

CSE 825: Project Proposal

USB Keyboard and Mouse Spoofing

Vince Fasburg, Bonnie Reiff, and Josh Thomas
Dept of Computer Science and Engineering, Michigan State University
East Lansing, MI, USA
{Vincent.Fasburg, Bonnie.Reiff, Josh.Thomas}@ge.com

1. Problem Definition

2. Approach

The team is proposing a two part project which includes exploiting a few weaknesses in a computer system by creating a suite of attacks, then offering a solution to those weaknesses by developing a driver to prevent future attacks. This section will describe the approach for each part of the project as well as the equipment the team will need.

For the first part of the project, the team will program a Teensy microcontroller, shown in Figure 1, to run a suite of attacks on a Linux operating system. The main idea would be to emulate a USB keyboard and mouse with the Teensy so that the computers Linux operating system will recognize the Teensy as a safe device. Once the operating system believes that the Teensy is a safe device, the program can send keyboard or mouse commands to the operating system. The operating system would not notice anything different about these commands as compared to the commands coming from an actual keyboard or mouse. A broad range of attacks will be created including some which would require admin privileges.



Figure 1. Teensy Microcontroller

This is a potentially very dangerous attack because once the program has the ability to use the keyboard and

mouse, a Linux terminal can be opened to do almost anything to the system including “rm rf *” which would wipe all of the files on the drive. The program could easily run without the user being aware since a terminal window could be opened up and immediately minimized while the program is still running. After only a few seconds, the program could be finished running and have the terminal window closed all before the user realizes anything is wrong.

For the second part of the project, the team will work on preventing the attacks that were created in the first part by developing a linux driver to authenticate the USB device before any attached keyboard or mouse is allowed to be used normally. Some preliminary ideas include using a series of Turing tests to determine if the input device is being used by a computer or a human. Essentially a Turing test is a test of a machines ability to exhibit human behavior indistinguishable of that of a human [6].

To test a keyboard device, a random text captcha, as shown in Figure 2 could be used. The program installed on the Teensy would not know what the random Captcha text is going to be, so a human user would be required to type in the letters by hand before the keyboard is allowed to be used. In addition, a mouse could be verified by requiring a user to follow a randomly generated pattern on the screen or point and click to something in random on the screen to verify that a human is the one sending the commands as a mouse. If any of these tests fail, no other input is allowed from the device until the captcha or pattern is complete.



Figure 2. Captcha Example

3. Previous Work

The Teensy microcontroller has been used for many purposes, including mounting many different HID based attacks on various systems that accept input from USB human interface devices. An excellent resource that demonstrates the need for HID authentication is the website of Samy Kamkar [2], in which the author shows an attack he calls “USBDriveby”. The USBDriveby attack uses a Teensy microcontroller, emulating a keyboard and a mouse simultaneously, to create a permanent connection to a remote server that is controlled by the attacker. This connection will be re-established regularly so that the attacker can issue commands to the machine at any point in the future, including after the microcontroller has been disconnected from the victim's machine.

This example of an emulated HID based attack serves to show the severity with which a system can be compromised when a malicious USB device is connected for even a short time, and without requiring an administrator password. Although the USBDriveby attack was created for the Apple OSX operating system, similar attacks may be possible on any other system that accepts USB inputs. The list of potential attacks is very large, including attacks with various effects to the victim, and ranging from very simple to very complex in implementation.

Although some attempts have been made to develop a mechanism for the authentication of USB devices, much less attention has been paid to the authentication of USB human interface devices. In a paper [5] proposing an extension to a USB driver to authenticate USB connections, the proposed 2-way authentication mechanism specifically excludes “non-programmable devices” such as a USB keyboard. This seems sensible, but leaves a system vulnerable to programmed attacks hiding inside a “non-programmable device”.

Among the literature dealing with USB device security, some papers such as “Plug & Prey: Malicious USB Devices” come closer to a solution regarding malicious HID's. The paper describes how, on Windows, you can use registry changes to prevent USB devices from being installed. However, this would prevent legitimate HID's from being installed also, unless a whitelist of vendor IDs and product IDs is used to allow certain devices. However, these properties can be spoofed, so the Teensy device may appear to be a whitelisted device.

The final suggestion in this paper is to apply settings to allow administrators to override the policy preventing the installation of a HID. This technique would run into problems if the device in question were the first HID to be connected to the machine, as the user would have no way to allow the installation. Furthermore, it requires administrative privileges to plug in a keyboard or mouse. The mecha-

nism to disable installing USB devices is very different for Linux system, but the results and limitations are roughly the same.

A novel approach to USB authentication specifically for HID's was proposed by Barbhuiya et al. [1] which relies on either continuous or periodic authentication based on keystroke dynamics, including metrics such as the speed of typing, and the relative times of presses and releases of the keys. This method uses a stored profile of typing behaviors from the authentic user to determine whether the keystroke dynamics exhibited by the inputs are a match to the known patterns. If the keystroke dynamics are determined not to match the stored profile, the input will not be processed.

The keyboard dynamics approach to preventing emulated HID attacks would likely succeed most of the time. However, there are some significant drawbacks in this type of technique from a usability standpoint, stemming from both false positive and false negative authentication. Consider that an attack from a malicious device may be extremely short, and thus more likely to be unintentionally allowed through the authentication (consider the command “rm rf *”). If the malicious device were programmed to enter input in a human seeming way (perhaps emulating keyboard dynamics of an average human), there is a significant chance that small amounts of inputs would be accepted [4]. Furthermore, a user using a computer equipped with keystroke dynamics based authentication may reject input from a legitimate user if keystroke patterns change do to any variation in the environment or the user. Lastly, the principles of keyboard dynamics cannot easily be generalized to include authentication for a mouse as well as a keyboard.

Finally, at the 2014 Las Vegas Black Hat Security Conference, Karsten Nohl and Jakob Lell presented several USB based attacks that they collectively referred to as BadUSB. Among the attacks described is the malicious emulation of a HID. This section of the presentation describes how an emulated keyboard could be used to steal the administrative password on a linux system, then allowing the device to perform other actions using the sudo command. The only defenses proposed in this presentation that would be suitable against an emulated keyboard or mouse were block USB devices entirely, or by using a blacklist or whitelist of device types. As stated previously, such defenses can easily be overcome by spoofing a legitimate device ID.

References

- [1] F. Barbhuiya, T. Saikia, and S. Nandi. An anomaly based approach for hid attack detection using keystroke dynamics. In Proceedings of the 4th International Conference on Cyberspace Safety and Security, 2012.
- [2] S. Kamkar. Usbdriveby. <http://samy.pl/usbdriveby/>.
- [3] K. Nohl and J. Lell. Badusb - on accessories that turn evil. Presented at Black Hat, Las Vegas, August 2014.

- [4] M. Shahzad. ... Presented to ECS-825, February 2015.
- [5] Z. Wang and A. Stavrou. Attestation & authentication for usb communications. In Proceedings of the IEEE Sixth International Conference on Software Security and Reliability Companion, 2012.
- [6] Wikipedia. Turing test.
http://en.wikipedia.org/wiki/Turing_test. (Date accessed 07-FEB-2015).