Network Security

Assignment 6, 25 May 2020, version 1.0

Handing in your answers: Submission via Brightspace (https://brightspace.ru.nl)

Deadline: Friday 5 June, 23:59:59 (midnight)

Teaching assistants. Please email *all* of us if you have a question.

- Pol Van Aubel <pol.vanaubel@cs.ru.nl>
- Daan Sprenkels <d.sprenkels@cs.ru.nl>

This final assignment uses the homework WiFi network again.

In this assignment you will be using the following tools:

- aircrack-ng: http://www.aircrack-ng.org/
- arpspoof: http://www.monkey.org/~dugsong/dsniff/
- nmap: http://nmap.org/
- ip: `man ip`
- iptables: http://netfilter.org/projects/iptables/index.html
- sslstrip: https://pypi.python.org/pypi/sslstrip/0.9.2
- wireshark, tshark or tcpdump for packet capturing: https://www.wireshark.org/
- wireguard: https://wireguard.com

Again, do not compile these programs from source, but install them using your distribution's package manager. This week, there is one exception to this rule: *In the case of* sslstrip*, use the instructions listed in the exercise.*

Please turn in all your work in plain text files (program source files are also plain text), unless specified otherwise. If you prefer a document with formatting for whatever reason, like including images, use the PDF format to turn in your work (most editors allow you to export to PDF). Note that it's okay to include images separately and then refer to them from within the text files.

When we ask for packet captures, please try to make them as minimal as you can. Don't send us the entire capture that you did for an hour while getting everything to work – select and save only the relevant final few minutes, or make a new capture showing it works.

For this assignment, you will need updated versions of the `netsec-peer1`, `netsec-peer2`, and `netsec-gateway` VMs. You may also want to download version 1.1 of the `kali` VM, although you *can* install all software required for this assignment by hand. **When importing the updated versions of the VMs, remember to select "Include all network adapter MAC addresses"!** Also, you may want to rename your existing group of VMs to avoid confusion.

The assignment continues on the next page!

WPA2

1. Create a folder called `exercise1`.

   (a) Although WPA2 is more secure than WEP, just like any other good cryptographic system it is only as strong as the key material in use. To demonstrate this, you will use aircrack-ng to crack the passphrase of the wireless network where the course administrator is working. We have already taken care of capturing the WPA2 connection handshake, you can download it at https://cs.ru.nl/~dsprenkels/netsec2020/handshake.cap.

   To crack WPA2 passphrases, you need wordlists. A tutorial on how to crack WPA is on http://www.aircrack-ng.org/doku.php?id=cracking_wpa. Ignore the stuff about injecting packets, capturing the handshake etc. We've already taken care of that. The interesting part is section 4. Pointers on where to find wordlists are on http://www.aircrack-ng.org/doku.php?id=faq#how_can_i_crack_a_wpa-psk_network. Since it is not our intention to have you spend hours on WPA cracking, use the wordlist at https://cs.ru.nl/~dsprenkels/netsec2020/GDict_v2.txt.tar.gz. Note that you have to unzip it first (`tar -xzvf GDict_v2.txt.tar.gz`).

   The bssid of the network is `00:0f:c9:0c:f7:93`. If you want to decrypt the capture to see whether you have the correct key, you also need the essid. This is "netsec-wpa". The capture should contain a single DHCP packet. Beware of the Ubuntu decryption bug, however: if you see other stuff you may still have the correct key. The best way to check is to try to connect to the network.

   Keep in mind that the network may not have a running DHCP server so if you fail to connect, try to set a static IP address in the `192.168.84.200-249` range, with netmask `/24` and gateway `192.168.84.1`.

   Write the passphrase you found to a file called `exercise1a`.

sslstrip

2. Create a folder called `exercise2`.

   This exercise is a multi-stage attack. One of the VMs is trying to login to some kind of website, and you are going to sniff their credentials, CTF[1]-style.

   You will be using `sslstrip` in this exercise. You can either install it into your `kali` VM yourself (instructions are given in the relevant exercise), or download an updated version of the VM (version 1.1) which also has the tools you will need for the wireguard exercises.

   First, spin up the *updated* `netsec-gateway` and `netsec-peer1` VMs. Keep the `netsec-peer2` VM offline for the time being.

   Then "physically" disconnect the `eth0` interface of the `kali` VM. In Virtualbox, under `Devices > Network`, uncheck `Connect Network Adapter 1`. After doing this, `ip link` should show `NO-CARRIER` for the `eth0` interface. You should still have access to the internet via the `eth1` interface. After finishing exercise 2, you can reconnect the virtual cable again.

   (a) Use nmap to discover all the hosts present on this network.

   Given the info that one of the hosts is the gateway, and one of the hosts is yourself, what is the IP address of the other host on the network? Save your answer to a file called `exercise2a`.

   (b) Using arpspoofing and wireshark, figure out which websites this host is contacting. Save the network capture in `exercise2b.cap`. Write the URL(s) to `exercise2b`. Note that you may need to also arpspoof its gateway.

   Note: There is some delay between requests in order to not abuse the target website. This delay is approximately 60 seconds in total for all requests as of this writing, so wait at least 60 seconds before concluding that what you are doing is not working!

The assignment continues on the next page!

[1] "Capture the flag." (https://en.wikipedia.org/wiki/Capture_the_flag#Computer_security)

Last Monday, Pol told you about `sslstrip`, which we are going to use in this exercise. This piece of software has been unmaintained for almost a decade now. As such, installing it from the Kali package manager is not supported. We have built a new `kali` VM (version 1.1) that includes everything you should need for this assignment. If you want to install it in your existing VM, however, follow this script:

```
# Install pip2 and libssl-dev
sudo apt update
sudo apt install libssl-dev python-pip

# Install python2 dependencies
python2 -m pip install Twisted==18.9.0 pyOpenSSL==19.1.0 service_identity==18.1.0

# Download sslstrip source and cd into the source tree
git clone https://github.com/moxie0/sslstrip.git
cd sslstrip

# Run sslstrip using the following command
python2 sslstrip.py --help
```

Apart from the instructions above, the documentation (i.e. `python2 sslstrip.py --help`) and explanation in the readme (https://pypi.python.org/pypi/sslstrip/0.9.2) should be enough to get it to work.

A note of caution though: As you have probably somewhat realized by now, sslstrip is academic software that has been unmaintained for a very long time. It tends to throw a lot of errors, most of which are not at all very helpful. However, most of the time it's working just fine[2].

(c) Now, use sslstrip to strip out SSL from its web traffic. The attack can seem to fail without reason. In this case you might have been unlucky with the ARP spoof race. In this case, wait for a couple of minutes, keep the commands running and see if the attack succeed in a subsequent attempt.

Look at the traffic in wireshark and figure out the login credentials that the "user" is submitting to the website. (Hint: Login credentials are usually sent in a HTTP `POST` request.) Save the network capture to `exercise2c.cap`. Explain your approach and write the flag you found to a file called `exercise2c`.

(d) Give a brief description of the process of SSLstripping: what does it do? How does it receive the traffic it should strip SSL from? Does it simply forward traffic, or does it rewrite outgoing traffic as well? Write your answer to `exercise2d`.

(e) Can you think of countermeasures that clients and servers can take to alleviate the threat of SSLstripping? Write your suggestions to `exercise2e`.

The assignment continues on the next page!

[2] I tested the setup with the VMs earlier this week and it all worked fine. Still, ask for help if you are unsure about this.

Wireguard

For the next exercises you must set up some wireguard networks between virtual machines. For exercise 3, you will set up a wireguard "server" on your `kali` VM to which the *updated* `netsec-peer2` VM can connect, and route its traffic over your `kali` VM.

For exercise 4 you will set up your `kali` VM as a wireguard "client" that connects to the *updated* `netsec-peer2` VM and routes all its traffic over that peer.

The reason "server" and "client" are between quotation marks is that wireguard doesn't really have the distinction between servers and clients – all wireguard nodes are *peers*. "Client" and "server" are simply names to give you an idea of the function of each peer. The "client" peer is e.g. your laptop or PC, sitting in a Starbucks, using wireguard to reach a "server" peer that then forwards the traffic to the internet.

In these exercises, the remote endpoint is the `netsec-peer2` VM. Ensure that it is running. Although not required, you may want to turn `netsec-peer1` off.

For exercise 3 you also should not need `netsec-gateway`, because your `kali` VM can handle all the internet routing required, but for exercise 4 you need to ensure that `netsec-gateway` is turned on.

The IP address to use for `netsec-peer2` is 172.21.152.79. The ports used are different for each exercise; for exercise 3 it is port 12345 on both peers, for exercise 4 you will need to use 6789 on both peers.

You can perform this exercise either on a fresh download of the *updated* `kali` VM, or by installing wireguard and downloading our base configuration files yourself.

To install wireguard yourself, run the following commands in the VM:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install linux-headers-amd64
sudo apt-get install wireguard
```

This will:

(a) update your install to a version where you can install wireguard,

(b) ensure that you still have screen resize capabilities by rebuilding the virtualbox modules for the new kernel,

(c) and then actually install the wireguard tooling.

Reboot afterwards.

3. Create a folder called `exercise3` to hold your answers and configuration files.

   Start by reading the frontpage of the wireguard project at `https://www.wireguard.com`, paying particular attention to the Conceptual Overview, which includes Cryptokey Routing.

   Unless stated otherwise, you are only allowed to use the `wg` command and network configuration commands such as `ip route`, `ip address`, and `iptables`. In particular, you are *not* allowed to use `wg-quick`, a quick-setup script provided by wireguard.

   In this exercise, the network that the wireguard peers will use to communicate within wireguard is `10.90.42.0/24`.

   (a) Usually you would have to generate keys to exchange between wireguard peers. However, due to the nature of this setup, we need to use pregenerated keys, which we provide in skeleton configuration files for wireguard. You will need to complete these configuration files for the later exercises.

   However, to ensure you understand how private and public keys are generated, try if you can generate a wireguard private key, and then derive the corresponding public key, using argument for the `wg` command. Of course you can consult documentation and online resources.

   To turn in this exercise, provide a single file called `exercise3a` containing the commands you used, and the private and public keys on separate lines.

. . . . . **The assignment continues on the next page!** . . . . . .

(b) Many of the following operations have to be performed as `root`. You can enter a root-shell with `sudo -i`, that way you do not need to prefix every command with `sudo`.

To verify that your setup is working during this exercise, you can run `ping -I wg0 10.90.42.1` (of course, replace `wg0` with some other interface name if you did not call yours `wg0`). If you get replies, you can reach the other wireguard peer.

You can find a skeleton configuration file for this exercise in `/etc/wireguard/wg0.conf`. Note there are other files; these are for other exercises.

Using the wireguard documentation, the minimum setup you should get working is a VPN allowing communication from your `kali` VM to the *updated* `netsec-peer2` VM. Note that at this point neither machine needs to tunnel all its network traffic over the VPN, it only needs to be able to reach the other endpoint through the VPN.

Edit the `wg0.conf` file in order to achieve this. Try to make the AllowedIPs configuration as restricted as possible.

You will also need to actually create the wireguard interface, add an address to it, and set it "up" using `ip` commands. For this exercise, use any address in the `10.90.42.0/24` range except `10.90.42.1` – that is `netsec-peer2` *inside* the wireguard network.

Don't forget to apply the configuration in `wg0.conf` to this interface using the `wg` command.

Make a shell script of the commands you use.

To turn in this exercise, include this shell script, the configuration file(s) for your endpoint, and the output of `wg` on the command line, in a folder called `exercise3b`

(c) Perform a set of short packet captures on your `kali` VM, while doing a ping from your `kali` VM to the wireguard address of `netsec-peer2`.

The packet captures should be done on two interfaces: one capture on the wg-interface created for the VPN, and another capture on the `kali` VM's `eth1` interface, which is the network interface that is actually carrying the VPN-tunneled traffic. So there should be two captures in total.

To turn in this exercise, create a folder called `exercise3c`. Include these captures, and name them along the lines of `kali-wg.cap`, `kali-eth1.cap`, etc. Also include the commands and (partial) output of the `ping` command and other commands you used during the capture in a file called `capture-commands`.

(d) As you may have seen in exercise 3c, the `netsec-peer2` wireguard peer is sending pings to you, destined for an IP address on the internet. Of course, at this stage, these pings never receive any replies because you are not routing them onto the internet.

Make it so that your `kali` VM *does* route these replies onto the internet, and replies go back to the wireguard peer. Note that this is not meant to be a difficult exercise, and you should not have to change any *wireguard* configuration at this stage. Rather, you need to apply two things you learnt about in earlier assignments.

To turn in this exercise, provide the commands you used to set it up, and a packet capture showing the ping requests and ping replies from both the `eth1` interface and the wireguard interface. Of course, capturing on the `eth1` interface will show wireguard traffic, rather than ping requests and replies. If you see both, that is likely the result of how you set up the routing and is expected.

Hints:

- Remember that acting as a router in linux is called "IP forwarding".
- Bear in mind that the IP addresses being used are in the RFC1918 private address ranges.
- If you run into trouble because both `eth0` and `eth1` on your `kali` VMs have default routes, there *is* a way to configure this so that it works regardless of route taken. If that doesn't help you,

you can either temporarily disconnect the connection on `eth0` (labeled "DHCP" in Network Manager), or remove the default route for the `netsec-gateway` VM using
`ip r del default via 172.21.152.1`

Routing all your traffic over Wireguard

4. Create a folder called `exercise4` to hold your answers and configuration files.

   Using the wireguard documentation and the previous exercise's answers, you will now set up the VPN so that your `kali` VM uses the VPN for all its network traffic, routed by the other endpoint. This is a fairly common usage scenario, and is what is usually meant by "using a VPN service".

   We assume you will accomplish this by using routing tables akin to the one we analyzed in Assignment 4. In this exercise, you *are* allowed to use `wg-quick` if you want to.

   In this exercise, the network that the wireguard peers will use to communicate within wireguard is `10.90.84.0/24`. The wireguard peer address for `netsec-peer2` is `10.90.84.1`

   (a) In the previous exercise, you took care of the *server* side of a client/server VPN setup. However, the client also has to do some work to actually make sure it sends its network traffic into the VPN, and not just onto the internet. How can a client make sure that the traffic goes onto the internet? (There are multiple possible answers, only one is required.) Write your answer to a file called `exercise4a`.

   (b) There is a skeleton configuration file containing the keys you need to use in `/etc/wireguard/wg1.conf`. Set up wireguard on your `kali` VM, so that the `kali` VM uses the VPN for all its network traffic.

   Remember that for this to work, the AllowedIPs configuration must allow all IPs on the wireguard tunnel.

   Make a shell script of the commands you use, and include the configuration file for your endpoint, in a folder called `exercise4b`.

   (c) Perform the same kind of packet capture as in the previous exercise, however, this time your *kali* VM functioning as the client should ping some host on the internet (e.g. www.google.com) instead of the wireguard peer. You can use `traceroute` or `mtr` to figure out whether traffic is actually going through the VPN or whether it's taking the normal route to the internet.

   To turn in this exercise, include the capture, and the commands and output of the `ping` and `traceroute` commands you used during the capture, in a folder called `exercise4c`.

   If you have network issues during this exercise, one thing to do would be to look at your routing table (`ip r show`) and see if you can figure out why traffic is or is not going through the VPN. Of course, send an e-mail if you're stuck.

Turning in

5. Place the directories `exercise1`, `exercise2`, `exercise3`, and `exercise4` and all their contents in a folder called `netsec-assignment6-SNR1-SNR2`. Replace `SNR1` and `SNR2` by your respective student numbers, and accomodate for extra / fewer student numbers.

   Make a `tar.gz` archive of the whole directory `netsec-assignment6-SNR1-SNR2` and submit this archive in Brightspace.