Network Security

Assignment 5, May 24, 2019, version 1.0

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

Deadline: Friday, June 7, 23:59:59 (midnight)

Note: This is a *long* assignment. You have two weeks available for this assignment. Expect more than a day's work.

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

- Daan Sprenkels `<d.sprenkels@cs.ru.nl>`
- Alireza Vahdad `<alireza.vahdad@gmail.com>`
- Jos Craaijo `<jos635@outlook.com>`

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/`
- sslstrip: `http://www.thoughtcrime.org/software/sslstrip/`, `https://pypi.python.org/pypi/sslstrip/0.9.2`
- wireshark, tshark or tcpdump for packet capturing: `https://www.wireshark.org/`
- openvpn: `https://openvpn.net/index.php/open-source/documentation.html`
- wireguard: `https://wireguard.com`

Again, do not compile these programs from source, but install them using your distribution's package manager.

The assignment consists of one long practical 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.

Commands that need to be run with root rights are denoted by a prefix `#`. When a command should be run without root rights, it will be prefixed with `$`. Do not include the prefix when typing the command.

DNS amplification precautions

1. Create a folder called `exercise1`.

   Suppose you are the administrator of a network. Imagine the gateway from the LAN to the internet is *non-NAT'ing.* You want to prevent hosts, from inside the LAN, from being able to execute DNS amplification attacks. The LAN network is `203.0.113.0/24`, the gateway's internal IP address is `203.0.113.1`, and its external IP address is `198.51.100.78`.

   (a) What firewall measure should prevent DNS amplification attacks from originating from your network, *without* impeding normal operation of the network? Write your answer to `exercise1a`.

   (b) Write an iptables configuration that implements this measure. Write your answer to `exercise1b`.

DNS query randomization

2. Create a folder called `exercise2`.

   Assume you're an attacker who wants to trick a DNS cache into believing your server is actually hosting blackboard.ru.nl. You try to race a legitimate DNS server to provide the answer faster, but you are *not* a MITM between the DNS cache and the DNS server.

   (a) How would you ensure that you can predict the queries that the cache is going to produce, and how would you ensure that your answers will be accepted (i.e. pass the bailiwick check)? Describe the setup and/or process. Write your answer to `exercise2a`.

   QID randomization and port randomization are (somewhat) effective countermeasures against cache poisoning.

   (b) If you craft a single blind response, to a single DNS query, what are the odds that you guess right if the DNS cache is only using QID randomization in its queries? Write your answer to `exercise2b`.

   (c) What are the odds if the cache is also using source port randomization? Write your answer to `exercise2c`.

   Imagine that on top of that, these DNS servers also deploy 0x20 randomization (see slides, the random capital letters in the query).

   (d) What are the odds now that you will guess right on a query for the blackboard.ru.nl host? Write your answer to `exercise2d`.

   (e) Describe an attack that would allow you to get a good success rate, even though your odds of guessing a single response correctly are low. Explain the idea behind the attack, and the way it would be performed. Exact calculations of probability are not required. Write your answer to `exercise2e`.

   Until now, we assumed the attacker cannot see the DNS queries. The countermeasures we mentioned are designed with that assumption as well.

   (f) Explain, in your own words, why all these countermeasures are less effective against a DNS attack performed by a passive MitM attacker. Write your answer to `exercise2f`.

   (g) Explain, in your own words, why all these countermeasures do not work against a DNS attack performed by an active MitM attacker. Write your answer to `exercise2g`.

   (h) *Optional.* As a completely optional bonus, calculate the number of queries the cache needs to make on average for your attack from *2e* to succeed with a 90% success rate, assuming you win every race (i.e. for each query it makes, you successfully inject your own response), for all three situations of only QID randomization, QID and port randomization, and QID, port, and 0x20 randomization on blackboard.ru.nl. Write your answers to `exercise2bonus`.

Firewalls and UDP "connections"

3. Create a folder called `exercise3`.

   (a) The firewall configuration you made in assignment 4, exercise 1a, should still allow DNS conversations initiated by an outgoing DNS query, but block unsolicited incoming traffic. However, DNS usually runs over UDP and UDP is a connectionless protocol. Try to explain how the firewall still knows that it should allow this DNS traffic. Feel free to consult iptables documentation for this. Write your answer to `exercise3a`.

   Arya and Bran Stark want to set up a UDP peer-to-peer connection. However, they are both behind a firewall that does not allow incoming connections. They are smart, however, so they look up what UDP hole punching is.

   (b) *Optional.* How can you set up a peer-to-peer UDP session between two different hosts, both behind a firewall that blocks incoming UDP traffic? Show the `netcat` commands that should achieve this goal. Write your answer to `exercise3b`.

WPA2+sslstrip

4. Create a folder called `exercise4`.

   This exercise is a multi-stage attack. Somewhere, there's a website containing your grades for this exercise. Everybody starts out with an O. It is up to you to give yourself the grade you want. (Note that you can still receive an NSI if we feel you have not put in sufficient effort.)

   You are *not* allowed to sniff the general network traffic in order to eavesdrop on other groups performing the attack. Also, please do not change other people's grades while performing this exercise, and don't do anything else on the target website.

   (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/netsec/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/netsec/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 `exercise4a`.

(b) Connect to the network. There should be a DHCP server running. If not, use an IP address in the range of `192.168.84.200-249`, with netmask `/24` and gateway `192.168.84.1`.

Use nmap to scan this network. Find the hosts in the range `192.168.84.1-80`. Disable reverse DNS lookup to speed up things. There should be many hosts, apart from the gateways (`192.168.84.1-20`). Write which hosts you find to `exercise4b`.

(c) From this point onwards you will need to coordinate with other groups, since there is only a limited number of hosts to arpspoof. Do not get in each others way.

Pick one of the hosts that are not the gateways (`192.168.84.1-20`). Its gateway is matched modulo 20 (so `192.168.84.32` and `192.168.84.52` both have gateway `192.168.84.12`, whereas `192.168.84.23` has gateway `192.168.84.3`).[1]

Using arpspoofing and wireshark, figure out which websites this host is contacting. Save the network capture in `exercise4c.cap`. Write the URLs to `exercise4c`. 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 300 seconds as of this writing.

(d) Now, use sslstrip (`http://www.thoughtcrime.org/software/sslstrip/`, `https://pypi.python.org/pypi/sslstrip/0.9.2`) to strip out SSL from its web traffic. The documentation and explanation on the websites should be enough to get it to work[2].

Look at the traffic in wireshark and figure out the login credentials to use. Save the network capture in `exercise4d.cap` and write the login credentials you found to `exercise4d.creds`.

(e) Now, log in to the website, find your grades, and edit them to your desired result. After that, write your student numbers and the result you set to `exercise4e`.

(f) 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 `exercise4f`.

(g) Can you think of countermeasures that clients and servers can take to alleviate the threat of SSLstripping? Write your suggestions to `exercise4g`.

OpenVPN

5. Create a folder called `exercise5` to hold your answers and configuration files.

CNCZ provides an OpenVPN-based Science VPN, which you may find useful at some point. Instructions for this are at `http://wiki.science.ru.nl/cncz/index.php?title=Vpn&setlang=en#OpenVPN_.5Bvoor.5D.5Bfor.5D_Linux_.26_MacOS`.

(a) See if you can get this to work with your Science account. If you are unable to do this within 30 minutes, skip to the next exercise and come back if you have time left over.

(b) Look up in the OpenVPN man page (`man openvpn`) what each line of the configuration file means. For easy searching, append "--" to the first word on the line. So searching for "dev" becomes "--dev".

(c) Perform traceroutes (`traceroute`) to blackboard.ru.nl, www.google.com, www.cs.ru.nl, and to the VPN server itself, with and without the VPN running. Paste the commands you used and their output to a file. Look at the routing table (`ip route show`), and paste it as well.

(d) Explain the differences between the traceroutes, paying special attention to the one to the VPN server itself.

(e) Explain why it is not straightforward to run other services on the OpenVPN server and contact them via the VPN tunnel. Can you think of a solution for this problem?

[1] This somewhat weird network configuration is required to enable you to arpspoof in parallel with other groups. Without going into too much detail, the problem is that if we only had one gateway IP address, we would need as many hardware devices as IP addresses for you to spoof. In most normal situations, a network only has one gateway which all clients will use.

[2] One note of caution though. sslstrip is academic software. It tends to throw a lot of errors. However, most of the time it's working just fine.

Wireguard

For the next exercises you must set up a Wireguard network between two machines. These can be physical machines, e.g. your and your lab partner's laptops, both booting some form of Linux, e.g. the Kali USB stick. Note you can make such a stick yourself using the compressed image available at `https://www.cs.ru.nl/~dsprenkels/netsec/kali-linux-2019.1a-amd64.iso` (with credentials root:toor). You can write this to any USB stick that is at least 8GB in size. This does destroy any partitions and data already present on the stick.

Note that you will not be able to reach each other's machines through eduroam[3], but a direct link using an ethernet cable or an ad-hoc WiFi network usually works. The netsec-wep and netsec-wpa networks should also be usable for this purpose.

You can also use virtual machines. Virtualbox and KVM/QEMU are decent options in this regard. The bootable USB iso image linked above should also be directly bootable as a virtual machine disk.

When using virtual machines, the easiest option is setting up two separate VMs and using VM-to-VM networking. You can do this on a Windows host. On the bootable Kali USB sticks, there is not enough storage to run more than one VM, and if your machine does not have sufficient RAM you may find your system freezing. Setting up virtual machines with Virtualbox is covered in plenty of tutorials so we will not cover that here.

You are also allowed to use the *official* Windows client for Wireguard, or the official Android, or the iOS client. However, you must use the Linux client on *at least one* of the endpoints!

6. 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` and `ip address`. In particular, you are *not* allowed to use `wg-quick`, a quick-setup script provided by Wireguard.

   You will have to generate keys to exchange between Wireguard peers. Don't worry, they're rather short and should be easy to transfer even by manually typing them.

   Bear in mind that due to the nature of Cryptokey Routing, if AllowedIPs stanzas for multiple peers on a single Wireguard interface overlap, they behave the same as routes: more specific routes overrule more generic routes.

   (a) Using the Wireguard documentation, the minimum setup you should get working is a VPN allowing communication between the two machines. Note that 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.

   Try to make the AllowedIPs configuration as restricted as possible.

   Make a shell script of the commands you use, and include configuration files for both endpoints.

   (b) Perform a set of short packet captures on the machine that runs Linux, while doing a ping from one VPN endpoint to the other endpoint and vice versa. The packet captures on each end should be done on two interfaces: one capture on the wg-interface created for the VPN, and another capture on the network interface that is actually carrying the VPN-tunneled traffic (either your normal network interface, or a virtual interface created by e.g. virtualbox). So there should be two captures in total. Include these captures, and name them along the lines of `endpointA-wg.cap`, `endpointB-wlan0.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`.

[3]Unless you use the trick from Exercise 3b

Wireguard VPN (optional)

7. *Optional.* Create a folder called `exercise7` to hold your answers and configuration files.

This exercise is the final boss. It is the exercise where lots of stuff comes together. I know, you're exhausted. Ga ahead and skip it. Many students did not even reach this point. It's okay.

...

You're still here? You *do* have the allure for more network security exercises? Enchanted by the magic of virtual private network setups? Well, you're in the right place.

Using the Wireguard documentation and the previous exercise's answers, you will now set up the VPN so that one endpoint 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 assignment, you *are* allowed to use `wg-quick`.

(a) Think of which *four* configuration settings—apart from a working Wireguard setup—will you need to set up a Wireguard VPN client/server setup? Use this spoiler to check your answer.

(b) Set up Wireguard as a VPN client/server setup, so that the client uses the VPN for all its network traffic.

Try to make the AllowedIPs configuration for the server as restricted as possible. For the client, the AllowedIPs configuration must allow all IPs on the wireguard tunnel.

Make a shell script of the commands you use, and include configuration files for both endpoints.

(c) Perform the same kind of packet capture as in the previous exercise, however, this time the endpoint functioning as the client should ping some host on the internet (e.g. www.google.com) instead of the VPN server. 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. Include the commands and output of the `ping` and `traceroute` commands you used during the capture in a file called `capture-commands`.

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.

8. Place the directory `exercise1`, `exercise2`, `exercise3`, `exercise4`, `exercise5`, `exercise6`, and `exercise7`. and all its contents in a folder called `netsec-assignment5-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-assignment5-SNR1-SNR2` and submit this archive in Brightspace.