3.9 Guillaume Jeanne, François Desplanques/ Attacks using malicious devices : a way to protect yourself against physical access

3.9.1 Guillaume Jeanne

`https://www.linkedin.com/pub/guillaume-jeanne/54/871/b9a`

3.9.2 François Desplanques

`https://www.linkedin.com/pub/fran%C3%A7ois-desplanques/67/7ba/475`

3.9.3 Attacks using malicious devices : a way to protect yourself against physical access

In recent years, attacks by external devices have experienced a growing interest. These devices are everywhere, we live with them and take them everywhere, even at work. By creating corrupted devices, we can break into private networks which are not connected to the Internet. Just plug the device. This study mainly focuses on attacks by programmable USB devices. To begin with, we make an inventory of the potential of these attacks. Then we analyse weaknesses of these attacks and we give several ways to improve them. Finally, we discuss about various existing measures to limit the impact of such attacks and give countermeasures to our own improvements. Talk can be downloaded from `http://grehack.org`

Attacks using malicious devices : a way to protect yourself against physical access

François Desplanques
Ensimag Student
firstname.lastname@ensimag.fr

Guillaume Jeanne
Ensimag Student
firstname.lastname@ensimag.fr

ABSTRACT

In recent years, attacks by external devices have experienced a growing interest. These devices are everywhere: we live with them and take them everywhere even at work. By creating corrupted devices, we can break into private networks which are not connected to the Internet. Just plug the device. This study mainly focuses on attacks by programmable USB devices. To begin with, we make an inventory of the potential of these attacks. Then we analyze weaknesses of these attacks and we give several ways to improve them. Finally, we discuss about various existing measures to limit the impact of such attacks and give countermeasures to our own improvements.

Keywords

Hardware attacks, USB-based microcontroller, Teensy, security, fake peripheral, pentesting

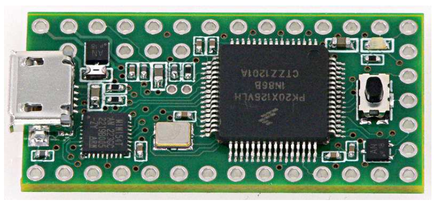
1. INTRODUCTION

Nowadays, hackers show more and more ingenuity to compromise their victims. Remote attacks are no longer standing alone. Recently a new way was opened : hardware attacks using external peripherals. These peripherals are everywhere, in our pockets, in our bags, at home and each one of us owns dozens of them. They are USB keys, external hard disks, mobile phones, mp3 players, cameras and even keyboards or mice. They are objects of daily life : we live with them and take them everywhere : in the street, at school, even at work. These objects contribute to the big flow of data, but without using the Internet. They participate in their own data stream that can be compared to old mails, which were taking a while to get from one point to another and where the information was vacant during the ride whereas the Internet would be emails of nowadays. These devices contain a lot of information and even private documents, such as business or personal data but they also allow to extract information quickly.

These attacks were quickly mediated because they hit critical industrial systems such as nuclear power plant with Stuxnet and Duqu[1], thought these systems were not even connected to the Internet.

These attacks need to interface with the computer, they target features of input-output controller (eg: network cards, firewire controller, programmable PCI controllers, Ethernet). Here we will focus on the USB port, which has the advantage of being present on all computers and is reachable by everyone.

The context is the following : an attacker has a malicious USB device and has a way to physically connect it to the USB port of the victim's computer. We program these at software level, which means they can usurp functions of a peripheral like a keyboard or a mouse. This facilitates the implementation and makes the detection more complex since it is difficult to differentiate the actions of a legitimate user between those coming from the microcontroller. This can happen in different ways such as an employee who compromise the network of his company (intentionally or not). Another way is to hide the peripheral in a common object such as a keyboard or mouse and offer it to the target[2]. Indeed we can be attacked everywhere (at the office, at home and so on) by anyone (a colleague, a friend, ...) It just needs a few seconds while we are not aware of what is going on in our computer. Once this is set up the attacker no longer has physical access to the computer nor to the device. To carry out such attacks, one of the most appropriate device is The Teensy[3].


Teensy 3.0

The Teensy USB Development Board was presented for the first time as a malicious device at the DEFCON18 in 2010 by Adrian Crenshaw [4]. He demonstrated how the Teensy can act as a keystroke dongle. It can type commands unbeknownst to the user. Its small size allows it to be hidden in a real keyboard, in a mouse, or else on a USB key. This facilitates the step of connecting the device to the victim's computer.

Attacks of this type are being democratized and the community has grown a lot. Teensy is used today as a pentest device[5]. In this paper we will focus on possibilities offered by the Teensy microcontroller, in order to compromise other systems.

At first, we show a list of possibilities offered by the Teensy: what are the attacks that have been developed for 3 years. Then, we will focus on limitations with these attacks and how we can improve them. Finally we define some countermeasures appropriated against this type of attack. We are going to alternatively turn us into the skin of the attacker and defender.

2. AN OVERVIEW OF TEENSY ATTACKS

Kautilya[6] is an open-source toolkit which provides various programs for a Human Interface Device (HID) which may help in order to break in a computer, these programs are called payloads. The whole Framework is written in Ruby. Payloads are especially designed to be stored on the initial memory of the Teensy. It also gives the opportunity to work with payloads on Windows/Linux/Mac OS X. There is a great diversity of payload, especially on windows. Most important ones are :

- Change the default DNS server
- Add a user en enable Telnet
- Download and Execute
- Keylog
- Sniffer
- Browse and Accept Java Signed Applet
- Code execution using DNS TXT queries
- Wireless Rogue AP

All payloads are provided with a description when you select them. Then, there is a help menu, which step by step help to design the payload as you wished. Some payloads require the administrator mode but a lot of them do not need this. For example, the keylogger payload does not need such privileges. This payload runs a keylogger written in powershell (the Windows advanced shell) and pastes keys to the website Pastebin, where we can anonymously download and upload text documents, as a private paste after a given interval which can be configured.

Furthermore, Kautilya also gives some tools like scripts in order to make the life of the attacker easier. For example, a script can be used to translate the raw data from Pastebin to the letters that the user typed.

It is very usual for Kautilya payloads to use Pastebin in order to communicate with the outside. This is also the case for the "Download and Execute". It is one of the most interesting payload because it allows to execute any program by the targeted computer. In this case, the executable must first be uploaded in text format on Pastebin using a provided script. When executed, it downloads the file and converts it into an executable file thanks to another script, then it executes the file. This allows to avoid detection by an intrusion detection system (IDS) but this not bypass the antivirus detection because it is activated when the file is executed. It must take care of using a malware whose signature is not yet recorded by antivirus vendors.

Another framework we have experimented is the Social Engineering Toolkit (SET)[7]. This framework is about pentesting and is more general than Kautilya : it does not focus on the Teensy even if a part is devoted to Arduino microcontrolers. Those attacks are in the "Social-Engineering Attacks" then "Arduino based vector attack". Here are the most important ones:

- Powershell HTTP GET MSF Payload
- Internet Explorer/Firefox Beef Jack Payload
- Go to malicious java site and accept payload
- SDCard 2 Teensy Attack (Deploy Any EXE)
- X10 Arduino Sniffer PDE and Librairies
- Peensy Multi Attack Dip Switch + SDCard attack

An interesting point of the SET framework is the use of the SD Card to deploy any files on the victim's computer.

We decided to focus on the Windows Operating System because it is the most used in the world, both by individuals and companies. It is also the most studied operating system on the computer security field. Nevertheless, since the Teensy acts as a keyboard, our solutions could be easily adapted for Linux and MacOS.

3. PROBLEMS AND SOLUTIONS

3.1 Apparent shell window

All payloads are using shells to execute commands. It is launched by the cmd or the powershell on windows. When they are launched, a window is opened on the desktop and therefore the user can see it. It would be better to do this while the user is gone and we will see later how it is possible to detect the presence of the user. As there is no 100% reliable method to detect the user, we have to take into consideration the possibility of him being there. By seeing the shell, the user will be suspicious. So we decided to make this window less conspicuous. It is important to know that the window shell must be in the foreground in order to intercept keystrokes typed. To reduce the visibility of the window shell, we decided to move it to the bottom right corner of the screen. Furthermore on Windows, users cannot click on the close button because it is not apparent.

To achieve that goal, we use a nice feature of the Teensy: it can act as a USB mouse. We also noticed that this feature was never used on others payloads that we studied. It is probably because we can do few actions: click, unclick and move the mouse pointer. But we can only move the pointer to a relative position from the current position of the pointer and we do not have this initial position. Besides, we do not know where the windows are displayed on the screen. To hide the window, we used the trick corresponding to the following algorithm:

```
Procedure Hide window:
    Press Alt+Space+N
    Move pointer to top left
    Shift pointer
    Click
    Move pointer to bottom right
    Shift pointer
    Unclick
end;
```

The command Alt+Space+N is standard on Windows and works for every windows. Its aim is to put the window on the maximize size and stuck it to the top-left corner.

3.2 Auto Correction

As Teensy acts as a keyboard, it can send data to the system, but the reception is more complicated. We have no clue of what is going on the victim's system. When payload are launched, there are several factors that can prevent the payload from being executed: the user closes the window, types on the keyboard or if commands are sent to fast for the computer which executes them.

The goal is to know if the script failed in order to restart later if need be. It is known as auto-correction and it was studied in [8]. The only data sent by the system to the Keyboard are the state of 3 leds: NumLock, CapsLock and ScrollLock.

To detect this, the Teensy began writing a little powershell script which allows to turn the CapsLock led on.

The Teensy checks if the CapsLock led is on and if it is, it turns it off. Otherwise it restarts everything from the beginning or return to the state "detection of user", depending of the payload implementation. Then this script could be launched after each major step of the payload to see if everything is going well.

Instead of using the CapsLock led, we decided to use the ScrollLock led in order to make that check. Indeed this led is usually less present on Keyboards. It is especially true on laptops. Thus the user will not see the led blinking. It has also no impact of the action of the user whereas CapsLock can bother the user and make him to realize that something is happening on his computer.

Another technic to verify data that have been transmitted successfully is to store each command in a batch. Then we can compute a CRC on this file in order to be sure that no perturbation happened during the typing of the script. If it is a sucess, the powershell can turn a specified led of the keyboard on. Thus the Teensy would know that the execution went well. Otherwise, the Teensy would demand again the execution of the script after a delay if the led is not lighting on. By this way, we ensure that the script executed on the computer is correct.

3.3 Limited internal memory force the use of the Internet

As Teensy is a low cost peripheral, it has limited features, only 16kB of RAM and 130kB of storage.

Specification	Teensy 3.0
Processor	MK20DX128 32 bit ARM Cortex-M4 48 MHz
Flash Memory (B)	131072
RAM Memory (B)	16384
EEPROM(B)	2048

This implies that all payloads cannot be stored on the memory. For example an attacker may want to adapt his program to several type of computers which increases the size of the program a lot. It is also especially true in the case when you want to deploy a file on the computer victim: the size is yet limited by the capacity of the device.

We have already talked about the trick that Kautilya uses for that problem: it downloads the payload from Pastebin. However it fosters another problem: the use of the Internet. It is a problem for two reasons. First, the attacker must assume that his victim owns an Internet connection, which will be able to access on the website. Then we loose the advantage of having a physical access to the computer by reaching the web. The activity of the Teensy is more visible when using the Internet because packets are being transferred through the network. Thus, the network administrator will be more easily informed of a suspicious activity and may be able to detect the device. The Internet in a company is very often monitored by the responsible of the Security. By scanning the connection, he may detect a suspicious activity because there is a permanent flow of data trying to reach Pastebin at a regular interval. In a security audit, one of the first thing that will be scanned is packets transferring through the network. Therefore it is really important to find a solution which do not use the Internet and stay local to the computer. It is a *sine qua none* condition for the discretion of an attack.

Therefore there are two problems which are related one to another: the limited capacity of the Teensy and the detection when using the Internet.

In order to avoid the problem of limited capacity, we have soldered a memory card reader to the Teensy. It can be done on the Teensy 3.0 as described in the official website[9]. By this way, we are able to connect SD card in order to take benefits of several GigaByte of storage. It is now therefore possible to embedded files.

It still remains the problem to deploy this file on the victim computer. The Social Engineering Toolkit provides a solution for transferring a file from the SD card to the computer.

The executable is translated into a text file just like it was done with Kautilya. Then the Teensy opens a NotePad in the aim to recopy every data from the SD Card. This is done by using only the Keyboard: every characters are directly pressed. Afterwards, it also uses a powershell script which converts this format into a real executable.

This method has a serious drawback. It can be very long to type all the code from your executable to the Notepad. Indeed the theoretical maximum rate of a keyboard is 500 characters per seconds. Some Operating System are even limiting up to 62 characters per seconds[10]. Moreover the text version of an executable doubles the size in bytes to be transmitted. For example, the magic flag 'MZ' of a PE is considered as two bytes in a program: 0x4d and 0x5a. When the text version is made, it is written "4d5a" in the text file which is actually 4 bytes. Given a 50KB executable, the text version will be 100KB. It will take between 3 minutes and 20 seconds, if you have the maximum bandwidth, and 28 minutes, with the limitation, to be transferred to the computer. Remember that during all this time, a window for the notepad is still active in the desktop of the user.

Therefore we wanted to improve this time by using directly a protocol by USB. The Teensy offers this kind of mode with RAW HID or the Serial mode.

The RAW HID mode has been created in the aim to send raw data on the USB port. With that protocol, we can transfer the file, segmented into packets of size 64 bytes, directly to the computer without using the network. It can only be detected locally, by the user of the computer.

The speed rate is also much better than the previous method, we have experimented a rate of 10,5 KB/s bytes per second on a normal laptop. Moreover the translation into the text mode is no longer required, it can directly send the data of the file using the SdFat library [11]. For the same 50KB executable, it needs about 10 seconds. Moreover there is no window opened during that time.

The Serial mode is also used to send data. In the case of the Teensy, it uses indifferently either the port COMX with X varying between 1 and 9. The first step is to identify the port really used by the Teensy to send data. With this mode, the Teensy can send or receive packets of one byte.

The speed rate is even better than the RAW HID method because we have experimented a rate of 100 KB/s, which is 10 times faster. That is the reason why we'll use this mode when we want to transfer data.

In order to use one of this mode, we need to have a program which runs on the computer's victim in the aim to receive data from Teensy. That's the reason why we need to set up a protocol of communication between the Teensy and the computer.

3.4 Communication

One of the biggest problem with the Teensy is the lack of communication. It is difficult to adapt the behavior of the Teensy in function of the behavior of the computer. As we have seen in the Auto-Correction part, it is possible to make a protocol using the Keyboard leds. There is three leds which can be turned on and off so it makes 3 bits of information. It is not very quick and it is not very discrete. Indeed it seems like Morse Code when you want to dialog with such a protocol because leds are always blinking.

In order to communicate between the computer's victim and the Teensy, we worked on creating a more efficient interface, which will be a listener. The goal of this program is to listen directly to the commands of the Teensy and execute orders that are demanded. It would be an infinite loop which wait for the order of the Teensy. For example, here is the case of a deployment of a file in the victim's system:

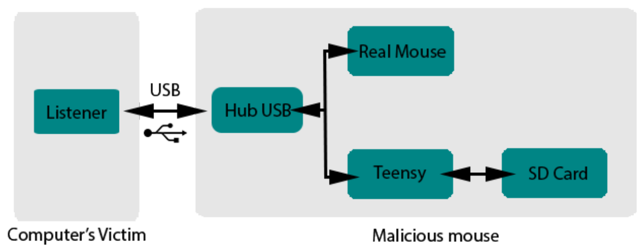
```
While (1):
Wait for an Order
    Switch (Order):
    Case : Deploy a file
        Scan until the COM port has been found
        Receive size of deployed file
        For i in 1..size:
            Copy received packets to the file
        Close the file
        Launch it in background
    Case : ...
```

We worked on a protocol of communication for the deployment of a file. At first, the listener scan which COM port is being used by the Teensy. Then the first data sent by the Teensy would be the size of the deployed file. It will be terminated by an "X" meaning that next packets sent are the data of the deployed file. For each data received, the listener copies it to the file previously opened. Then, the file is closed and it can be launched in background, meaning that there is no apparent window on the Desktop.

The Teensy would just have to send data directly from the SD Card:

```
Initialize the Serial configuration
Initialize the SD Card
Send on Serial the command to deploy a file
Send size of deployed file
Send a "X"
Send packets from SD Card of the file
```

Here is the architecture of the communication for a malicious USB mouse:



Architectural Diagram

There is a USB HUB in order to connect both mouse and Teensy. By this way, the Teensy is invisible to the victim and it can communicate with the SD Card in order to drop some executables, directly by sending data to the listener.

As you may have noticed, there still is one problem with this listener. How can we send this executable? We used the drop payload style of SET in order to do so. But the goal is to be as discrete as possible and we wanted to minimize the time of the transfer. The only parameter to play with is the size of the file. In order to achieve that, we made a program as light as possible. To that purpose we wrote it directly in assembly with MIASM[?] and make the executable with Elfesteem[?]. We used only one section that will contain both the code and the import directory. We reached a size of 996 bytes for our listener.

At this step, we are still complying to the SET payload type, which write bytes directly to the NotePad and then open a powershell to transform the text version into an executable. However we decided to go further : we don't want to open a Notepad and then a Powershell since it is two steps that can be done in one, directly in the Powershell. Besides the major part of the whole process is to transfer the text file. For that purpose, we decided to encode the text file into Base64. Indeed there are only 95 printable keys on the keyboard and the closest inferior multiple of 2 is 64. We lower the size of the file to 6828 Bytes.

We have experimented a time of 14 seconds to launch the listener, by only pressing keys with the Keyboard as the Teensy demanded.

Afterwards, the communication protocol is in place : everything can be done with a maximum of discretion. The Teensy can adapt his reaction to the behavior of the computer. For example it can be used to detect the version of Windows and then select an appropriate exploit from the SD Card adapted to the version of the system. Once the protocol is set up, the attacker have myriads of possibility.

The dropper is only an example of what can be done with such a listener but we could guess others commands which can be executed. For example the listener could provide pieces of information about the victim's computer in order to adapt payloads to the configuration of the computer. Indeed you can retrieve those by the command ”ver” or ”systeminfo” in cmd.exe. More interesting point, you can try to determine the antivirus of your victim and disable it. You also can modify the registry base of the computer in order to make the listener or another program persistent.

We could also improve the listener in order to act as a server between the computer and the Teensy. Once it is uploaded on the computer, it can send information to the Teensy, for instance sending user keystroke or private documents to the SD card if the attacker's plan is to get back his Teensy later or identifying version of softwares to try a privilege escalation. The advantage of using directly the Teensy and not doing it by a program himself is to usurp the user identity. Indeed some commands would be rejected by the system if it is a program that demanded them. But we can think of using the keyboard and the mouse to do it, making the system believe that the user is querying it. By this way, we can for example disable the firewall.

The listener is therefore very powerful when installed. In order to check that we can reinforce the success of the Installation by setting up a CRC at the very beginning of the file. It would compute it himself and then send it to the Teensy. If it is correct, the installation has been completed. Otherwise, if there is no answer or the checksum is incorrect, the Teensy would demand a reinstallation of the listener.

3.5 Detecting user presence

As we have seen previously, the execution of an attack by keystroke is very showy. The attacker can only type commands in a window at the forefront of the operating system. So it is essential to detect if the user is physically present at his post before starting execution because if he is, he may close the shell window that opens or type with his keyboard, which will modify the script we are writing.

To do that, we can turn on the capslock led, which has the effect that the user will write in capital letters. Then we wait ten minutes. If, after this time the led is still on, we can assume that the user does not use his computer. Of course this method is not optimal, it means only that the user is probably not using his keyboard. But he can be in front of the computer, watching a movie, or using only the mouse. Today, considering the small amount of information that Teensy can receive from the system (only status of 3 leds) there is no 100% reliable method. Therefore it justifies the fact that we try to minimize the visual impact and the execution time of Teensy attacks.

4. COUNTERMEASURES

First we have to contradict a popular belief: using an account without administrator privileges of the system is not sufficient to protect against Teensy payloads. In fact, several payloads do not need administrator privileges and they can be very harmful: keylogging, download and execute. Furthermore all Teensy payloads can bypass the Windows User Account Control (UAC), which is present since Windows Vista, and have for goal to prevent the system from undesired modifications. To do this, the Teensy acts as if the user clicked ”yes” by pressing the left arrow and then enter.

Another weak protection against Teensy is to block storage device, ie block devices that contain memory, such as flash drives or hard drives. Indeed this policy is often applied in companies to avoid the leak of information. But here the Teensy is not seen as a storage device but as a keyboard, even if it has a SD card.

At the recent conference SSTIC 2013, a talk[12] advises companies to establish a white list of allowed USB devices by matching the USB descriptor of the device. This descriptor contains following information: the USB class, the unique couple vendor ID/product ID, the string description of the device and many additional information such as the serial number. With this protection, you can allow only one type of keyboard of a certain brand. This protection can be bypassed by the Teensy by changing constants VENDOR_ID and PRODUCT_ID in the file usb_desc.h. This file contains all the variables related to the USB descriptor, it is dynamically compiled when we compile a payload for the Teensy. These constants represents the USB descriptor that the system will see. The attacker still have to determine exactly which keyboard the victim uses.

Another way in which we might think to protect us is to allow only a single pair keyboard / mouse for each computer. But once again, it is weak: it is possible to program the Teensy to become a real mouse, connecting it the laser sensor, clicks and wheel. In addition to being a real mouse, it may also behave maliciously[13]. This remark is also valid for a keyboard. So the system sees in this case only one mouse or keyboard connected. It also makes the detection of the activity of the user much easier because you can know if he is using his mouse or not.

Finally the best protection is, as far as possible, physically or software convict USB ports on the computer. This is possible via the bios or by setting this key to 4 in the Windows register: HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Services\UsbStor

If you know that you are a victim of these attacks, for instance you saw a window shell automatically type script or you discovered a Teensy in your mouse, there are various ways to trace the attacker. First is to check your log browsing history. Kautilya works mainly with Pastebin but it can be adapted to use other website such as dropbox or anything else which allows to store text. But interesting fact is when the payload send content to the internet (keylogging, hashdump, wlan password dump) it contains the login and the password of the user. So you can retrieve the information that have been leaked.

In the case where the payload download an executable, for instance like a meterpreter or a simple shell, you can find by using reverse engineering, the IP server of the attacker. The main problem is to know the code that was executed by the Teensy. Several tools exist to dump the memory of an arduino based controller.

It is difficult to protect against these attacks because the device tries to act as a normal user and the system cannot make any difference. The only difference that we noticed compared to a human is the typing speed of keystroke. As we see previously, the write speed of the Teensy is between 62 and 500 keystrokes per second, whereas a human person cannot exceed 10. We used this difference to create a tool that would block the system if the number of keys typed is too high. Thereby detecting the presence of an attack by keyboard.

To do this, we put a keyboard hook. A hook consists in intercepting function calls to perform pre-processing, such as changing the arguments. Here it is a hook on the function that handles keystrokes in order to compute the writing speed and to cancel the keyboard event if the writing speed is over 30 keystrokes per second. Thereby, Teensy actions will be blocked. This tool has blocked all payloads that we tested, and does not cause any false positive.

But if the attacker has knowledge of such a tool, it can bypass it by adding delays between keys. With our tool, an attacker must add a second delay whenever 30 keys are typed. For instance, the kautilya payload ”download and exec” (that was described in part 2) is one of the fastest payload to execute (14 sec) and sends 1134 key to the system. To bypass our tool with the method we have just explained, this payload will last 51 seconds, which is 3.6 times longer than the original one. This time factor is substantially the same for all payloads, which can lead to very long executions, and thus increase the probability of detecting an attack.

5. FUTURE WORK

Teensy attacks are only in their beginning, there is much point to improve. First, the detection part could be more developed : for instance, we should use sensors such as a micro to detect if the user is in front of the screen, watching a movie, or typing on his keyboard. An advantage of the Teensy is his arduino based architecture which allows to connect a lot of extensions. This method would be much more efficient than to look at the keyboard leds. It also remains to implement the possibilities mentioned in section 3.4 about the listener in order to have better interactions between the victim's computer and the Teensy. Finally, to validate our solutions, we could also test the effectiveness of these attacks in a real environment such as a company.

6. CONCLUSION

Today's attacks using external devices are growing. USB stick, mobile phone, mp3 player or even a mouse can be a source of infection. The range of possibilities opens up from day to day, for instance at the upcoming BlackHat Conference, a research team will present a way to inject malware into iOS Devices via malicious chargers[14]. In this article we look at the possibilities offered by this type of attack by focusing on the Teensy microcontroller on the Windows operating system, although the results are adaptable for all operating systems. The great advantage of this type of attack is that they are almost undetectable because they behave like a normal user. This article shows the weaknesses of various attacks and ways to improve them. In particular we implemented a technique for removing network queries during attacks by storing the payload on a SD card and establishing a communication protocol between the Teensy and the target. Finally the article shows that most methods set up against this are inefficient and we propose several ideas to detect or delay them. To illustrate this, we show a proof of concept that can detect an Teensy attack by measuring the tape speed on keyboard. The proliferation of articles on this subject shows that these attacks are underestimated by companies.

We also conclude from this analysis an interesting point: the more information the attacker has about the target (if it uses tools of detection, keyboard model in a company), the more chance he has to succeed in his attack.

This article feeds the debate of the ”Bring Your Own Device” (BYOD) [15], and the boundaries between personal and work spheres. This article shows that the solution to this issue will go through the development of detection tools, which are not sufficiently efficient today.

7. REFERENCES

[1] Stuxnet/Duqu, "http://www.cs.arizona.edu/ collberg/teaching/466-566/2012/resources/presentations/2012/topic9-final/report.pdf."

[2] H. pierce network with jerry-rigged mouse, "http://www.theregister.co.uk/2011/06/27/mission impossible mouse attack."

[3] Teensy, "http://www.pjrc.com/store/teensy3.html."

[4] A. Crenshaw, "https://www.defcon.org/images/defcon-18/dc-18-presentations/crenshaw/defcon-18-crenshaw-phid-usb-device.pdf."

[5] labofapenetrationtester, "http://labofapenetrationtester.blogspot.fr/2012/04/teensy-usb-hid-for-penetration-testers.html."

[6] "Kautilya, http://code.google.com/p/kautilya/."

[7] TrustedSec, "https://www.trustedsec.com/downloads/social-engineer-toolkit/."

[8] O ensiveSecurity, "http://www.o ensive-security.com/o sec/advanced-teensy-penetration-testing-payloads/ - advanced teensy penetration testing payloads."

[9] ZTiKnl, "http://forum.pjrc.com/threads/16758-teensy-3-microsd-guide - teensy 3 microsd guide."

[10] PJRC, "http://www.pjrc.com/teensy/td _keyboard.html."

[11] SDFatLib, "https://code.google.com/p/sdfatlib/."

[12] B. Badrignans, "Attaques applicatives via peripheriques usb modifies infection virale et fuites d'informations," 2013.

[13] nes, "http://www.nes.fr/securitylab/?p=532."

[14] B. iOS malicious chargers, "http://www.blackhat.com/us-13/brie ïňAngs.html."

[15] P. Pailloux, "http://www.lemagit.fr/technologie/securite-technologie/menaces-informatiques/2012/10/05/patrick-pailloux-anssi-declenche-une-polemique-autour-du-byod/."