Blind Mice: Compromising Web Browsing Privacy Through Mouse Clicks

Vikash Kumar, Aditya Sankar, Sidhant Gupta

Abstract

Many computer systems that are otherwise secure are vulnerable to side channel attacks. In this work we show how mouse movement and clicks data can leak sensitive information about which webpage a user is browsing based only on the pattern of mouse clicks and how they differ according to a webpage's layout and structure. Such information leakage is concerning especially when the proliferation of wireless mice is taken into consideration. Through our initial experiments and hardware setup, we show that it is possible for an adversary to sniff mouse data at a distance without the user's knowledge and permission. We collected mouse movement and clicks data (along with the ground truth as to which website was being browsed), for several users over a period of 6 weeks, by means of a Google Chrome extension. We then developed a preliminary probabilistic classifier to identify unique *click patterns* produced when browsing a certain website. In our experiments, we could successfully identify which webpage the user is browsing from a known corpus of websites with a mean accuracy of 77% given a single click and 100% when given 10 consecutive clicks. We also propose potential defenses and highlight opportunity for future work and extensions that may significantly improve our findings.

Introduction and Prior Research

Side channel attacks on an otherwise secure system is a well-studied discipline within the security community. Pioneering work in analysis of risks associated with various side channel attacks was performed by Eck et. al [1] in mid 1980s. Since then, within the academic community several side channel attacks have been identified and demonstrated, such as differential power measurements for cryptanalysis [2] [3], timing attacks on SSH by monitoring keystrokes patterns [4], electromagnetic interference for inferring a CRT and LCD's screen contents [5] [6] [7] [8], and more recently inferring human activity and privacy leakage by analyzing a home's or a device's power consumption [9].

In this paper we examine information leakage from computer peripherals, in particular a computer mouse. The use of wireless computer peripheral is quickly becoming ubiquitous due to increasing numbers of people who browse the Internet and do business on their mobile devices. Analyzing the information sent between the peripherals and the computer could reveal information about how the computer is being used and often *what* it's being used for. It is well known that by tapping into the information being sent between a keyboard and a PC, for example by use of a physical or software *keylogger* one can learn quite a lot of information about the user, including passwords and other sensitive data. Such key loggers can either be purely software based, that hook onto certain system calls or purely hardware. The latter are generally installed in line with a keyboard, that is, the keyboard being compromised is plugged into the logger and the logger is plugged into the PC. It does not alter the function of the keyboard itself, but records

each and every keystroke sent to the PC. Key loggers and defenses against them have extensively been studied both in academia and in the industry. Harley et. al propose a scheme to inject random keystrokes by switching the focus away from a textbox and then refocusing on it so that the key logger records data injected with random key strokes [10]. Defenses like these in addition to defenses against software key loggers, which are akin to antiviruses, are not effective when using a side channel attack.

In [11] and [12] authors compromise data sent from a keyboard using acoustic and electromagnetic interference emitted, which are immune to many of the common defenses against key loggers. To tackle key logging on a broader spectrum many security companies and services such as online banking, stock trading and tax filing services have proposed and implemented the use of an on screen keyboard. Figure A-1 (appendix) shows an online banking website that makes use of an on-screen keyboard as its primary method of inputting user password.

The use of an on-screen keyboard is based on the premise that a user will use their mouse to input information, thus making the input secure. This misconception that mouse data is "invaluable" is further strengthened by the fact that we were unable to find any hardware mouse logger and very few software packages that record mouse data. In this paper we wanted to challenge such a premise and ask – *What sensitive information can be learnt from mouse movement and mouse clicks data?* In particular, we chose to examine if private browsing patterns of a user, that is: what website a user is browsing can be inferred from such mouse data, and we find that the answer is **yes**!

As mentioned before, the use of wireless computer peripherals is rapidly increasing and this makes it more important to protect the mouse data. As we demonstrate in this work, it is likely that mouse movement and click data can be sniffed wirelessly. Thus imagine, an adversary going to a café or even in an office building and record mouse movement data and figure out not only what website someone is browsing but potentially the passwords (assuming the website makes use of on-screen keyboard) to those websites.

Our work is not the first to use mouse movement and clicks data to infer human activity on a computer, though, to the best of our knowledge, it is the first to use to infer what website the user is browsing. Prior work in the security community has primarily focused on using a user's mouse usage patterns to augment authentication and make it more secure. [13] [14] both propose making use of unique mouse movement patterns for user authentication. Thus, like our work, this further bolsters the importance of (1) mouse data being more valuable than it is generally considered, (2) the fact that mouse data can be a source of sensitive information leakage. Another recent development in mice hardware has been the incorporation of fingerprint scanner and other biometric sensing systems. Though we do not investigate such mice in this paper, we would like to highlight that protecting data transmission from mouse to PC, especially over a wireless medium thus becomes even more important.

Background

Mouse Click Patterns

Patterns of mouse clicks are commonly used in HCI research to better understand how users navigate through a user interface (UI) and which graphical elements of the UI are most commonly used in order to identify areas of improvement. Such click patterns are visualized using Heat Maps, which are essentially 2-D scatter plots, with the color in a region being brighter depending on the number of points. Thus greater number of points produces a brighter region. As our first step to test the hypothesis that mouse click data can yield interesting information, we collected mouse clicks data along with the co-ordinates of clicks for a short browsing session. Figure 1 shows heat maps overlaid on top of the two websites used in this experiment. Observe how the heat maps and the *hot spots*, that is the regions of most clicks are quite different. We build on this intuition to develop a probabilistic classifier (as described later in this paper) to classify, based on the mouse click data, which website the user is browsing.

Figure 1: Heat maps of mouse click data overlaid on the website. Observe the "hot spots" and their unique position and general spread over the two website.

Wired and Wireless Mice

Mouse movement and click data can potentially be sniffed from both a wired or wireless mouse. In addition, much like key loggers, it can either be a completely hardware based implementation or a software only. There are a few software packages available that can log all keyboard and mouse activity on a computer. Unlike, such software, a hardware implemented mouse logger only gives delta changes in mouse motion. That is, there is no notion of an origin. In this paper, for the purpose of highlighting the potential information leakage from mouse data, we assume an origin, that is mouse movement and click data is collected in software.

This paper is organized in two main themes – first we will present how data from a wired and wireless mouse can be sniffed. We then detail how we reverse-engineered a popular wireless mice to reach a conclusion that it is very likely that data from such mice can be remotely ac-

quired without permission and attracting attention of its user. Second – given such mouse movement and clicks data, we show what sensitive information is leaked. In particular, we show that private browsing patterns of a user can be inferred, given less than 10 consecutive clicks.

Sniffing Mouse Data

Wired Mouse

Sniffing data from wired mouse was performed by use of inline hardware. In particular, as a proof-of-concept, we connected a USB Logic analyzer and captured the various signals generated when the mouse sends movement and clicks data. The data sent from either a USB or PS/2 mouse was similar though the protocol differed. In both cases, the mouse data consisted of a 4-byte packet. Byte 2 and byte 1 encode delta X and delta Y respectively, byte 3 encodes the amount the scroll wheel was turned and the lower 3 bits of the 1st byte encode which of the three mouse buttons is pressed. Since a wireless mouse is of most interest and can allow data capture at a distance, we chose to investigate wireless mice rather than create logging hardware for a wired mouse.

Wireless Mouse

For compromising data from wireless mice, we procured two brands of mice – Logitech and Microsoft. Both were 2.4 GHz version and were marked as most popular wireless mouse on Amazon.com. In this paper we present a detailed analysis of reverse engineering the Logitech mouse. We chose Logitech, since we have 3 of Logitech and thus could perform more extensive tests. Nonetheless, we believe our findings and approach to reverse engineering is equally applicable to other mice brands.

On the Logitech mouse, we found that it uses a 2.4 GHz channel in the ISM band, with a proprietary protocol called the "Enhanced Shockburst" (Figure A-2 (appendix) shows the IC that we identified). This protocol and the baseband IC are both from Nordic Semiconductors. Figure 2 shows the protocol packet structure, as described in the product specification.

Our initial approach was to assume that we could use a similar Nordic semiconductor transceiver IC to communicate with the mouse, but we found that a transceiver can only communicate, that is, send or receive data from another transceiver if it knows the **3-5 byte address.** Thus without this address, the IC would not let us read the data. Thus, we needed to figure out the address. This address can be figured out by analyzing the baseband signal at 2.4 Ghz, that is we need to examine the bits being sent over the air. The direct way to accomplish this is to make use of the Nordic transceiver IC itself, since it is tuned to 2.4 Ghz, can be set to any channel, or rather of particular interest to us, we know it can efficiently communicate with the mouse. That is, an IC that allows access to baseband outputs the raw bits and bytes that the RF frontend captures with-

Figure 2: Enhanced ShockBurst protocol packet.

out processing them through any software protocol stack. This would allow one to see all packets being sent over the air. This is analogous to putting a WiFi Ethernet card in promiscuous mode, such that all packets are captured and output irrespective of the MAC address. After pouring over the datasheet for this IC, we found that it *does not* allow access to the baseband, which is the IC would not output the bits it sees on the RF channel.

Next, we identified a 3rd party 2.4 Ghz ISM band IC that does allow access to baseband signal and uses similar frequencies and channels as the Nordic RF IC, but due to logistical issues we could not procure it (minimum order of 1000 pieces, shipped from China), thus our next step was to use other tools and build our own RF frontend and demodulator from scratch. This would require considerable more effort and resources than if were to simply make use of the 3rd party IC.

Though we knew the device transmits in the 2.4 Ghz ISM band, we did not know what particular channel or frequency the mice we had employ. We tried to find these frequencies using a spectrum analyzer but due to various other signals at 2.4 Ghz (such as WLAN), it proved extremely difficult, and thus we began to look elsewhere for hints. On closely inspecting the circuit board of the mouse's receiver module (the one that plugs into the PC's USB port), we noticed a certain set of numbers. We immediately identified these numbers as a frequency for one of the 125 channels in the ISM band. Figure 3 (left) shows the circuit board and the number we identified. We then tuned our Tektronix SA2600 spectrum analyzer to this frequency and immediately saw the signal every time the mouse transmitted. To be sure that is the mouse and that we are not picking up WiFi, we turned off the mouse and turned it on several times and saw no signal when the mouse was off. Figure 3 (right) shows the spectrum analyzer and the signal. The signal itself is modulated as GFSK (Gaussian Frequency Shift Keying). Notice the two small peaks on top of the broader peak. The two peaks are actually 1 and 0, which is the left peak represents 1 and the right a 0. At any given instant only one is transmitted, but due to certain limitation of the spec-

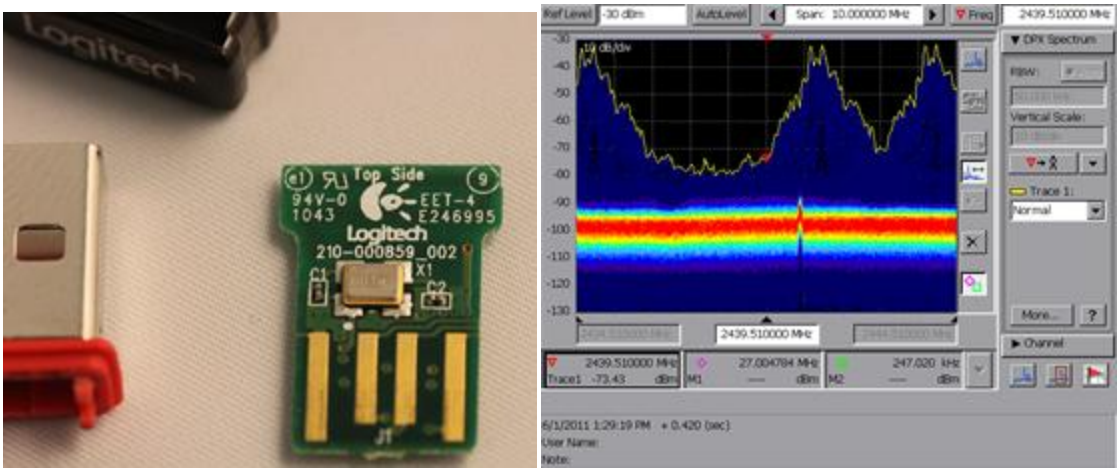

Figure 3: (Left) Circuit board for the receiver side of the mouse. The number on top right – E246995, is the exact frequency this board is programmed for, that is, 2.46995 Ghz ISM band.
(Right) Spectrum analyzer tuned to the mouse's transceiver channel frequency. The two tiny peaks on top of the broader peak is data being modulated in GFSK.

trum analyzer, to be specific, it is not fast enough, we see both peaks at the same time. To extract the address, we need to demodulate this signal.

We made use of a USRP (Universal Software Radio Peripheral) platform to capture and further analyze these signals. The USRP is an open source and flexible software radio platform that can be outfitted with a variety of RF frontends. For our purposes we used the XCRV2450 2.4 and 5.0 Ghz transceiver frontend (Figure A-4 shows our USRP setup). The USRP digitizes the RF signal from the frontend, conditions it and streams it over to a PC over USB connection for further processing. The software on PC that manages the USRP is GNURadio, which not only allows capture of data but also includes a large library of signal processing tools for realizing various radio protocols. On the PC side, we chose to log the data and post-process it in Matlab for ease of processing instead of using GNURadio. The downside of this approach is that the system cannot demodulate the signal in realtime, but this can easily be overcome by programming the correct demodulation scheme in C++ and adding it to GNURadio's signal processing blocks library.

Once the RF data was logged and imported into Matlab, we made use of an off-the-shelf GFSK demodulator and recovered the address bytes as: 0xF8 0x15 0x15 0x78 0x09. The location of the address bytes in the RF packet was estimated based on the ShockBurst protocol as described earlier. The last step in sniffing mouse data packets is to use the extracted address and program it in the address register of Nordic RF transceiver and confirm: (1) the address extracted is correct (2) with the correct address, mouse data packets can be received. We were unable to perform this last step due to unavailability of a Nordic RF hardware development kit, nonetheless, given our understanding of the IC's working, we are confident that mouse data packets can be recovered once the address is programmed.

Next, assuming that mouse data packets can be recovered, we examine if sensitive browsing patterns are leaked via this data.

Methodology

To test if a website could be inferred given mouse data, we needed to build a model-based learning algorithm. This section describes how we collected the data to perform experiments, how we process them and then build models for our probabilistic classifier.

Data Collection and Experiment Setup

In order to build a training data set, we collected actual mouse usage data from several users over the course of one month. This was accomplished by writing an open-source [15] Chrome extension that logs the web usage activity of consenting study participants (see Figure 4 (left)).

The Chrome extension operates by making use of the content script framework [16] that allows JavaScript files to be run in the context of web pages. By using the standard document object model (DOM), we are able to read details of web browser page visits by injecting a small, benign piece of JavaScript that observes and logs user activity. The script works by hooking onto the

OnMouseDown event of the parent document object and hence is alerted when a click occurs anywhere within the web page being browsed. Once invoked, the script uses HTML5 LocalStorage [17] to record information about the click. Our current version of the extension logs the domain of the webpage (example "www.facebook.com"), a Boolean flag indicating whether the page was the domain homepage or a child page (we did not record full URLs for privacy reasons), the (x, y) position of the click relative to the document origin, (w, h) the inner width and height of the browser window and finally a timestamp for when the click occurred. [1]

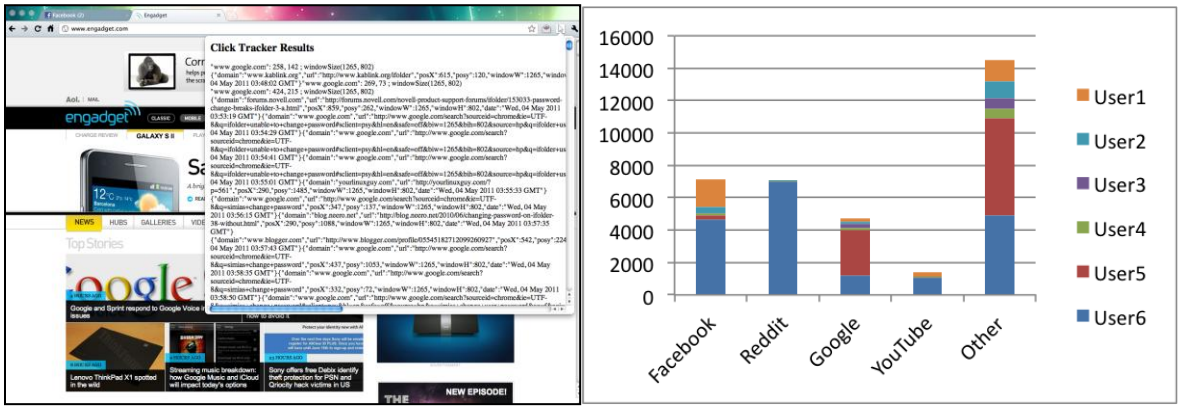

Figure 4: (Left) Google Chrome extension running in the background to collect mouse click and website data. The presence of a mouse icon in the upper right corner indicates that data is currently being recorded. (Right) Distribution of clicks per website across users.

Data Processing steps

At the end of the study, participants submitted the click tracker results to us via e-mail. The data was analyzed to determine popular websites which could be used as exemplars for our learning algorithm. We chose four websites that had a large volume of clicks across all users. The websites chosen were: Facebook, Google, Reddit and YouTube. Our study consisted of 6 participants, all of whom are computer science graduate students and are familiar with the Chrome web-browser. A distribution of clicks tracked per website is shown in Fig 4 (right).[1]

Merging data across various browser sizes

Browser size can vary arbitrarily between users based on a variety of factors (operating system, screen resolution, user preference). In order to merge all the collected data into one useful training set, it is required that we normalize the browser sizes across all the data before we build our model. In order to account for this, we studied the resizing patterns of a few websites. We noticed that a few websites such as facebook.com and youtube.com have a fixed width layout (see Figure A-3(appendix)). This means that resizing them only affects the horizontal padding that surrounds the content.

[1] Source code for the extension available online so that participants can choose verify the privacy and security of their data and also point out potential bugs or vulnerabilities in the extension code.

Based on this resizing pattern, we can generalize our training data to fit any browser size by simply padding the content, and therefore also the click locations, to fit a particular browser size. Note that other websites studied such as Google and Reddit have *flowing* layouts, wherein the content resizes dynamically. Flowing layouts can also be modeled in a similar fashion, but we leave that for future work.

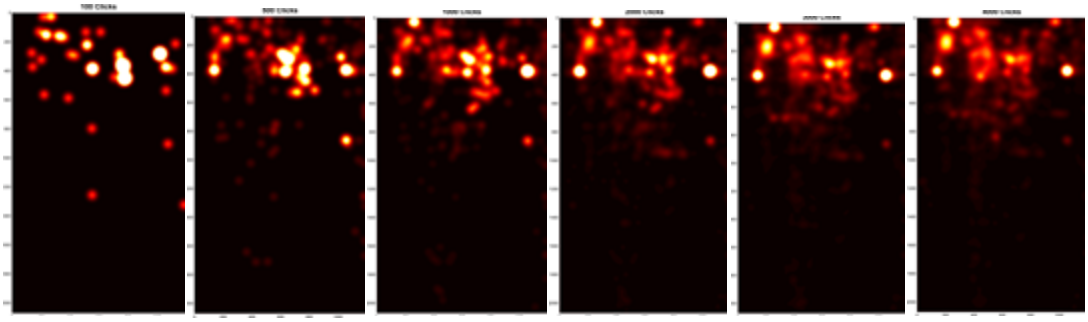

Figure 5: Probability maps generated with increasing number of test clicks. (left to right) 100, 500, 1000, 2000, 3000, 4000 clicks. Observe how the probability map starts to become stable after 2000 clicks.

Creating Probability Maps

The labeled data collected using the chrome extension was used to train the probabilistic classifier, which later was used for inferring what website a user is browsing given their test click data acquired using sniffing. The classifier generates a *Probability map* for each website in the corpus and an addition website called 'Others' (all non-corpus websites) which we later inference for mounting the attack.

The probability map is a 2D matrix(*Website'sXdimension* X *Website'sYdimention*) laid over the website layout that captures, given a website *W*, the likelihood that there will be click at pixel position *(x,y)* i.e. $P(C_{xy}|\underline{W})$. A probability map starts as a null 2D matrix P, such that every cell in P is 0. Each click in the training data is replaced with a Gaussian of unit mass and fixed spread. For example, if a click at *(x,y)* occurs, P matrix at *(x,y)* and its neighboring cells are updated with values from Gaussian of unit mass centered at (*x*,*y*). The premise behind choosing to represent the probability with a diffused Gaussian centered at *(x,y)* is to capture *user intent*, that is, though the user clicked the pixel *(x,y)*, they could have meant to click anywhere around it, and hence non-zero probability must be assigned to pixels around *(x,y)*.

Matrix P is finally normalized by the total number of clicks to represent probabilities $P(C_{xy}|\underline{W})$ and thus is appropriately named as *Probability map*.

For each click at x,y for a given website W: $P(C_{X,Y} \mid W) = P(C_{X,Y} \mid W) + Gaussian_{unitMass}(x, y)$

$$P_{Normalized}(C_{X,Y} \mid W) = P(C_{X,Y}) / TotalClicks(W) \quad (1)$$

A probability map is called mature when using additional training click points leads to small entropy change in the map. It is important to monitor the entropy change in the probability map as

it is generated since it dictates the number of minimum test clicks required for generating a robust map. For certain websites, this minimum number could be as high as several thousands of clicks, but since such clicks can be generated in a lab setting it does not pose a significant practical challenge. For example, an attacker could get several of his collaborators to collect data for various websites to build a mature probability maps, and then used these to plant an attack on any user. Figure 5 illustrates probability map generation.

Attack: Inferring Website Given a Click

As we attack and sniff data from the user using a wired mouse or a wireless mouse, we inference our probabilistic classifier that assigns probabilities that given a click it was done on a website in our corpus. Thus the problem could be formally defined as for each website in the corpus:

$$P(W \mid C_{x,y}) = ?$$

$$\textit{by Baye's Rule:} \quad P(W \mid C_{x,y}) = \frac{P(C_{x,y} \mid W)P(W)}{P(C_{x,y})} = \frac{P(C_{x,y} \mid W)P(W)}{\sum_{W_i} P(C_{x,y}, W_i)} = \frac{P(C_{x,y} \mid W)P(W)}{\sum_{W_i} P(C_{x,y} \mid W_i)P(W_i)} \tag{2}$$

where, $P(C_{x,y} \mid W)$ is the probability map and $P(W)$ is uniform over the websites that can generate *(x,y)*. We call a website can generate a click if *(x,y)* lies with its layout limits i.e ($x \leq$ *Website'sXdimension*) ($y \leq$ *Website'sYdimension*). If our classifier assigns a high probability (P_W) to one of the websites W and low probability to the rest we infer, with a confidence of P_W, that the user was presently browsing W.

Results

As mentioned earlier, we collected data from several users over a period of one month, but the results presented in the following sections are only from a single user (user 6, Figure 4 (right)) since that was the only user whose data had enough clicks (more than the minimum clicks to achieve probability map maturity) across several websites. For other users, there were enough

Figure 6: Confidence output for each click from a test set of 50 random clicks from Youtube.com. Observe that the confidence for match with Youtube's corpus (red line) is much higher. The higher and the more separated the red line from the rest, the better. If red line were consistently maximum, we would have a 100% prediction rate.

clicks for Facebook.com but very few for other websites, thus we do not compute comprehensive results for these additional users. This is important because if we use immature probability maps then either good or bad results do not hold any significance because they are heavily dependent on the whether the probability map is biased (yielding good results) or not (yielding poor results) towards the test data. Nonetheless, we make use of the Facebook.com data from *all users* to demonstrate that our approach can potentially work across multiple users and browser sizes.

Figure 6 shows results from user 6, when 50 randomly selected test clicks from Youtube.com were run through our classifier. The algorithm matches every click to every website in our corpus (4 websites, 1 others category) and outputs the confidence. The red line in the graph is the confidence that the given click is from Youtube.com, thus higher and more separated from other, the better. Observe that for certain clicks (15,19 etc) the classifier mis-classifies the click to be from Facebook.com (blue line). Table 1 summaries the results for user 6 for all websites.

Website	No. Of Clicks Correctly Classified (total 50 test clicks)	Accuracy (%)
Youtube.com	42	84
Facebook.com	27	54
Reddit.com	45	90
Google.com	40	80
Average		**77**

Table 1: Summary of correctly classified clicks for each website. 50 clicks were picked randomly for each website. Note that *each click* was classified, that is given one click point, the classifier predicted the output.

As evident from Table 1, the average accuracy when classifying one click at a time is a satisfactory 77%, but this can be greatly improved, if instead of one click at a time, *multiple clicks are used to make a prediction decision*. Figure 7 show the same scenario as Figure 6, but instead of a single click used to make decision, 10 clicks at a time are used.

This is analogous to having a moving average filter with a window of 10. Observe that the classifier predicts the correct website *100% of the time*. The selection of 10 clicks was decided after characterizing the system and observing the quality of predictions as the number of clicks used

Figure 7: Predicted output with 10-clicks at a time for 50 randomly selected clicks from Youtube.com. Note that the classifier predicts the correct website 100% of the time.

was raised from 1 through 10. We found that beyond 8 clicks, the results were above 95%, thus suggesting that with as few as 8 clicks, remarkable accuracy can be achieved.

Additional Experiments

A key caveat with the results presented above is that they are from a single user and a single browser, thus it raises the questions about how generalizable the findings are. To this end, we performed an experiment that aims to answer two questions: (1) can this approach work irrespective of the browser size?(i.e. training and classification on a different browser sizes) (2) can a model built from data of one person be used to predict clicks from *another* user?

We find that the answer to both the questions is *yes*. We constructed a dataset that merged the data for Facebook.com from all users, using different browser sizes, who participated in the study by using the merging technique described earlier. Our classifier was run on the merged data and we found with a window of 10-clicks we could predict the website correctly every time.

With the advent of multi-tabbed browser, a user could be switching between multiple websites and browsing multiple websites during the same session. We tested whether our approach works despite tab switching or in a more general case, simply navigating to a new website and found that our classifier can successfully detect transitions. Figure 8 shows the transition between Youtube.com to Google.com around click 25.

Discussion, Limitations and Opportunities

Using mouse clicks and movement data to infer one's browsing habits is a promising approach as evident from our preliminary results. Though we believe our analysis touched upon a breadth of issues in using mouse click and movement data for information leakage, it lacked depth in the number of users and hence showing wider generalizability. We see this as a key limitation of our work presented here, but nonetheless the results achieved were promising enough that we believe that information leakage through mouse data is quite possible. This was demonstrated specially by the experiment we performed where the model was learned offline from merged data of multiple users and was tested on a randomly chosen user. In addition, the same experiment proved our approach of making the data browser size invariant. As mentioned earlier in the paper, we

Figure 8: Note the transition detected around click 25 when the user switched tab from Youtube.com to Google.com

merged data from various users for only Facebook.com and Reddit.com, thus, other approaches for browser invariance need to be extended to other websites in our corpus.

To address the issue of generalizability, we intend to collect data from more users and for a longer period of time. Our estimate that a few weeks of data from 4-6 users would be sufficient was incorrect, since each website reaches critical maturation with different number of clicks. On an average we expect this number to be if the order of 4000.

Though we were able to extract the 5-byte hardware address that the Nordic RF IC in Logitech wireless mice use, we were unable to experimentally verify whether programming this address into the IC would allow us to capture the mouse data packets. We are confident that this is possible based on our understanding of the manufacture's supplied datasheets, but further experimentation is needed. We intend to procure the nRF24L01+ hardware development kit from Nordic Semiconductors to try this approach out, which is the last step in the reverse engineering chain to show that wireless mice are insecure and open to data leakage. Another direction for future investigation is to scrutinize mice that have built-in biometric fingerprint scanners and/or smart card readers.

The first defense against this sort of attack is to make it harder for an adversary to capture the mouse data packets –by securing physical access to the mouse in case of wired and by encrypting the communication between the wireless transceivers for wireless mice. Another possible defense is similar to that proposed by Harley et. al [10], wherein random mouse movement and click packets are transmitted to confuse the attacker. These random packets do not affect the user experience since they can be automatically ignored by the PC.

Conclusion

We have shown that the premise that mouse data reveals little information and is invaluable from a security perspective is untrue. Our experiments designed to infer private browsing habits of a user, entirely from mouse clicks and motion data show promise. Our preliminary results indicates that a probabilistic classifier could be trained with model data generated in a lab to compromise private browsing habits and potentially password from applications that use on-screen keyboard of user 'out in the wild'. Additionally, we presented our experience with reverse engineering a wireless mouse and showing that it is quite likely that an adversary would be able to gain this now-valuable mouse data remotely by simply sniffing RF packets. The implications of our findings are far reaching than just compromising private browsing patterns and can potentially be extended for tracking user behavior and compromising other sensitive information.

APPENDIX

References

[1] W. van Eck. Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk? Computers & Security, 4(4):269 – 286, 1985.

[2] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Advances in Cryptology (CRYPTO). 1996.

[3] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In Advances in Cryptology. 1999.

[4] D. X. Song, D. Wagner, and X. Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. In Proceedings of the 10th conference on USENIX Security Symposium, 2001.

[5] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Cryptographic Hardware and Embedded Systems (CHES). 2001.

[6] Cryptome. NACSIM 5000 TEMPEST Fundamentals. http://cryptome.org/jya/nacsim-5000/nacsim-5000.htm, April 2003.

[7] M. G. Kuhn. Electromagnetic Eavesdropping Risks of Flat-Panel Displays. In Privacy Enhancing Technologies, pages 88–107, 2004.

[8] M. G. Kuhn. Security Limits for Compromising Emanations. In Cryptographic Hardware and Embedded Systems (CHES), 2005.

[9] Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, 2010.

[10] C. Harley, D. Florencio How to login from an Internet café without worrying about keyloggers

[11] Martin Vuagnoux and Sylvain Pasini. 2009. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th conference on USENIX security symposium* (SSYM'09). USENIX Association, Berkeley, CA, USA, 1-16.

[12] Li Zhuang, Feng Zhou, and J. D. Tygar. 2009. Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur.* 13, 1, Article 3 (November 2009), 26 pages. DOI=10.1145/1609956.1609959 http://doi.acm.org/10.1145/1609956.1609959

[13] Ahmed Awad E. Ahmed and Issa Traore. 2007. A New Biometric Technology Based on Mouse Dynamics. *IEEE Trans. Dependable Secur. Comput.* 4, 3 (July 2007), 165-179. DOI=10.1109/TDSC.2007.70207 http://dx.doi.org/10.1109/TDSC.2007.70207

[14] Maja Pusara and Carla E. Brodley. 2004. User re-authentication via mouse movements. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security* (VizSEC/DMSEC '04). ACM, New York, NY, USA, 1-8. DOI=10.1145/1029208.1029210 http://doi.acm.org/10.1145/1029208.1029210

[15] Mouse Tracker Chrome Extension – Source http://www.cs.washington.edu/homes/aditya/files/misc/clicktracker/

[16] Content Scripts – Google Chrome Extension Developer Documentation
http://code.google.com/chrome/extensions/content_scripts.html

[17] HTML5 LocalStorage http://dev.w3.org/html5/webstorage/

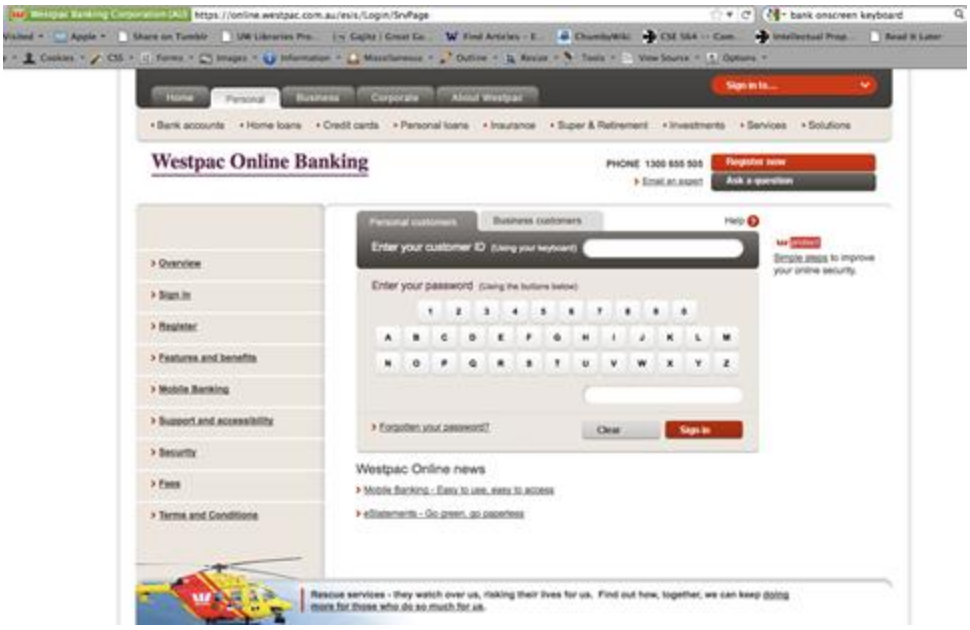

Figure A-1: A banking website making use of an on-screen keyboard as its primary password entry method to circumvent

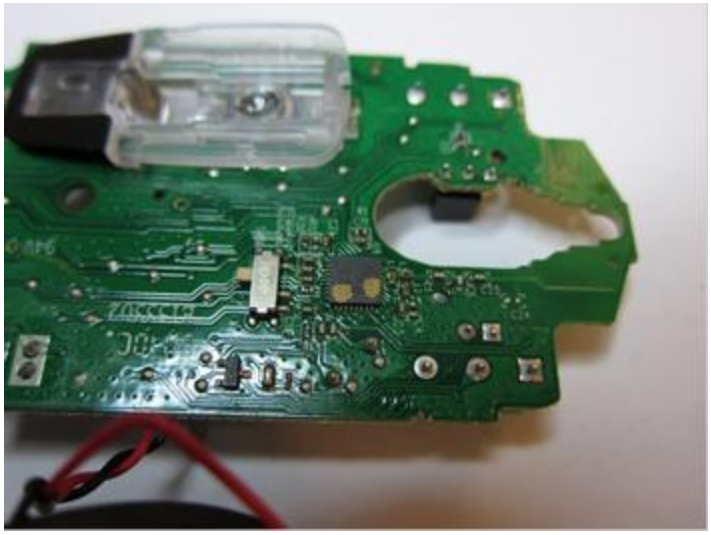

Figure A-2: Nordic nRF24L01 RF transceiver that is used on the Logitech mouse.

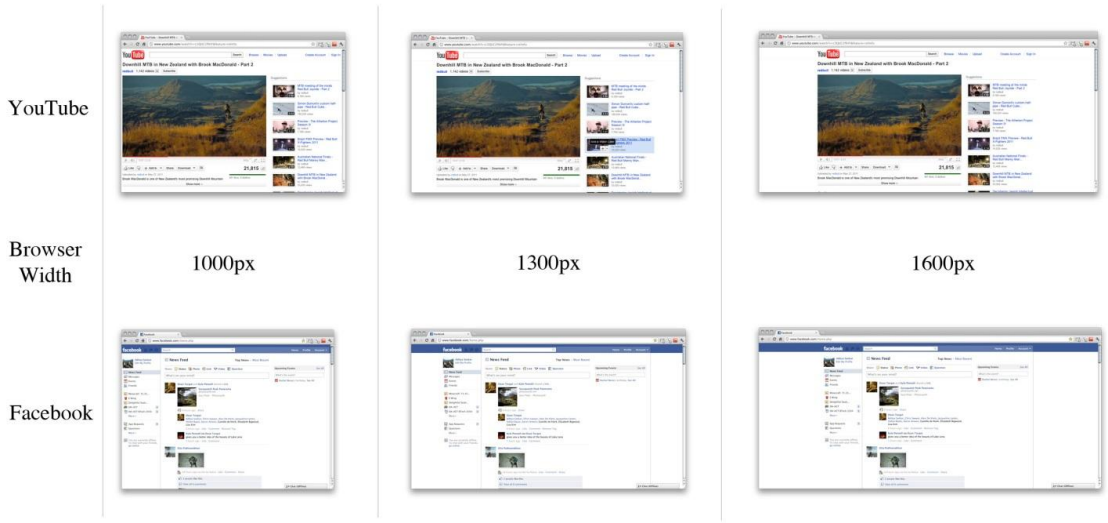

Figure A-3. Width of the content remains the same across browser sizes, but padding is varied.

Figure A-4: USRP, a software radio platform that was used along with GNURadio for capturing RF signals.