Touch Interface and Keylogging Malware

Samuel Moses
Brigham Young University
Cybersecurity Research Lab
Provo, Utah

Allie Larson
Brigham Young University
Cybersecurity Research Lab
Provo, Utah

Jon Mercado
Brigham Young University
Cybersecurity Research Lab
Provo, Utah

Dale Rowe
Brigham Young University
Cybersecurity Research Lab
Provo, Utah

ABSTRACT-*Software keyloggers have been used to spy on computer users to track activity or gather sensitive information for decades. Their primary focus has been to capture keystroke data from physical keyboards. However, since the release of Microsoft Windows 8 in 2012 touchscreen personal computers have become much more prevalent, introducing the use of on-screen keyboards which afford users an alternative keystroke input method.*

Smart cities are designed to enhance and improve the quality of life of city populations while reducing cost and resource consumption. As new technology is developed to create safe, renewable, and sustainable environments, we introduce additional risk that mission critical data and access credentials may be stolen via malicious keyloggers. In turn, cyber-attacks targeting critical infrastructure using this data could result in widespread catastrophic systems failure. In order to protect society in the age of smart-cities it is vital that security implications are considered as this technology is implemented.

In this paper we investigate the capabilities of keyloggers to capture keystrokes from an on-screen (virtual) keyboard and demonstrate that different keyloggers respond very differently to on-screen keyboard input. We suggest a number of future studies that could be performed to further understand the security implications presented by on-screen keyboards to smart cities as they relate to keyloggers.

Keywords-Malware, Software Security, Privacy

I. INTRODUCTION

Human-Computer Interfacing is an ever evolving discipline in which numerous new and innovative technologies have recently become mainstream in society. Touchscreens, speech recognition, gesture control and swipe passwords have found their way into the hands of consumers within the last decade. However, it is often the case that our ability to develop this technology precedes our understanding of the security risks involved.

One of these security risks is keyloggers. Keyloggers have long been a serious threat to computer systems throughout the world. In 2013, stolen passwords played a role in 48% of total data breaches caused by hackers. In many cases keyloggers were the medium by which these passwords were obtained [1]. In Verizon's 2014 Data Breach Investigations Report, keylogger malware was one of the top 10 threats in Point-of-Sale (POS) intrusions and crimeware, and was involved in 2% and 13% of attacks respectively. Keyloggers are also heavily used in cyber-espionage and were involved in 38% of related data breaches [2]. In Verizon's 2015 Data Breach Investigations Report, the use of keyloggers was shown to decline, but are still observed in about 5% of breaches. In addition, POS attacks continue to evolve through successful phishing campaigns and network penetrations involving keyloggers which are still used in at least 9.5% of successful attacks [3].

In this paper, we will focus our attention on the particular threat that keyloggers will pose to the terminals which provide direct access to the groundwork that smart cities will be built upon. This threat gained significant momentum with the recent inclusion of an on-screen keyboard into Microsoft's Windows 8 operating system. As the Windows operating system is currently used by billions of users and in a number of critical infrastructure applications, the principles and ideas presented through this research can be theoretically applied to future systems involved in the development of smart cities. Specifically, we will look at how current keylogging applications interact with the on-screen keyboard and

978-1-4673-9511-4/15/$31.00 ©2015 IEEE

the potential for future malware development in this area.

II. KEYLOGGERS

Keyloggers are traditionally used to capture and track a user's keyboard input, often to attempt to steal the user's private information or login credentials. These keyloggers were originally simple surveillance tools, but advanced keyloggers have become more sophisticated and can track complex functions like copy and paste operations. They have also been found to be able to conceal themselves, gather system data, and communicate and report to external sources. Hardware keyloggers and software keyloggers exist that can be used for this purpose.

Hardware keyloggers are small devices that are physically attached to a machine to locally store a user's keystrokes. They are normally installed between the keyboard and its I/O port so that it can accurately interrupt key-press data and capture it before forwarding it on to the motherboard. In order to install a hardware keylogger, the attacker must have physical access to the machine.

Software keyloggers are applications that are installed directly into the operating system of the target computer and will run in the background while logging user's keystrokes. There are three main methods for developing software keylogging systems: The Keyboard State Table Method, the Windows Keyboard Hook Method, and the Low-level Kernel-mode method [4].

Malware built using the Keyboard State Table method uses the active application's windows interface table to access the status of 256 virtual keys, which correspond to physical keys on the keyboard. This table is normally used by applications to determine the use of key combinations (e.g. Ctrl+Shift). This variety of keylogger reveals keystroke information by attaching its thread into the window's thread message loop and recording the information before it is sent to the Keyboard State Table and passed on to the Window's procedure [4].

The Windows Keyboard Hook method utilizes a method known as "hooking" to log keystrokes. Hooking intercepts the normal process of a running subroutine and calls a malicious function to gather or alter information before continuing on with the initial subroutine. This hooking method is unique to Windows Operating Systems, and can be implemented at any level of the operating system. In a Windows operating system, keystroke events from the user are flagged through a message mechanism which passes the keyboard data to the windows procedure. This mechanism is hooked and can provide an attacker with the ability to record keystrokes and even intercept and alter them before they reach the intended windows procedure [5].

Some keyloggers are designed to run with kernel level privileges. Keyloggers at this level are advanced spyware rootkits (rootware). These applications are well hidden and hook into vital system routines to collect and transport the keystrokes without the user being aware [5]. This type of keylogger is more difficult to implement but is also harder to detect. Kernel level keyloggers require administrator privileges to plant on the target machine, but once accomplished a keyboard filter driver is installed which allows the keylogger to capture keystrokes even before the operating system receives them [4].

Keylogger programmers are constantly trying to come up with new and innovative ways to capture information from their victims. With antivirus and antimalware companies working to detect and block keyloggers, keylogger programmers have to come up with new and stealthier approaches to attack the problem. For this reason it is imperative to understand how today's keyloggers are interacting with virtual keyboards and discover if they are capable of capturing touchscreen keyboard input.

III. PAST RESEARCH

Little research has been performed to identify the response of current keyloggers to the keystrokes of on-screen keyboards. There has, however, been some research done to develop keyloggers designed to work with touchscreen computers (a.k.a. touchloggers), but plans for this research is generally intended for mobile operating systems. For example, some work done by HAO Chen utilizes smartphone jiggle during key-presses to calculate which key of the virtual keyboard the user has tapped. His app has shown to correctly track over 70% of keystrokes [6].

The University of the Aegean worked on developing a touchlogger that can be a reliable means of profiling the user of a mobile device. To achieve their goal there were two fundamental steps: First, the touchlogger needed to gain administrative permissions to be able to hook and override operating

system subroutines which were responsible for the detection and management of touch events. Second, it needed to run in the background of the operating system and constantly track and collect the user's touch behavior. The basic steps involved were very similar to traditional keyloggers, but in order to track specific keystrokes within the virtual keyboard they needed to identify the touch events that occurred inside the soft keyboard area and translate every touch to the corresponding key which proved difficult [7]. Again, their work was very mobile focused and specific to Apple Inc.'s iOS mobile platform.

Another study was completed in 2008 on how attackers were using keyloggers to enter into digital crime and created an underground economy. In 2008, keyloggers utilizing dropzones were a new emerging threat. Dropzones are publically writeable directories on a server that serve as an exchange point for keylogger's stolen data[8].

A basic understanding of the attack is straightforward. The attacker first infects victims with keylogging malware, which then secretly logs credentials and the victim's online service information including financial data and healthcare information. After the malware sends the data to the dropzone the attacker can then access the dropzone and utilize the data for any purpose. Typically this data is sold via illegal underground networks. Researchers used honeypots and spam traps to analyze and collect the different keylogger techniques actively being used. Analyzing these keyloggers, the researchers were able to extract the location of the dropzone, access them, and harvest the keylogger data just like the attackers. They observed these attacks during a seven month period and the results were impressive [8].

Over the seven month period, the researchers found a total of 33GB of keylogger data from more than 70 unique dropzones. This data was stolen from more than 173,000 compromised machines and worth several million dollars. In their analysis, they found 10,775 unique bank account credentials. Looking at the results from the dropzone, they were able to see that 25 victims had more than $130,000 in their checking accounts and on average $5,225 in savings. This resulted in the attackers having access to millions of dollars through those accounts [8].

Within the dropzone, 5,682 valid credit card numbers were found which could have caused a potential loss of over one mission dollars. They also recovered 149,458 login credentials for Yahoo, Google, Windows Live and AOL and 78,359 stolen credentials for social media sites [8].

A conservative estimate based on a study by Symantec suggests an attacker can earn several hundred up to thousands of dollars per day from keylogger based attacks [8]. PWC, in The Global State of Information Security Survey 2015, stated that the black market for stolen data is growing in size and complexity [9]. "A complete identity-theft kit containing comprehensive health insurance credentials can be worth hundreds of dollars or even $1,000 each on the black market, and health insurance credentials alone can fetch $20 each; stolen payment cards, by comparison, typically are sold for $1 each" [9]. The market is growing, and keyloggers are still an actively used tool for hackers.

IV. METHODOLOGY

In order to discover the ability of keyloggers to capture touch-screen keystroke data, we selected a sample of five software keyloggers with which we performed our tests. In order to observe a variety of applications, we selected one commercial keylogger, two freeware keyloggers, one JavaScript browser-based keylogger, and a malware keylogger. We also tested a hardware keylogger to confirm our assumption that hardware keyloggers are unable to capture virtual keyboard data.

The Test Machine

The computer that was used to test all keyloggers was a Dell Inspiron 1545 laptop with attached HP KU-0316 USB Keyboard and Dell ST2220T touch screen monitor. Microsoft Windows 8.1 64-bit was the installed operating system. The application through which all tests were performed was Microsoft WordPad with the exception of the Metasploit JavaScript keylogger which utilized Internet Explorer to carry out tests.

The Keyloggers

Free Keylogger version 3.95 – Freeware available from multiple application hosting web sites

Actual Keylogger version 3.2 – Commercial software available from actualkeylogger.com

- We utilized the trial version, which limits keylogging session time to forty minutes.

Metasploit Meterpreter Keylogger – A keylogger built into the meterpreter malware payload available through Metasploit

- We used Metasploit version 4.11.1-2015032401 on a remote Kali Linux 1.1.0 machine to create the meterpreter payload. Metasploit was then set up to listen for incoming meterpreter connections and the malicious payload was installed on the test machine via USB flash drive. Once meterpreter was installed it connected to the remote Metasploit server. We then started the built-in keylogger and were able to view the keystrokes made on the test machine on the remote Kali Linux server.

Metasploit JavaScript Keylogger – A browser-based keylogger designed to capture keystrokes of users browsing the Internet

- This was a Metasploit module written by Marcus Carey that creates a website with a malicious script that captures the keystrokes of anyone visiting the website. Again, the Kali Linux remote host was setup as the server to host this malicious website which the test machine was to navigate to in order to perform the tests. The web browser used in our test was Internet Explorer 11.

Spyrix Keylogger Free version 7.0 – A freeware keylogger with a paid for version available from spyrix.com

KeyGrabber Wi-Fi Premium – A hardware keylogger that stores keystrokes on internal flash memory local to the keylogger itself

Testing Methodology

We used a United States English keyboard and tested all keys on the main keypad that had corresponding keys on the Windows virtual keyboard. The keystroke pattern we used in our test is as follows, each line ending with a carriage return:

1234567890-=
~!@#$%^&*()_+
qwertyuiop[]\
QWERTYUIOP{}|
asdfghjkl;'
ASDFGHJKL:"
zxcvbnm,./
ZXCVBNM<>?
<ctrl>
The quick brown fox jumped over the lazy dog
The quick red *<backspace>* *<backspace>* *<backspace>*

Below is the guideline that we followed for testing each keylogger:

- Turn off Windows Defender. This is required for the installation of Actual Keylogger and the Meterpreter payload.
- Install the keylogger
- Perform the test keys on the physical keyboard
- Save the results (screenshots and log files depending on how keylogger results were recorded)
- Perform the test keys on the touch screen keyboard
- Save the results (screenshots and log files depending on how keylogger results were recorded)
- Reset the Windows 8.1 installation to factory defaults
- Repeat for each keylogger being tested

V. FINDINGS

All keyloggers were able to successfully capture all keystrokes performed on the physical keyboard. We will focus the subject of our findings on those keystrokes captured by each keylogger through the virtual keyboard.

Complete Virtual Keyboard Capture

The only Windows keylogger application that was able to capture all keystrokes from the virtual keyboard was "Actual Key Logger". The JavaScript keylogger was also able to successfully log all keystrokes though the Internet Explorer web browser.

Partial Virtual Keyboard Capture

The Meterpreter keylogger, Free Keylogger and Spyrix keylogger were able to register the carriage return and backspace virtual keystrokes but were unable to log any others from our test.

No Virtual Keyboard Capture

The only keylogger that did not register any virtual keystrokes was the KeyGrabber physical keylogger.

Our software keyloggers only fell into two categories: those that could log all virtual keystrokes and those that only registered the backspace and carriage return characters. This suggests that there is a marked difference in how keyloggers respond to

virtual keyboard use within Windows based on the technique used to monitor keystrokes. We hypothesize that this difference might be a result of where the keylogger is capturing keyboard data. Return and backspace perform additional functions apart from ASCII translation which may have resulted in keylogging software intercepting these keyboard interrupts differently than in the case of typical ASCII characters. Since the results vary between keyloggers, we redirect our attention to what can be done in the future to further understand these differences and their implications

Keylogger	100% Keystroke Coverage	Entr + Bksp Only	0% Keystroke Coverage
Actual Keylogger	x		
Metasploit JavaScript Keylogger	x		
Free Keylogger		x	
Meterpreter Keylogger		x	
Spyrix Keylogger		x	
KeyGrabber Physical Keylogger			x

Table 1: The percentage of keystrokes that each keylogger was able to capture from the Windows on-screen keyboard

VI. RELEVANCE

Understanding the potential impact of keyloggers on upcoming technologies is important to be aware of while developing smart cities. Technology can enhance the quality of life and performance within a city, but at what cost? We need to analyze the arsenal of tools available to malicious actors and propose suitable countermeasures as we design new city-wide networks. Otherwise we will be unprepared to mitigate the risks to people's privacy and security that are created when developing smart cities. Indeed we have already witnessed a severe lack of security consideration within the critical infrastructure of today's urban centers. Recently the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) of the Department of Homeland Security released alerts ICS-ALERT-11-343-01 and ICS-ALERT-11-343-01A after having responded to a number of reports of vulnerable supervisory control and data acquisition (SCADA) systems that are open to attack and are at high risk. These are the systems responsible for controlling high-profile critical infrastructure including power grid, water, and communications networks. Attacks on these systems have the potential to result in catastrophic consequences including loss of life and billions of dollars in recovery costs [10]. Specifically, these alerts speak to SCADA systems that are internet accessible, which will likely become the backbone of many new systems used to create smart cities. Our research into how current keyloggers interact with touch-screens is just a first step in one area toward understanding the security implications that arise with the emergence of connected city infrastructure.

VII. STUDY LIMITATIONS

This small study has several limiting factors. We limited the number of software keyloggers to five, which allowed us to witness the differences in how they can respond to a virtual keyboard environment. From this we can suggest from our testing that using a virtual keyboard will decrease the likelihood that sensitive information will be obtained from a system where a keylogger is installed. However, this limited sample could not tell us the likelihood that virtual keyboards will circumvent keyloggers if in use in smart city applications when deployed via malware, nor can we determine the degree to which using a virtual keyboard might be safer than using a physical one. Among the five keyloggers selected, there were also a variety of methods for installation and logging used that caused the testing process to be inconsistent.

With the exception of the Metasploit JavaScript keylogger all tests were performed using the Microsoft WordPad application. This was done so that we could easily test a large number of keystrokes. However, this testing does not indicate whether these keyloggers would perform the same way in other applications.

Since we are unaware of the specific mechanisms used in each of the keyloggers we tested we do not yet understand the reasons that they performed differently, nor can we draw any conclusions about how different programming techniques affect virtual keylogging in real-world scenarios.

VIII. FUTURE WORK

There are many areas in which this research can be expanded. Given a larger sample of keyloggers, a more complete analysis of keylogger response can be performed in which keyloggers are grouped by type (i.e. malware, commercial, x86, JavaScript etc.) and/or keylogging mechanism. This would provide a deeper understanding of the reasons keyloggers differ in their ability to log keystrokes from on-screen keyboards. The research could also be expanded to include other operating systems that may be involved in smart city infrastructure if future touch based applications become available.

Keyloggers could also be tested in a specific user scenario. For example, in a situation where login credentials are used to access a touch-based SCADA control terminal we can evaluate the security differences between virtual and physical keyboards. It would also be useful to discover which keylogger architectures are most prevalently flagged and quarantined by anti-virus software and determine if a correlation exists between keyloggers that are caught by anti-virus software and those that are able to log virtual keystrokes.

Additional research can also be conducted to uncover techniques that are effective in capturing virtual keystrokes and understand the feasibility of implementing these techniques in future touchlogging software. This has an inherent risk to it since it can be used maliciously, but attackers will be evolving their malware to fit this use case. If we can stay ahead of the curve and understand the risks better, we can be prepared to defend against the threat.

IX. CONCLUSION

We have demonstrated that currently available keyloggers respond differently using the Microsoft Windows on-screen keyboard verses a physical keyboard. We have presented an overview of the different methods that can be used to create keyloggers and suggested that these architectural differences directly affect the keylogger's ability to successfully log keystrokes made on a virtual keyboard. We have also presented a number of future studies that could be performed to further understand the security implications involved when dealing with on-screen keyboards and keyloggers.

As the smart cities of the future are planned and infrastructure developed it is important to be aware of what security risks may be created during the process. As the threat of keylogging evolves to include additional input methods, understanding their functionality will be key to developing effective defensive capabilities to mitigate the risk. If we are willing to more fully address the security concerns inherent in the advancement of smart city technology in its early stages of development we stand a chance at maintaining a secure and reliable infrastructure now and far into the future.

VII. REFERENCES

[1] Dark-Reading, "'The 8 Most Common Causes of Data Breaches And How You Can Prevent Them,'" *Dark Reading*, pp. 2013–2015, 2015.

[2] Verizon Business, "2014 Data Breach Investigations Report," *Verizon Bus. J.*, vol. 2014, no. 1, pp. 1–60, 2014.

[3] V. E. Solutions, "2015 DBIR Contributors," *Verizon Bus. J.*, 2015.

[4] S. Sagiroglu and G. Canbek, "Keyloggers: Increasing threats to computer security and privacy," *IEEE Technol. Soc. Mag.*, vol. 28, no. 3, pp. 10–17, 2009.

[5] C. a Wood and R. K. Raj, "Keyloggers in Cybersecurity Education," 2015.

[6] J. Aron, "Smartphone jiggles reveal your private data," *New Sci.*, no. August 2011, p. 21128255, 2015.

[7] D. Damopoulos, G. Kambourakis, and S. Gritzalis, "From keyloggers to touchloggers: Take the rough with the smooth," *Comput. Secur.*, vol. 32, pp. 102–114, 2013.

[8] T. Holz, M. Engelberth, and F. Freiling, "Learning more about the underground economy: A case-study of keyloggers and dropzones," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5789 LNCS, pp. 1–18.

[9] PWC, "Managing cyber risks in an interconnected world," no. September 2014, 2015.

[10] ICS-CERT, "Alert (ICS-ALERT-11-343-01A)," 2014.