Active Reconnaissance

**Task 1 – Introduction**

In the first room of the Network Security Module, we focused on passive reconnaissance. In this second room, we focus on active reconnaissance and the essential tools related to it. We learn to use a web browser to collect more information about our target. Moreover, we discuss using simple tools such as ping, traceroute, telnet, and nc to gather information about the network, system, and services.

As we learned in the previous room, passive reconnaissance lets you gather information about your target without any kind of direct engagement or connection. You are watching from afar or checking publicly available information.



Active reconnaissance requires you to make some kind of contact with your target. This contact can be a phone call or a visit to the target company under some pretence to gather more information, usually as part of social engineering. Alternatively, it can be a direct connection to the target system, whether visiting their website or checking if their firewall has an SSH port open. Think of it like you are closely inspecting windows and door locks. Hence, it is essential to remember not to engage in active reconnaissance work before getting signed legal authorization from the client.



In this room, we focus on active reconnaissance. Active reconnaissance begins with direct connections made to the target machine. Any such connection might leave information in the logs showing the client IP address, time of the connection, and duration of the connection, among other things. However, not all connections are suspicious. It is possible to let your active reconnaissance appear as regular client activity. Consider web browsing; no one would suspect a browser connected to a target web server among hundreds of other legitimate users. You can use such techniques to your advantage when working as part of the red team (attackers) and don’t want to alarm the blue team (defenders).

In this room, we go through various tools commonly bundled with most operating systems or easily obtainable. We begin with the web browser and its built-in developer tools; furthermore, we show you how a web browser can be “armed” to become an efficient reconnaissance framework. Afterwards, we discuss other benign tools such as ping, traceroute, and telnet. All these programs require connection to the target, and hence our activities would fall under active reconnaissance.

This room is of interest to anyone who wants to become familiar with essential tools and see how they can use them in active reconnaissance. The web browser developer tools might take some effort to gain familiarity, although it offers a graphical user interface. The command-line tools covered are relatively straightforward to use.

**Task 2 – Web Browser**

The web browser can be a convenient tool, especially that it is readily available on all systems. There are several ways where you can use a web browser to gather information about a target.

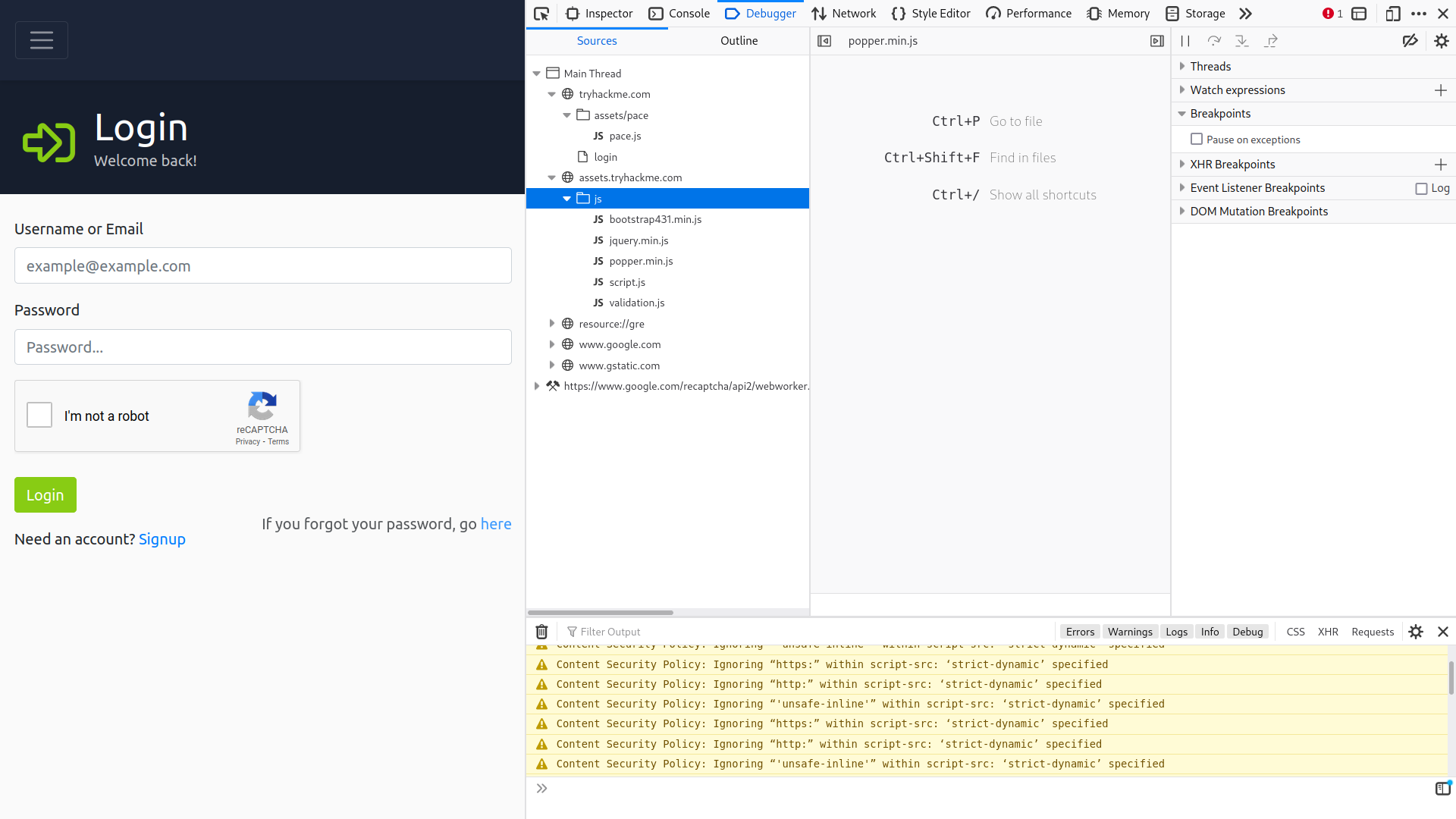
On the transport level, the browser connects to:

TCP port 80 by default when the website is accessed over HTTP

TCP port 443 by default when the website is accessed over HTTPS

Since 80 and 443 are default ports for HTTP and HTTPS, the web browser does not show them in the address bar. However, it is possible to use custom ports to access a service. For instance, https://127.0.0.1:8834/ will connect to 127.0.0.1 (localhost) at port 8834 via HTTPS protocol. If there is an HTTPS server listening on that port, we will receive a web page.

While browsing a web page, you can press Ctrl+Shift+I on a PC or Option + Command + I (⌥ + ⌘ + I) on a Mac to open the Developer Tools on Firefox. Similar shortcuts will also get you started with Google Chrome or Chromium. Developer Tools lets you inspect many things that your browser has received and exchanged with the remote server. For instance, you can view and even modify the JavaScript (JS) files, inspect the cookies set on your system and discover the folder structure of the site content.



**Task 3 - Ping**

Ping should remind you of the game ping-pong (table tennis). You throw the ball and expect to get it back. The primary purpose of ping is to check whether you can reach the remote system and that the remote system can reach you back. In other words, initially, this was used to check network connectivity; however, we are more interested in its different uses: checking whether the remote system is online.

In simple terms, the ping command sends a packet to a remote system, and the remote system replies. This way, you can conclude that the remote system is online and that the network is working between the two systems.

If you prefer a pickier definition, the ping is a command that sends an ICMP Echo packet to a remote system. If the remote system is online, and the ping packet was correctly routed and not blocked by any firewall, the remote system should send back an ICMP Echo Reply. Similarly, the ping reply should reach the first system if appropriately routed and not blocked by any firewall.

The objective of such a command is to ensure that the target system is online before we spend time carrying out more detailed scans to discover the running operating system and services.

On your AttackBox terminal, you can start to use ping as ping 10.10.169.203 or ping HOSTNAME. In the latter, the system needs to resolve HOSTNAME to an IP address before sending the ping packet. If you don’t specify the count on a Linux system, you will need to hit CTRL+c to force it to stop. Hence, you might consider ping -c 10 10.10.169.203 if you just want to send ten packets. This is equivalent to ping -n 10 10.10.169.203 on a MS Windows system.

Technically speaking, ping falls under the protocol ICMP (Internet Control Message Protocol). ICMP supports many types of queries, but, in particular, we are interested in ping (ICMP echo/type 8) and ping reply (ICMP echo reply/type 0). Getting into ICMP details is not required to use ping.

In the following example, we have specified the total count of packets to 5. From the AttackBox’s terminal, we started pinging 10.10.169.203. We learned that 10.10.169.203 is up and is not blocking ICMP echo requests. Moreover, any firewalls and routers on the packet route are not blocking ICMP echo requests either.

user@AttackBox$ ping -c 5 MACHINE\_IP

PING 10.10.169.203 (10.10.169.203) 56(84) bytes of data.

64 bytes from 10.10.169.203: icmp\_seq=1 ttl=64 time=0.636 ms

64 bytes from 10.10.169.203: icmp\_seq=2 ttl=64 time=0.483 ms

64 bytes from 10.10.169.203: icmp\_seq=3 ttl=64 time=0.396 ms

64 bytes from 10.10.169.203: icmp\_seq=4 ttl=64 time=0.416 ms

64 bytes from 10.10.169.203: icmp\_seq=5 ttl=64 time=0.445 ms

--- 10.10.169.203 ping statistics ---

5 packets transmitted, 5 received, 0% packet loss, time 4097ms

rtt min/avg/max/mdev = 0.396/0.475/0.636/0.086 ms

In the example above, we saw clearly that the target system is responding. The ping output is an indicator that it is online and reachable. We have transmitted five packets, and we received five replies. We notice that, on average, it took 0.475 ms (millisecond) for the reply to reach our system, with the maximum being 0.636 ms.

From a penetration testing point of view, we will try to discover more about this target. We will try to learn as much as possible, for example, which ports are open and which services are running.

Let’s consider the following case: we shut down the target virtual machine and then tried to ping <machin\_ip>. As you can would expect in the following example, we don’t receive any reply.

PING 10.10.169.203 (10.10.169.203) 56(84) bytes of data.

64 bytes from 10.10.169.203: icmp\_seq=1 ttl=63 time=225 ms

64 bytes from 10.10.169.203: icmp\_seq=2 ttl=63 time=220 ms

64 bytes from 10.10.169.203: icmp\_seq=3 ttl=63 time=258 ms

64 bytes from 10.10.169.203: icmp\_seq=4 ttl=63 time=232 ms

64 bytes from 10.10.169.203: icmp\_seq=5 ttl=63 time=227 ms

--- 10.10.169.203 ping statistics ---

5 packets transmitted, 5 received, 0% packet loss, time 4110ms

rtt min/avg/max/mdev = 220.298/232.565/257.771/13.184 ms

**Answer the question:**

Q1. Which option would you use to set the size of the data carried by the ICMP echo request?

Ans - -s # see on the ‘ping man’ page

Q2. What is the size of the ICMP header in bytes?

Ans - 8

Q3. Does MS Windows Firewall block ping by default? (Y/N)

Ans - Y

Q4. Deploy the VM for this task and using the AttackBox terminal, issue the command ping -c 10 10.10.169.203. How many ping replies did you get back?

Ans - 10

**Task 4 - Traceroute**

As the name suggests, the traceroute command traces the route taken by the packets from your system to another host. The purpose of a traceroute is to find the IP addresses of the routers or hops that a packet traverses as it goes from your system to a target host. This command also reveals the number of routers between the two systems. It is helpful as it indicates the number of hops (routers) between your system and the target host. However, note that the route taken by the packets might change as many routers use dynamic routing protocols that adapt to network changes.

On Linux and macOS, the command to use is traceroute 10.10.169.203, and on MS Windows, it is tracert 10.10.169.203. traceroute tries to discover the routers across the path from your system to the target system.

There is no direct way to discover the path from your system to a target system. We rely on ICMP to “trick” the routers into revealing their IP addresses. We can accomplish this by using a small Time To Live (TTL) in the IP header field. Although the T in TTL stands for time, TTL indicates the maximum number of routers/hops that a packet can pass through before being dropped; TTL is not a maximum number of time units. When a router receives a packet, it decrements the TTL by one before passing it to the next router. The following figure shows that each time the IP packet passes through a router, its TTL value is decremented by 1. Initially, it leaves the system with a TTL value of 64; it reaches the target system with a TTL value of 60 after passing through 4 routers.

**Task 5 - Telnet**

The TELNET (Teletype Network) protocol was developed in 1969 to communicate with a remote system via a command-line interface (CLI). Hence, the command telnet uses the TELNET protocol for remote administration. The default port used by telnet is 23. From a security perspective, telnet sends all the data, including usernames and passwords, in cleartext. Sending in cleartext makes it easy for anyone, who has access to the communication channel, to steal the login credentials. The secure alternative is SSH (Secure SHell) protocol.

However, the telnet client, with its simplicity, can be used for other purposes. Knowing that telnet client relies on the TCP protocol, you can use Telnet to connect to any service and grab its banner. Using telnet 10.10.169.203 PORT, you can connect to any service running on TCP and even exchange a few messages unless it uses encryption.

Let’s say we want to discover more information about a web server, listening on port 80. We connect to the server at port 80, and then we communicate using the HTTP protocol. You don’t need to dive into the HTTP protocol; you just need to issue GET / HTTP/1.1. To specify something other than the default index page, you can issue GET /page.html HTTP/1.1, which will request page.html. We also specified to the remote web server that we want to use HTTP version 1.1 for communication. To get a valid response, instead of an error, you need to input some value for the host host: example and hit enter twice. Executing these steps will provide the requested index page.