CO331: Network and Web Security

Tutorial 2: Setting up a virtual security lab and Cracking passwords*

January 25, 2021

Sections 1 and 2 below illustrate how to set up a virtual network to practice pentesting, and how to import VMs hosting the targets to analyse. Section 3 is our first practical tutorial.

1 Setting up a virtual dirty LAN

By default, VirtualBox shares the host OS's network connection with newly-created VMs via a *virtual network adapter*, which appears like a real network adapter in the guest OS. VirtualBox features many different kinds of virtual networks; the two we'll consider below are:

NAT. VirtualBox shares the host OS's IP address with VMs that have NAT adapters attached, and acts as a network gateway for them. This allows VMs with NAT adapters attached to access the Internet, provided that the host OS is online. However, VMs can't communicate with each other via their NAT adapters, because VirtualBox isolates them. The network adapter that VirtualBox creates for a VM by default is a NAT adapter.

Internal network. VMs in the same internal network can communicate with each other, but *not* with the host OS or the wider network to which the host OS is attached (including the Internet).

A VM can be connected to up to four virtual networks of any type simultaneously.

We'll create a *virtual dirty LAN*, a VirtualBox internal network suitable for VMs running vulnerable guest OSes or hosting exploitable services:

> 1. Open a terminal on the host OS.
>
> 2. Run `VBoxManage dhcpserver add --netname dirtylan --ip 10.6.66.1 --netmask 255.255.255.0 --lower-ip=10.6.66.64 --upper-ip=10.6.66.128 --enable`.

We now have a VirtualBox internal network named `dirtylan` that uses the 10.6.66.0/24 IP address block, and a DHCP server that will automatically assign IP addresses between 10.6.66.64 and 10.6.66.128 when VMs in `dirtylan` request them; this saves us from having to configure IP addresses manually on each VM. The network only needs to be set up once: the settings for `dirtylan` will be saved in a VirtualBox configuration file in your home directory, and the internal network will be re-created every time you start VirtualBox.

Now we'll attach `kali-vm`[1] to `dirtylan`, and set up a NAT adapter so it can still access the Internet when necessary:

> 1. Shut down `kali-vm`.
>
> 2. In VirtualBox Manager, right-click on `kali-vm` in the left-hand list and choose **Settings**.

---

*Based on material prepared by Chris Novakovic `c.novakovic@imperial.ac.uk`.

[1] You should have previously installed Kali following the guide "Setting up VirtualBox and Kali", available on `https://materials.doc.ic.ac.uk`.

3. Choose **Network** from the left-hand list.

4. *Adapter 1* should already be enabled and attached to *NAT* — change this adapter so it is attached to **Internal Network** and enter the name **dirtylan** in the box below.

5. Go to the **Adapter 2** tab, check **Enable Network Adapter**, and attach it to **NAT**.

6. Click **OK**.

For Kali to be able to use both the internal network adapter and NAT adapter at the same time, some extra configuration is needed:

1. Start `kali-vm`.

2. Run `nm-connection-editor`.

3. Click **Wired connection 1**, then click **Edit**.

4. In the **Connection name** box, type **Dirty LAN**. In the **Device** dropdown box, choose **eth0** (this is the internal network adapter from the perspective of the guest OS). Go to the **IPv4 Settings** tab, click **Routes…** and check **Use this connection only for resources on its network**. Click **OK**, then **Save**.

5. Click **Add**. Make sure **Ethernet** is the selected connection type, then click **Create…**.

6. In the **Connection name** box, type **NAT**. In the **Device** dropdown box, choose **eth1** (this is the NAT adapter from the perspective of the guest OS). Click **Save**, then **Close**.

7. Reboot `kali-vm`.

`kali-vm` is now connected to both the dirty LAN and the Internet; you should be able to browse the web using Firefox, for instance. You can control each of the virtual adapters attached to `kali-vm` independently; for instance, you can keep `kali-vm` connected to the dirty LAN while allowing or denying it access to the Internet by doing the following:

1. In the VirtualBox `kali-vm` window, right-click the icon in the bottom toolbar.

2. To connect `kali-vm` to the Internet, make sure **Connect Network Adapter 2** is checked. To disconnect `kali-vm` from the Internet, make sure **Connect Network Adapter 2** is unchecked.

> ### WARNING
>
> Disconnect `kali-vm`'s NAT adapter while using `kali-vm` to probe or attack other VMs in the dirty LAN to prevent any possibility of accidentally targeting a computer on the host OS's network (or the Internet).

Each enabled virtual adapter is visible in the guest OS using the normal networking tools available in that OS; for instance, in Linux, adapter 1 is assigned the standard network interface name `eth0`, adapter 2 is assigned the name `eth1`, and so on. You can use the standard Linux networking utilities (e.g. `ifconfig`) to view the properties of these adapters, including discovering which IP address each one has been assigned by VirtualBox; e.g. in the output for `ifconfig <interface name>`, the virtual adapter's IP address is listed next to `inet addr:`.

Virtual networking is a complex topic, and there's much more to it than what we've covered in this section — if you'd like to know more, including the other types of virtual network you can set up, see Chapter 6 of the VirtualBox manual.

No more changes need to be made to `kali-vm` for now, so it is a good time to take a final snapshot. If you make more changes of your own in future, you might like to take further snapshots.

2 Importing virtual appliances

You don't just have to create VMs and install guest OSes inside them yourself: you can also *import* a VM that was created by someone else and exported as a *virtual appliance*. A virtual appliance is simply an archive file containing a VM's virtual disk and a file describing all of its settings (but not its snapshots).

For the final part of this guide, we'll import a virtual appliance created for our first lab tutorial, named `password-leaker`, and add it to our dirty LAN.

1. Download `password-leaker` from `https://www.doc.ic.ac.uk/~maffeis/331-vms/password-leaker.ova` (requires authentication, only students registered at level 2 or 3 should be able to access).

2. In the VirtualBox Manager menu, choose **File**, then **Import Appliance…**.

3. Click 🖼, then browse to the file you downloaded in step 1. Click **Next**.

4. In the next pane, you'll see the VM's settings as they were when I exported the VM on my own computer; these need not be changed. Select **Generate new MAC addresses for all network adapters**: this will set the MAC address of the four virtual network adapters to new random values, to eliminate the possibility of a MAC address collision between different VMs on the same virtual network (which would cause problems). Click **Import**.

5. After the appliance has been imported, it appears as a regular VM in the VirtualBox Manager window. Right-click on `password-leaker` in the left-hand list and choose **Settings**.

6. Choose **Network** from the left-hand list, then go to the **Network** tab. In the *Adapter 1* tab, check **Enable Network Adapter** and attach it to **Internal Network** with the name **dirtylan**, like you did with adapter 1 for `kali-vm`. Click **OK**.

7. If you're on a lab machine, move the `password-leaker` VM to your external drive, in the same way that you did for `kali-vm`: follow the instructions from Section 2 of the Virtualbox and Kali guide, replacing `kali-vm` with `password-leaker` in the commands.

Start the `password-leaker` VM; after it finishes booting, it shows a login prompt.

Now it's time to find `password-leaker` in the dirty LAN. Since you don't have a user account on this VM (yet!), you can't run `ifconfig` to find out the IP address it has been given, so you'll have to guess it; in section 6, we set up VirtualBox to assign IP addresses beginning at 10.6.66.64, so if you don't have any other VMs running in the dirty LAN, chances are it's been assigned the IP address 10.6.66.65 this time. In `kali-vm`, open Firefox and visit `http://10.6.66.65` — if everything worked, you should see a congratulatory message; if the connection times out, work your way up to `http://10.6.66.70`. If you still don't see a web page, something wasn't set up properly — ask for help in a lab session or office hour, or ask on Piazza otherwise. In a future tutorial, we'll see how to use `nmap` to automatically scan a network to find hosts that are present on it.

You now have a working virtual security lab: you can import VMs running vulnerable guest OSes or hosting exploitable services into VirtualBox, attach them to your virtual dirty LAN, and safely perform whatever security analysis you like on them.

3 Breaking password hashes

In Module 5, we saw that guessing passwords associated with user accounts is a typical way of gaining and maintaining access to a system. This task becomes significantly easier if you know the users' *password hashes*: rather than having to guess a user's password *online* (i.e., by repeatedly attempting to log in to the target system with a user's name and a guess at the password), you can instead guess it *offline* (i.e., hash a password guess in the same way the target system originally did when the password was set, and compare the result to the password hash you have for that user). Offline password-guessing attacks have several advantages for attackers: it's orders of magnitude faster to break password hashes than it is to perform login attempts on the system in real time, and it eliminates the risk of getting caught making an excessive number of failed login attempts by the system administrator.

On standalone Linux systems, passwords are stored in hashed form in a file at `/etc/shadow` — this is known as

the system's *shadow file*. Each line in the shadow file represents a single user account; the fields for each user's account are separated with a colon. The user name is the first field, and the hashed password is the second. Given its sensitive nature, the shadow file has very restrictive permissions, and you won't be able to read it as a regular user. Unfortunately, the administrator of the `password-leaker` system has accidentally backed up the shadow file to a directory used by the web server — see if you can find the backed-up copy of the shadow file using Firefox on `kali-vm`. When you do, save the file somewhere on `kali-vm`. (If you're having trouble finding it, see the spoiler at the end of this section for the answer.)

Now that you have the backed-up shadow file, you can mount an offline *brute-force attack* against the user accounts on `password-leaker` by systematically trying to guess passwords for the user accounts contained inside the shadow file. One way to do this is to write a program that extracts the password fields from the shadow file, generates hashes from a list of common words, and compares the generated hashes with the hashes in the shadow file to see which ones match. Thankfully, there's already a tool that does this: it's named **John the Ripper**, and it's available on `kali-vm`. Notice that a lot of the password hashes in this shadow file begin with `$1$`: this indicates that the password was hashed using *MD5*, an old and more easily brute-forcible hashing algorithm; we'll focus on trying to break these hashes.

1. Find a *wordlist* — a text file containing common words from which password guesses will be derived — and save it somewhere on `kali-vm`. To get you started, there are some wordlists in `/usr/share/dict/`.

2. Run John the Ripper in *wordlist mode* against the MD5-hashed passwords: `john --format=md5crypt --wordlist=<path to wordlist> <path to shadow file>`. John will read lines from the wordlist and hash them to generate password guesses; as it finds generated hashes that match hashes in the shadow file, it will output the user name and the corresponding word from the wordlist.

3. Since hashing data is a CPU-intensive activity, you'll get faster results if John can use more CPU cores: if you're on a lab machine, assign 4 virtual CPUs to `kali-vm` and see how much faster the wordlist is processed.

4. When John manages to break a password hash, try to log into the user account associated with that password on `password-leaker`, either via the console on `password-leaker` or via SSH from `kali-vm`. Some passwords will work, and some won't: can you think of two reasons why this might be the case?

By now you should have access to some user accounts on `password-leaker`. The passwords for these accounts will probably be quite weak — just names or dictionary words. A lot of the passwords in the shadow file will remain uncracked; this is because, in reality, people are told to make their passwords stronger by adding some complexity to them (e.g., by appending a number). Attackers can respond to this by mounting brute-force attacks that make more intelligent password guesses that better reflect the kinds of passwords people choose.

John is also capable of generating additional password guesses by applying *word-mangling rules* to words in a wordlist: one rule might be to capitalise the first letter; another might be to replace the letter "e" with the number "3". Word-mangling rules themselves can be combined, which generates more and increasingly complex password guesses. Bear in mind that the longer the wordlist, and the more complex the chain of word-mangling rules, the longer it will take to generate and test every possible password guess. Choosing a good wordlist and writing a good word-mangling ruleset is something of an artform — in fact, there have been entire competitions dedicated to it.

1. Run John in wordlist mode with the default word-mangling ruleset enabled: `john --format=md5crypt --wordlist=<path to wordlist> --rules <path to shadow file>`. You'll notice that John manages to break hashes that it wasn't able to before, thanks to the new password guesses that are being generated by the ruleset.

2. Take a look at the `[List.Rules:Wordlist]` section of `/usr/share/john/john.conf` to see how the wordlist words are being mangled. There are other John configuration files in this directory containing more realistic rulesets that might give better results. If you're feeling like a 1337 hax0r, try writing some of your own!

Spoiler

The shadow file can be found at `http://10.6.66.65/shadow` (if `password-leaker`'s IP address is 10.6.66.65).