfer speed limit on the packets. We then collected around 60-100 samples per website, recording video at each sample. After that, the sample was used in OpenCV to get the number of blinks per unit time.

OpenCV, a framework for processing images and videos, was used to detect blinks and generate a data-set of time-stamps of switch blinks of each singular website sample. We used OpenCV to iterate through these videos and subtracted two consecutive frames in order to detect the precise regions of change. This technique detected when exactly those switch LED blinks were occurring. We defined a threshold value to measure the illumination of each frame, and when the value of the recorded illumination of a frame crossed that particular threshold, we deduced that a blink must have been recorded. On the other hand, when the illumination came below the pre-defined threshold, we inferred that the current blink had ended. Using this methodology, we recorded the time-stamps of the blinks.

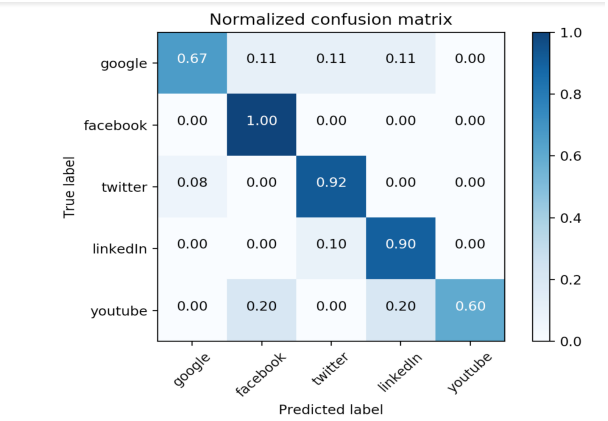
The link to our GitHub Repository is: **here.**

E. Data Analysis

To analyze the information from this, we use machine learning techniques like bag of words and normalization.

Bag of words is used to generate features by defining a range of values and adding the frequency of occurrence of an attribute within that range. By applying bag of words to the timestamps of the blink patterns of each website, we are able to obtain sigature patterns for each patterns. By varying the bag size, we can control the granularity with which we aim to generate the signatures. A larger batch size would give coarse signatures, whereas making the batch size smaller and smaller would give very fine fingerprints, approaching the use of individual timestamps. This approach of using finer timestamps requires the collection of a larger amount of data to account for the effect of noise on the values in each bag. We have experimented with different batch sizes and observed the effects on accuracy. These observations are presented in the result section.

Normalization is another technique we use with multiple aims in mind. It refers to bringing the timestamp sample of each website to a [0,1] space by dividing by the total time



taken for the website to load. This presents the advantage of making the sample theoretically invariant to some changes in the time taken for a website. Without using normalization, the classifier tends to become very sensitive to the time taken for each website to load. When loading over different network speeds, this can cause misclassification because of different load times. Normalization tends to take away the element of loading time, by retaining the signature distribution of packet arrival times. We have experimented by varying the speeds of the internet connection and have observed the effect that has on the normalized classification accuracy. The results of this experiment have been listed in the result section.

By using these techniques, we generate datasets of reasonable quality, which help use to form approximate fingerprints of these websites.

We then use an SVM classifier in order to classify based on the loading patterns. We perform one-vs-one classification, as well as one-vs-all classification to test if classification is successful when comparing two websites and all websites respectively.

Due to the data size being smaller than ideal, we use K-Fold cross validation in order to perform a fairer assessment of the classification accuracy.

IV. Results & Discussions

A. Inferences from the data

Data receieved:blinks and time taken for websites to load We collected the data and observed some interesting instances. Since every website has vastly different amounts of information present on the landing page, the number of packets and their distributions would vary a lot. This would impact the number of blinks, as well as the time for each website to load.

Fig. 4 lays down the number of blinks alongside the total time taken for each website to load. As expected, websites which took longer to load had a larger number of blinks.

Website	No. of blinks	Time to load (s)
Google	117.23	13.59
Facebook	193.96	26.64
Twitter	121.56	15.52
LinkedIn	82.96	9.53
Youtube	343.5	31.00

Fig. 4. Table of blink comparison

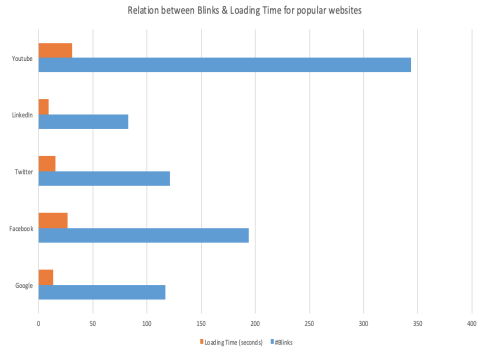


Fig. 5. Blinks and loading time

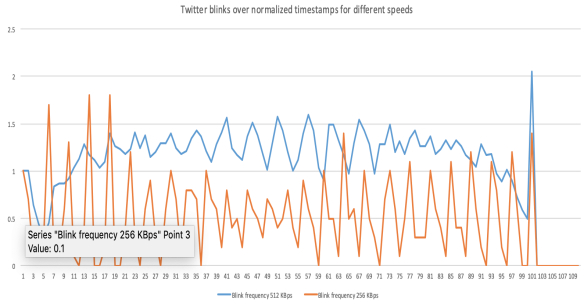


Fig. 6. Comparison of Twitter Response

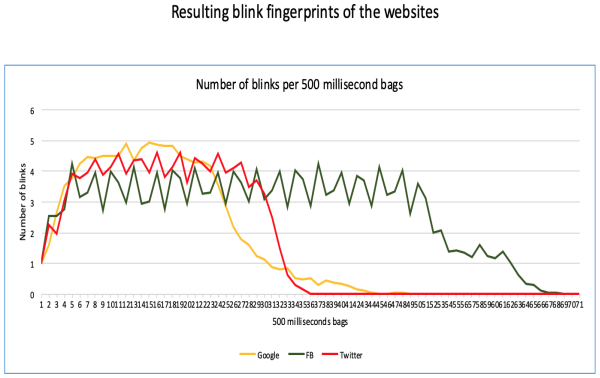


Fig. 7. Fingerprint of websites

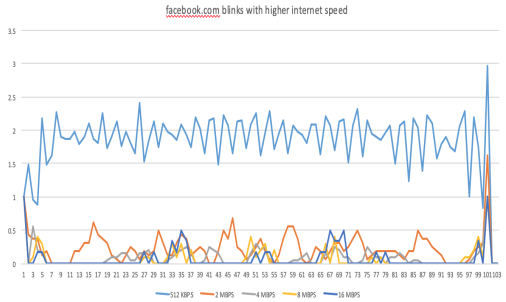


Fig. 8. fb speed comparison

Moreover, websites like LinkedIn, Google and Twitter are in the same territory of number of blinks, while facebook and youtube have similar load times to one another. This leads to the hypothesis that classification among LinkedIn, Google and Twitter would be tougher than say, Google vs LinkedIn. We explore these results in the section below.

B. Experiments run with fixed speeds

1) One vs One classification: We trained a one-vs-one SVM classifier to compare between two websites at time and observe the classification accuracy. As mentioned above, we used K-Fold Cross Validation in order to run multiple trials with different train and validation each time and aggregate the results.
We found extremely good accuracy when running the one-vs-one classifier on the normalized data. Note that the bag size in this particular trial is 0.01, out of a total range of 1
   - For google vs facebook, we observed an accuracy of 96.4%,
   - For facebook vs twitter, we observed an accuracy of 99.1%
   - For google vs twitter, we observed an accuracy of 65%

- For youtube vs facebook, we observed an accuracy of 72.2%

We observed excellent classification accuracy for google vs facebook and twitter vs facebook. We attribute this to the sheer difference in the number of blinks of the respective websites. However, things become interesting when we compare google and twitter. Due to the two websites having very similar load times and number of blinks, classification becomes more challenging. However, we believe that with more data, we can obtain better results.

We also observe the results for different bag sizes, varying from 0.001 to 0.5 to 1. A bag size of 1 basically is a classifier running on the total number of blinks. As such, it is extremely bad at distinguishing between google and twitter, with an accuracy of 50%.

One vs All classification:

We also trained a classifier to differentiate between all the websites in the sample simultaneously. We expected difficulty for the classifier given that some websites were similar in terms of time taken and blinks received. Additionally, the length had become a non-factor, given that we were normalizing. However. the classifier had relatively good performance, as can be seen in the Fig 3.

Facebook is classified correctly all the time, while youtube, being similar, is misclassified as facebook 20% of the time. As expected, google is difficult to classify, succeeding only 60% of the time.

Speed variation:

In order to observe the robustness of the model (scale-invariance), we collected some samples of facebook and twitter for different speeds. For twitter, we throttled the speed down to 256kbps, while for facebook, we attemped higher speeds(2Mbps, 4Mpbs, 8Mbps and 16Mbps).

Although we were aware that there would be a change in the blink pattern, as with increased speed, there would be combination of higher number of packets into one blink, and the opposite effect at lower speeds.

Even so, we attemped to see the accuracy of the model at different speeds. We trained the model as before, using the normalized datasets at 512 Kbps. Then we tested the accuracy of the trained model on the Twitter-256Kbps and Facebook-2/4/8/16Mbps models.

We found that for twitter, the accuracy went down to 10%, while for facebook, the accuracy of:

- 68.75%, for 2Mbps
- 68.1% for 4Mbps
- 50% for 8 Mbps
- 50% for 16Mbps

We see that the model shows reasonable robustness to scale; a model trained on 512Kbps shows 68% accuracy upto 4Mbps, ie. 8 times the network speed of the training data.

V. Limitations

Despite the successful results of classification, there are some limitation of our work that are listed below:

- Currently, we require still samples from a camera that does not move during the sampling
- We sample at relatively modest speeds, which are now not found commonly.
- The sample that we train on is collected while the website is loading. This scenario may not occur upto 80% of a user's browsing
- We create an environment with minimal background network traffic. This is far from real conditions.

VI. Conclusion

Side-channel attacks are powerful and effective. Redesigning hardware is expensive and an attacker can easily steal secret information from unusual channels as in this case from blinking pattern of a switch. In this paper, we have presented our analysis on the possibility of side channel attack on switch by its blinking pattern. It is clear that such an attack could have a catastrophic effect on user's security revealing his browsing habits. We have tried to identify relation between blinks and incoming and outgoing packets and applied various machine learning techniques on the collected datasets at various network speeds to see if it is possible to identify a website through blinks of switch. We aim to extend this work by sampling different websites at various speeds and welcome any other novel ideas and future works in this domain. We next discuss some of the future works that can be done.

VII. Future work

There are certain aspects that are can still be reviewed:

- One can expand upon the idea of normalization to enhance the invariance to speed
- With a big dataset, one can use fourier analysis to seperate the different blink patterns that are being mixed by treating them as signals. This can solve the problem of noise.
- By exploiting the packet transfer once a website has been loaded, one can determine patterns and classify websites.

VIII. Contribution

- Prabuddha Kumar: OpenCV and Machine Learning
- Rohit Aich: Setup, data collection and analysis, presentation and documentation.
- Divyansh Upreti: Setup, data collection and analysis, presentation and documentation.

References

[1] Website Detection Using Remote Traffic Analysis http://publish.illinois.edu/science-of-security-lablet files/2017/03/Website-Detection-Using--Traffic-Analysis.pdf

[2] Low-Cost Side Channel Remote Traffic Analysis Attack in Packet Networks https://ieeexplore.ieee.org/document/5501972/citations

[3] Traffic Analysis of Encrypted Messaging Services: Apple iMessage and Beyond http://www.sigcomm.org/sites/default/files/ccr//2014/October/0000000-0000000.pdf

[4] A Novel Network Delay Based Side-Channel Attack: Modeling and Defense https://ieeexplore.ieee.org/stamp/stamp.?tp=arnumber=6195628
[5] Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Nave-Bayes Classifier http://www.cs.jhu.edu/ sdoshi/jhuisi650/ /spimacs/SPIMACS_CD/ccsw/p31.pdf
[6] Inferring the source of encrypted HTTP connections https://dl.acm.org/citation.cfm?id=1180437
[7] Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations http://cs.jhu.edu/ cwright/oakland08.pdf
[8] On Inferring Browsing Activity on Smartphones via USB Power Analysis Side-Channel https://ieeexplore.ieee.org/document/7782756/
[9] Timing attacks on Web Privacy https://users.ece.cmu.edu/ vsekar/Teaching/17/18731/papers/dnssidechannel.pdf
[10] Side-channel attack. https://en.wikipedia.org/wiki/Side-channel_attack
[11] An introduction to side-channel attacks. https://www.rambus.com/blogs/an-introduction-to-side-channel-attacks/