DEMO: How Privacy Leaks from Bluetooth Mouse?

Xian Pan
UMass Lowell
xpan@cs.uml.edu

Zhen Ling
Southeast University
zhenling@seu.edu.cn

Aniket Pingley
Intel Inc., USA
aspingley@gmail.com

Wei Yu
Towson University
wyu@towson.edu

Nan Zhang
George Washington University
nzhang10@gwu.edu

Xinwen Fu
UMass Lowell
xinwenfu@cs.uml.edu

ABSTRACT

Raw mouse movement data can be sniffed via off-the-shelf tools. In this demo, we show that such data, while seemingly harmless, may reveal extremely sensitive information such as passwords. Nonetheless, such a Bluetooth-mouse-sniffing attack can be challenging to perform mainly because of two reasons: (i) packet loss is common for Bluetooth traffic, and (ii) modern operating systems use complex mouse acceleration strategies, which make it extremely difficult, if not impossible, to reconstruct the precise on-screen cursor coordinates from raw mouse movements. To address those challenges, we have conducted an extensive and careful study, over multiple operating systems, on the reconstruction of mouse cursor trajectory from raw mouse data and the inference of privacy-sensitive information - e.g., user password - from the reconstructed trajectory. Our experimental data demonstrate the severity of privacy leaking from un-encrypted Bluetooth mouse. To the best of our knowledge, our work is the first to retrieve sensitive information from sniffed mouse raw data. Video links of successful replay attack for different target OS are given in Section 3.2.

Categories and Subject Descriptors

B.4.3 [**Hardware**]: Input/Output and Data Communications—*Interconnections*

Keywords

Bluetooth, privacy, mouse, sniffing, passwords

1. INTRODUCTION

Many mouse manufacturers and users believe that mouse data is not as sensitive as other HID devices such as keyboards. For example, Logitech made the following statement in a white paper published on Mar. 2, 2009 [4]: "Since the displacements of a mouse would not give any useful information to a hacker, the mouse reports are not encrypted." In this demo, we show mouse movement information can leak extremely sensitive information such as passwords.

The attack of Bluetooth communication begins with sniffing Bluetooth mouse communication. Various off-the-shelf tools are available to perform the Bluetooth sniffing. In particular, USRP2 (Universal Software Radio Peripheral 2) [1] is a software-defined radio device, which can be tuned to any Bluetooth frequency with a 2.48GHz daughterboard. To sniff Bluetooth communication in its entirety (i.e., across all frequency channels), four USRP2s are needed. Tools like Ubertooth [5, 6, 7] can be used to determine the MAC address of undiscoverable devices, which can in turn be fed into FTS4BT [2], a commercial product that is able to synchronize with victim Bluetooth devices, follow the Bluetooth frequency hopping sequence, and thereby sniff the entire communication session. With customized antennas, packet loss can be reduced with USRP and Ubertooth.

Once raw mouse movements are eavesdropped, we introduce a *trajectory-reconstruction technique*, which reconstructs the on-screen mouse cursor trajectory and the topology formed by the positions where mouse clicks occur i.e., the *clicking topology*. Such clicking topology may reveal sensitive information such as user behavior (e.g., applications which a user interacts with) and passwords. To the best of our knowledge, our work is the first to retrieve sensitive information from sniffed mouse raw data. Our major contributions can be summarized as follows:

- We examine mouse data semantics and investigate how mouse events are processed in an operating system, for the purpose of reconstructing an on-screen mouse cursor trajectory from sniffed raw mouse movements. For reconstructing on-screen mouse cursor trajectory, we can either predict the trajectory from raw mouse data or replay the raw data on the same type of computer as the target.
- By analyzing the reconstructed cursor trajectory, we can infer much information about a user's interaction with a computer. To demonstrate the severity of such privacy leakage, we study a soft-keyboard-based authentication scheme used by many security-critical websites [3, 8] and propose two approaches, basic approach and smart approach, to map a clicking topology to a password sequence entered by the user using the soft keyboard. Our experimental results demonstrate that privacy leaking from Bluetooth mouse can be severe and mouse data encryption should be mandatory. The basic approach has a success rate of more than 98% detecting the passwords while the smart approach has a success detection rate of more than 95%.

At the conference, we will demo Bluetooth sniffing and our replay attack for inferring passwords. In the rest of this writing, we will briefly introduce our attack strategy in Section 2, present preliminary results in Section 3 and conclude in Section 4. Because of space limit, we skip detailed algorithm presentation and analysis.

2. MOUSE CURSOR TRAJECTORY

In this section, we first discuss how to reconstruct the mouse trajectory on screen from sniffed raw mouse data and then discuss how to infer a clicked character sequence from clicking points in the reconstructed trajectory.

2.1 Trajectory Reconstruction

An OS may use an acceleration algorithm to calculate the cursor position based on the raw mouse movement data. We define two acceleration strategies based on whether the time of arriving packets is considered in the calculation of mouse acceleration on screen. *Lightweight Acceleration Strategy* does not consider the time of arriving packets, and it is used in Linux OS with Xserver version before 1.5. *Complex Acceleration Strategy* uses the time of arriving packets to determine the cursor acceleration, and it is adopted in Linux OS with Xserver version after 1.5, current Windows and Mac OS X.

To reconstruct an on-screen mouse cursor trajectory from sniffed raw mouse packets, we propose *prediction attack* and *replay attack*. Given raw Bluetooth mouse movement data, if an attacker knows the mouse acceleration algorithm of the victim OS, the attacker can predict the cursor trajectory on the target display of the victim system. However, the attacker may not know the mouse acceleration algorithm before-hand, particularly if the operating system is proprietary. It is not always trivial to reverse engineer those operating systems and derive the hidden mouse acceleration algorithm. Therefore, we also propose the *replay attack*.

The basic idea of the replay attack is to replay sniffed Bluetooth packets to an impersonating computer, which uses the same OS as the victim computer of interest and observes the motion of the cursor from the impersonating computer directly. For example, we can use Computer B to impersonate the victim Bluetooth mouse and connect to the impersonating Computer A. After setting up the connection, the fake mouse, i.e., Computer B will replay the sniffed Bluetooth mouse packet according to their timestamps. Therefore, the cursor movement on Computer A is the reconstructed mouse trajectory that we want. We have implemented the replay attack against Linux, Windows and MAC OS X and our fake mouse can emulate various mouse brands. To guarantee that replayed packet timing is accurate, we use the high resolution timer (*nanosleep* and real time clock) in Linux and implement the fake mouse.

The benefit of replay attack is that we do not need to understand the complex acceleration algorithm on the victim computer if we can impersonate the victim computer in terms of the operating system. We can know the type of operating system on the victim computer by various scanning software such as *nmap* and *Nessus*.

2.2 Inferring Character Sequence

Cursor clicking topology is formed by connecting all clicking points in the reconstructed trajectory. Recall that the reconstruction can be conducted by either prediction or replay attack from raw mouse movements. We consider the scenario of inferring character sequences from a reconstructed cursor clicking topology when a user is clicking on an on-screen soft keyboard, and will evaluate how well we can infer passwords based on the reconstructed clicking topology.

We now introduce the *basic approach* to infer the character sequence from a cursor clicking topology. The basic approach directly maps the clicking topology to an on-screen keyboard. Assume that we have derived the raw mouse data containing clicks on a software keyboard, we can derive the clicking topology. However, we do not know the exact starting point of the trajectory, and therefore cannot determine which keys are clicked in the topology. To derive all candidates (i.e., all possible character sequences corresponding to the trajectory), we move the cursor clicking topology from top left to bottom right in the area of the on-screen keyboard. Every time the topology moves, the clicking points produce a character sequence. We record all different character sequences. Therefore, a set of character sequences based on a cursor clicking topology can be derived. We denote the set of character sequences as *candidate character sequences*. The true character sequence must be one of the candidates if there is no packet loss and packet timing is correct. Nonetheless, the problem of this approach is that it may generate a large number of candidates.

In order to reduce the number of candidate character sequences, *a smart approach* is proposed to utilize the statistical information of the area people click on the on-screen keyboard. Intuitively, when hitting a key, the user tends to click in the middle region (rather than edge) of the area corresponding to the key - which we refer to as the *hot area* for the key. Since the size of keys on the software keyboard is different, to derive a normalized hot area, we first obtain more than 1000 clicking positions for different characters on the same on-screen keyboard, and then normalize the rectangle area of a key to a 1×1 square area. The hot area is the area that contains 99% of the clicked positions. After obtaining the hot area, we map a cursor clicking topology to an on-screen keyboard from top left to bottom right, and a character sequence will be considered as a candidate sequence only if all the characters' clicking positions are in the hot area. With the hot area, the number of candidate character sequences will sharply decrease. For example, our experiments show that the smart attack has the number of candidate passwords smaller than 4 on average. The benefit of the smart approach is that the uncertainty of the clicked character sequence is significantly reduced. This is what the attacker prefers.

3. PRELIMINARY RESULTS

We now present preliminary results on reconstructing cursor trajectory so that an attacker can compromise sensitive information of a user. To quantify the results, we infer password input via a software keyboard. We have conducted extensive experiments and attacks were successful under Linux, Windows and Mac OS X. Because of the space limitation, we show selected results under Linux to illustrate the feasibility. We randomly generated 100 passwords of 8 characters length (including uppercase letters, lowercase letters, and numbers), and used a Bluetooth mouse (Logitech MX 5500) to click on a on-screen virtual keyboard, *xvkbd*, in order to input those passwords at a computer installed with openSUSE 11.1, which uses the lightweight mouse acceleration strategy. We also evaluated the success rate of inferring passwords with the complex acceleration on Fedora Core 13 and the replay attack. Success rate is defined as the per-

Table 1: Password Detection Rate for Lightweight and Complex Acceleration Algorithms

	Basic Inferring		Smart Inferring	
	small keyboard	large keyboard	small keyboard	large keyboard
Lightweight Acceleration	100%	100%	99%	99%
Complex Acceleration	99%	98%	98%	95%

centage of real passwords that are included in the set of candidate passwords generated by our prediction algorithm.

3.1 Prediction Attack

The packet arrival timing affects the attack accuracy on reconstructing the mouse cursor trajectory on screen for operating systems using the complex acceleration strategy. We conducted extensive experiments on Fedora Core 13, which uses complex acceleration strategy, to investigate how much packet timing can affect the result of inferring passwords. To reduce the impact from timing, we should use the data starting at the time when the first click of passwords happens and this reduces the prediction error.

Table 1 compares the results of inferring passwords for lightweight and complex acceleration strategies. A small virtual keyboard *xvkbd* has a size of 449 × 149. The large version of the keyboard has a size of 896 × 254. We have observed in the experiments for the large size keyboard with the basic inferring method that 98% of password clicking processes have a topology deviation in the range [0,25] pixels in both X and Y axes. In only one case, the deviation is 52 pixels on the X direction and 9 pixels in the Y direction. However, the large deviation does not always lead to a failure of password inference, because the predicted clicking topology may be still in the characters' areas on the software keyboard. We have observed similar results in experiments on the small size keyboard. Table 1 shows that passwords can be successfully derived for complex acceleration strategies by using approaches we developed. One reason for the high success rate is that mouse movement for entering passwords (clicking an on-screen keyboard) is different from mouse movement in other situations. Each character on the on-screen keyboard corresponds to a small area. Users always take caution when inputting passwords and will not move the mouse too fast to miss a key. Hence, mouse movement is slow when users input their passwords on an on-screen keyboard. This slow movement reduces the impact of packet timing on mouse acceleration and favors reconstructing a correct clicking topology.

3.2 Replay Attack

We now show the results of replay attack. The attack computer is installed with Ubuntu 8.04 and the target computer is installed with Fedora Core 13. Our experiments show that because of more impact from replayed Bluetooth packet timing, the performance of the replay attack is not as good as prediction attack. Bluetooth packet timing is seriously distorted during the replay. However, a detection rate of 69% is still achieved when the basic inferring is used. The detection rate for smart inferring is 31%. Therefore, basic inferring is recommended for the replay attack.

Here are videos of successful replay attack for different target OS: Fedora Core 13 `http://www.youtube.com/watch?v=qnjqgCCTVTk` Windows 7 default installation `http://www.youtube.com/watch?v=FVJK_m3UPj0` Mac OSX 10.6.5 `http://www.youtube.com/watch?v=iFJoHBiYDWg`. These videos show the whole replay attack process, and do not include the sniffing process. The attack computer is always installed with Ubuntu 8.04. It replays the sniffed raw mouse data of a real login session of WESTPAC online bank [8] on the same OS as the victim computer. From these videos, we can see that a victim's password can be disclosed.

4. CONCLUSION

In this demo, we demonstrate privacy leakage from unencrypted Bluetooth mouse traffic. By reviewing the process of establishing Bluetooth connections, we demonstrated how one can sniff Bluetooth traffic via multiple sniffers or a single sniffer. We examined the Bluetooth mouse packet semantics and discussed how raw mouse movements could be mapped to on-screen cursor trajectories when lightweight or complex cursor acceleration strategies are being used. Finally, we conducted an extensive evaluation of an application of Bluetooth mouse sniffing - the inference of passwords that a user enters via an on-screen soft keyboard. Specifically, we proposed two approaches for password inference: a basic approach to enumerate all candidate passwords from the clicking topology and a smart approach that utilizes the statistical distribution of human clicking patterns to reduce the number of candidate passwords from a clicking topology. Our experimental results demonstrated the seriousness of privacy leakage from unencrypted Bluetooth mouse. Thus, we recommend the encryption of Bluetooth mouse traffic to be mandatory.

5. REFERENCES

[1] M. Ettus. Usrp produce. `http://www.ettus.com/`, 2012.

[2] Frontline Test Equipment, Inc. . Fts4bt bluetooth protocol analyzer and packet sniffer. `http://www.fte.com/products/fts4bt.aspx`, 2012.

[3] HSBC. Security key demo. `http://www.banking.us.hsbc.com/personal/demo/cam/cam_demo.htm`, 2012.

[4] Logitech. Logitech advanced 2.4 ghz technology, revision 1.1h. `http://www.logitech.com/images/pdf/roem/Logitech_Adv_24_Ghz_Whitepaper_BPG2009.pdf`, March 2009.

[5] M. Ossmann. Project ubertooth. `http://ubertooth.sourceforge.net`, 2012.

[6] D. Spill. Final report: Implementation of the bluetooth stack for software defined radio, with a view to sniffing and injecting packets. `www.cs.ucl.ac.uk/staff/a.bittau/dom.pdf`, 2007.

[7] D. Spill and A. Bittau. Bluesniff: Eve meets alice and bluetooth. In *Proceedings of USENIX WOOT*, 2007.

[8] WESTPAC. Westpac online banking. `http://www.westpac.com.au/personal-banking/westpac-online/`, 2012.