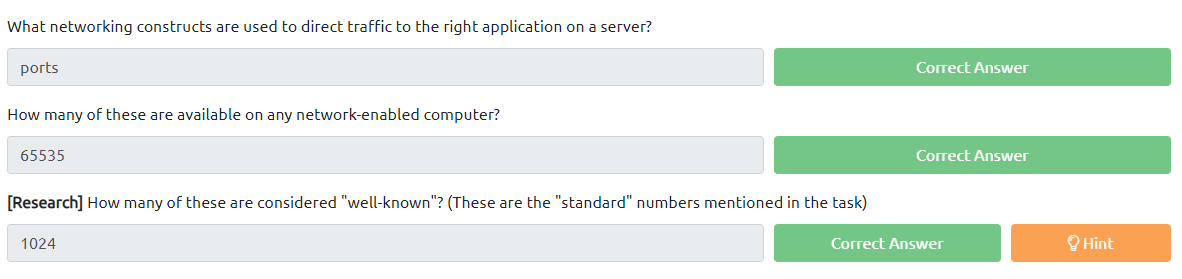
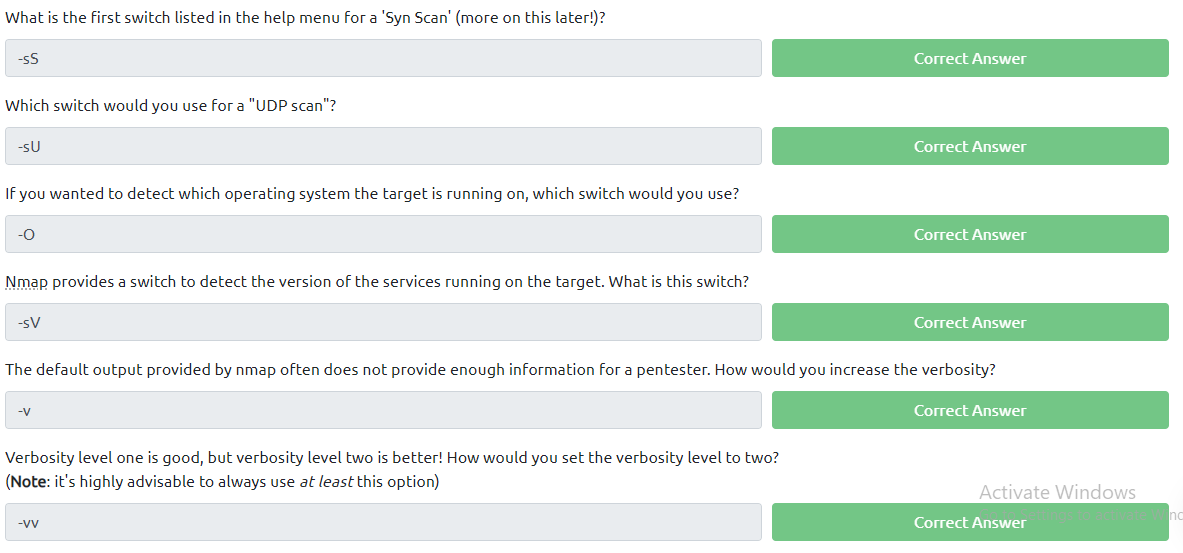
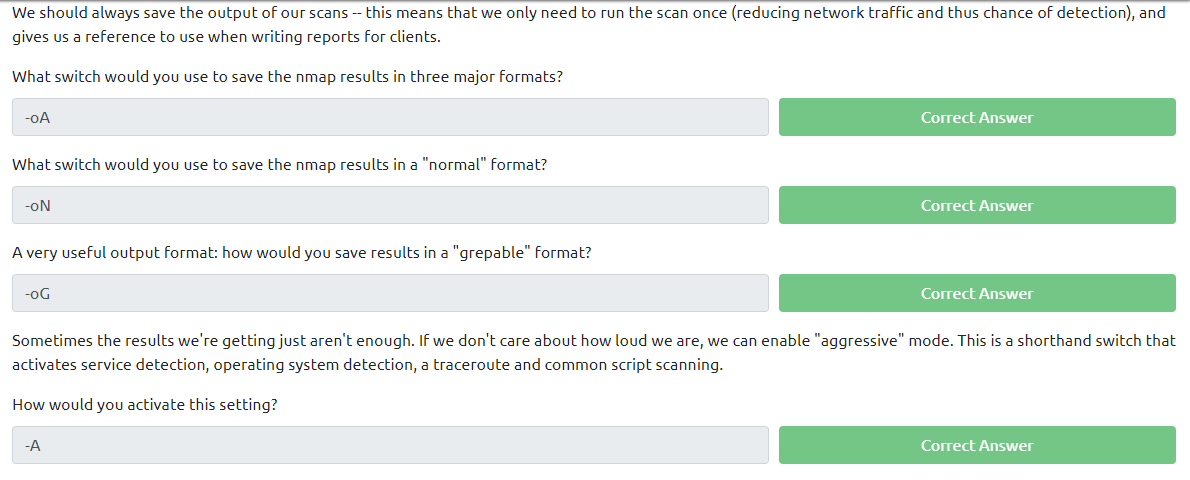
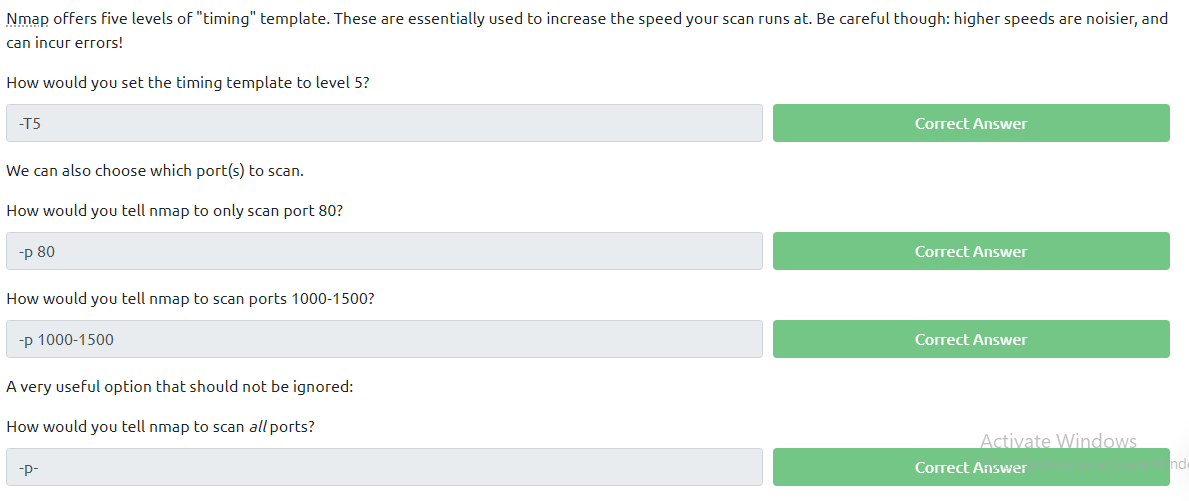
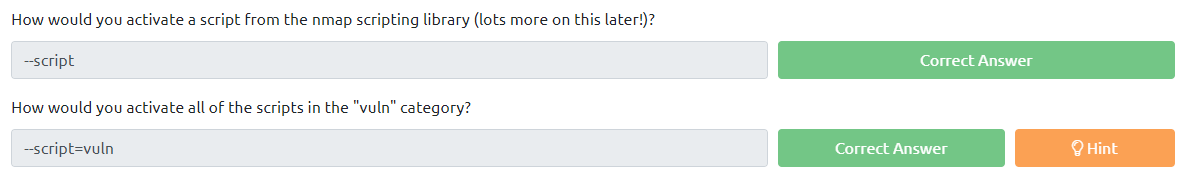
NMAP











When port scanning with Nmap, there are three basic scan types. These are:

* TCP Connect Scans (-sT)
* SYN "Half-open" Scans (-sS)
* UDP Scans (-sU)

Additionally there are several less common port scan types, some of which we will also cover (albeit in less detail). These are:

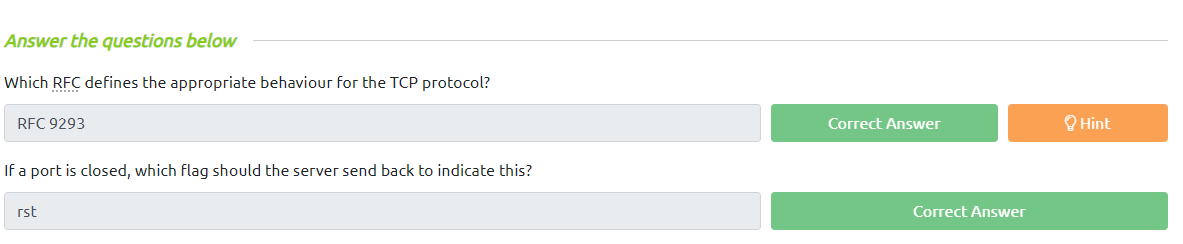
* TCP Null Scans (-sN)
* TCP FIN Scans (-sF)
* TCP Xmas Scans (-sX)

Most of these (with the exception of UDP scans) are used for very similar purposes, however, the way that they work differs between each scan. This means that, whilst one of the first three scans are likely to be your go-to in most situations, it's worth bearing in mind that other scan types exist.

In terms of network scanning, we will also look briefly at ICMP (or "ping") scanning.

Configure a firewall to respond with a RST TCP packet. For example, in IPtables for Linux, a simple version of the command would be as follows:

iptables -I INPUT -p tcp --dport <port> -j REJECT --reject-with tcp-reset



TCP scans, SYN scans (-sS) are used to scan the TCP port-range of a target or targets; however, the two scan types work slightly differently. SYN scans are sometimes referred to as "*Half-open"*scans, or *"Stealth"* scans.

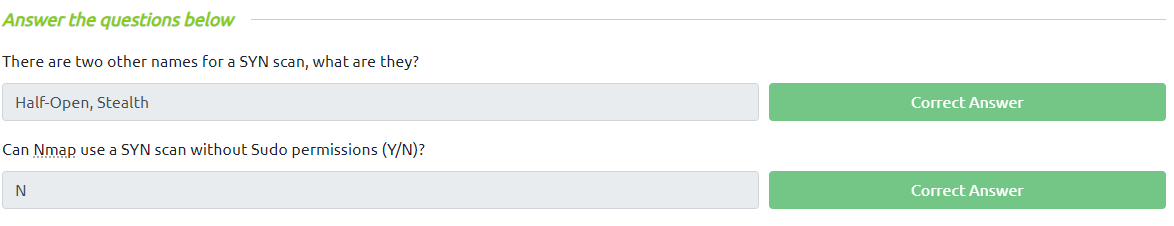
Where TCP scans perform a full three-way handshake with the target, SYN scans sends back a RST TCP packet after receiving a SYN/ACK from the server (this prevents the server from repeatedly trying to make the request).

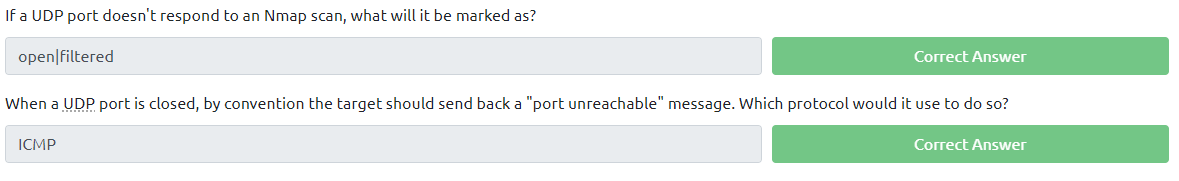
advantages for us as hackers:

* It can be used to bypass older Intrusion Detection systems as they are looking out for a full three way handshake. This is often no longer the case with modern IDS solutions; it is for this reason that SYN scans are still frequently referred to as "stealth" scans.
* SYN scans are often not logged by applications listening on open ports, as standard practice is to log a connection once it's been fully established. Again, this plays into the idea of SYN scans being stealthy.
* Without having to bother about completing (and disconnecting from) a three-way handshake for every port, SYN scans are significantly faster than a standard TCP Connect scan.

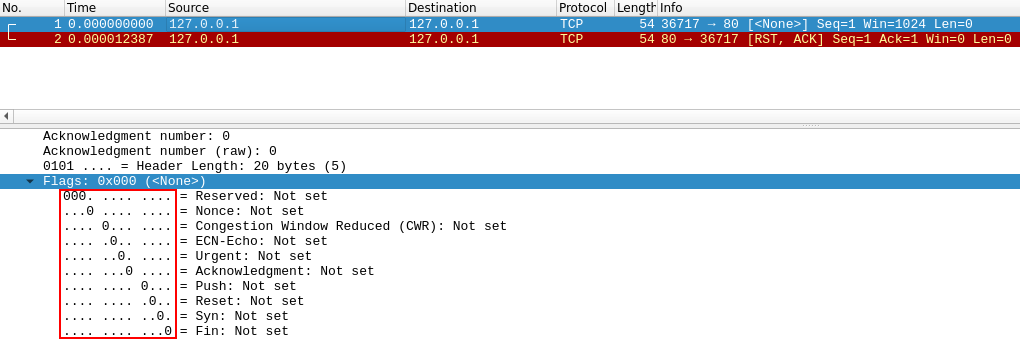
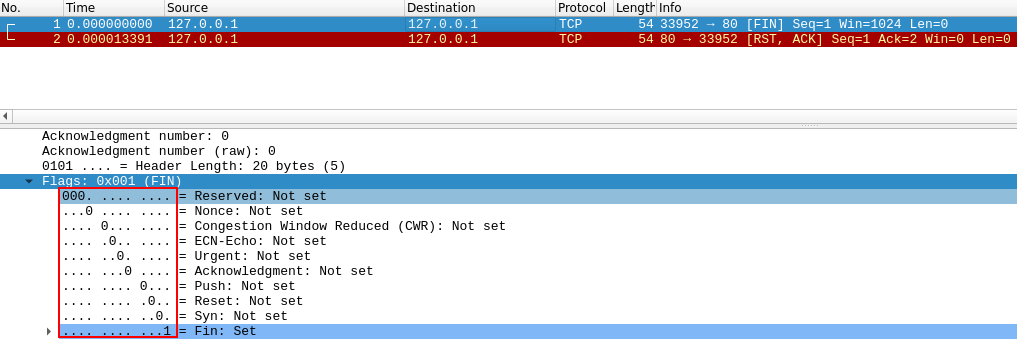
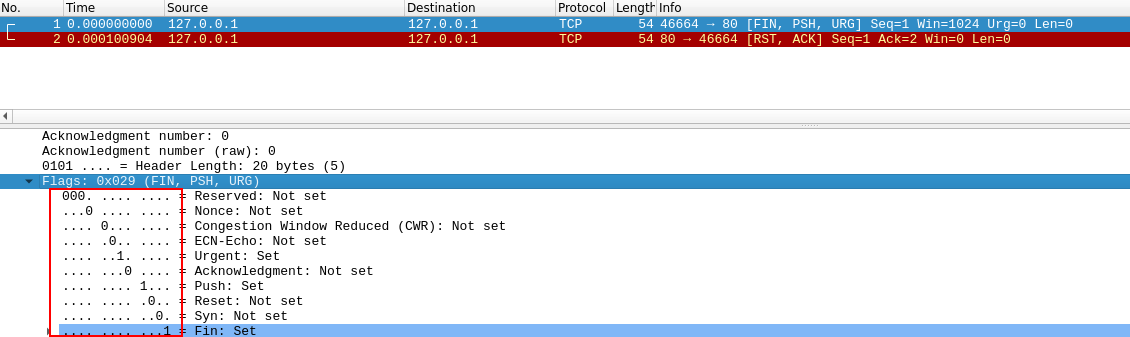
There are, however, a couple of disadvantages to SYN scans, namely:

* They require sudo permissions[1] in order to work correctly in Linux. This is because SYN scans require the ability to create raw packets (as opposed to the full TCP handshake), which is a privilege only the root user has by default.
* Unstable services are sometimes brought down by SYN scans, which could prove problematic if a client has provided a production environment for the test.





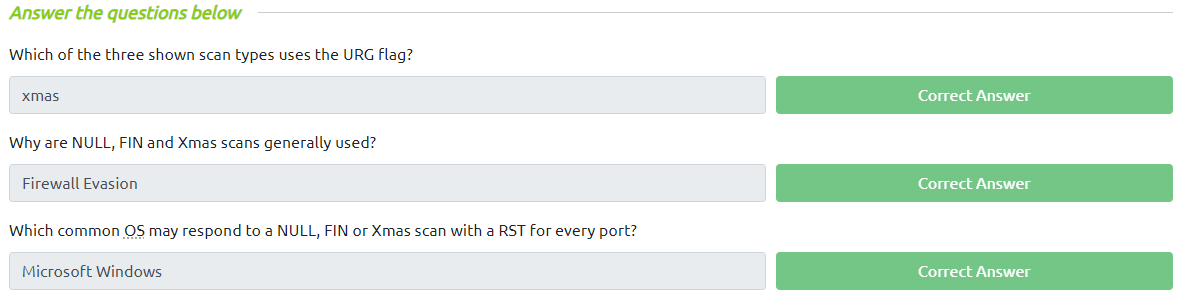
NULL, FIN and Xmas TCP port scans are less commonly used than any of the others we've covered already, so we will not go into a huge amount of depth here. All three are interlinked and are used primarily as they tend to be even stealthier, relatively speaking, than a SYN "stealth" scan. Beginning with NULL scans:

* As the name suggests, NULL scans (-sN) are when the TCP request is sent with no flags set at all. As per the RFC, the target host should respond with a RST if the port is closed.  
  
* FIN scans (-sF) work in an almost identical fashion; however, instead of sending a completely empty packet, a request is sent with the FIN flag (usually used to gracefully close an active connection). Once again, Nmap expects a RST if the port is closed.  
  
* As with the other two scans in this class, Xmas scans (-sX) send a malformed TCP packet and expects a RST response for closed ports. It's referred to as an xmas scan as the flags that it sets (PSH, URG and FIN) give it the appearance of a blinking christmas tree when viewed as a packet capture in Wireshark.  
  

The expected response for *open* ports with these scans is also identical, and is very similar to that of a UDP scan. If the port is open then there is no response to the malformed packet. Unfortunately (as with open UDP ports), that is *also*an expected behaviour if the port is protected by a firewall, so NULL, FIN and Xmas scans will only ever identify ports as being *open|filtered*, *closed*, or *filtered*. If a port is identified as filtered with one of these scans then it is usually because the target has responded with an ICMP unreachable packet.

It's also worth noting that while RFC 793 mandates that network hosts respond to malformed packets with a RST TCP packet for closed ports, and don't respond at all for open ports; this is not always the case in practice. In particular Microsoft Windows (and a lot of Cisco network devices) are known to respond with a RST to any malformed TCP packet -- regardless of whether the port is actually open or not. This results in all ports showing up as being closed.

That said, the goal here is, of course, firewall evasion. Many firewalls are configured to drop incoming TCP packets to blocked ports which have the SYN flag set (thus blocking new connection initiation requests). By sending requests which do not contain the SYN flag, we effectively bypass this kind of firewall. Whilst this is good in theory, most modern IDS solutions are savvy to these scan types, so don't rely on them to be 100% effective when dealing with modern systems.



On first connection to a target network in a black box assignment, our first objective is to obtain a "map" of the network structure -- or, in other words, we want to see which IP addresses contain active hosts, and which do not.

One way to do this is by using Nmap to perform a so called "ping sweep". This is exactly as the name suggests: Nmap sends an ICMP packet to each possible IP address for the specified network. When it receives a response, it marks the IP address that responded as being alive. For reasons we'll see in a later task, this is not always accurate; however, it can provide something of a baseline and thus is worth covering.

To perform a ping sweep, we use the -sn switch in conjunction with IP ranges which can be specified with either a hypen (-) or CIDR notation. i.e. we could scan the 192.168.0.x network using:

* nmap -sn 192.168.0.1-254

or

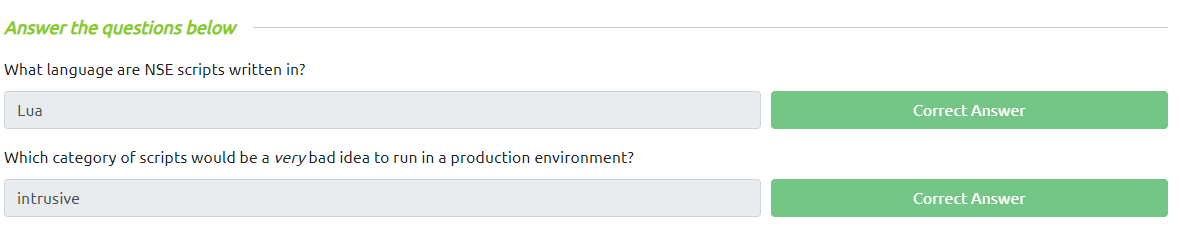
* nmap -sn 192.168.0.0/24

The -sn switch tells Nmap not to scan any ports -- forcing it to rely primarily on ICMP echo packets (or ARP requests on a local network, if run with sudo or directly as the root user) to identify targets. In addition to the ICMP echo requests, the -sn switch will also cause nmap to send a TCP SYN packet to port 443 of the target, as well as a TCP ACK (or TCP SYN if not run as root) packet to port 80 of the target.

The **N**map **S**cripting **E**ngine (NSE) is an incredibly powerful addition to Nmap, extending its functionality quite considerably. NSE Scripts are written in the *Lua*programming language, and can be used to do a variety of things: from scanning for vulnerabilities, to automating exploits for them. The NSE is particularly useful for reconnaisance, however, it is well worth bearing in mind how extensive the script library is.

There are many categories available. Some useful categories include:

* safe:- Won't affect the target
* intrusive:- Not safe: likely to affect the target
* vuln:- Scan for vulnerabilities
* exploit:- Attempt to exploit a vulnerability
* auth:- Attempt to bypass authentication for running services (e.g. Log into an FTP server anonymously)
* brute:- Attempt to bruteforce credentials for running services
* discovery:- Attempt to query running services for further information about the network (e.g. query an SNMP server).



We have already seen some techniques for bypassing firewalls (think stealth scans, along with NULL, FIN and Xmas scans); however, there is another very common firewall configuration which it's imperative we know how to bypass.

Your typical Windows host will, with its default firewall, block all ICMP packets. This presents a problem: not only do we often use *ping* to manually establish the activity of a target, Nmap does the same thing by default. This means that Nmap will register a host with this firewall configuration as dead and not bother scanning it at all.

So, we need a way to get around this configuration. Fortunately Nmap provides an option for this: -Pn, which tells Nmap to not bother pinging the host before scanning it. This means that Nmap will always treat the target host(s) as being alive, effectively bypassing the ICMP block; however, it comes at the price of potentially taking a very long time to complete the scan (if the host really is dead then Nmap will still be checking and double checking every specified port).

It's worth noting that if you're already directly on the local network, Nmap can also use ARP requests to determine host activity.

There are a variety of other switches which Nmap considers useful for firewall evasion. We will not go through these in detail, however, they can be found [here](https://nmap.org/book/man-bypass-firewalls-ids.html).

The following switches are of particular note:

* -f:- Used to fragment the packets (i.e. split them into smaller pieces) making it less likely that the packets will be detected by a firewall or IDS.
* An alternative to -f, but providing more control over the size of the packets: --mtu <number>, accepts a maximum transmission unit size to use for the packets sent. This *must* be a multiple of 8.
* --scan-delay <time>ms:- used to add a delay between packets sent. This is very useful if the network is unstable, but also for evading any time-based firewall/IDS triggers which may be in place.
* --badsum:- this is used to generate in invalid checksum for packets. Any real TCP/IP stack would drop this packet, however, firewalls may potentially respond automatically, without bothering to check the checksum of the packet. As such, this switch can be used to determine the presence of a firewall/IDS.



