

IP Scanning

From the passive reconnaissance, we should now have an IP or range of IP addresses to examine in more details. IP scanning will give us more information about the network possibly including:

- ▶ Services running
- ▶ Operating Systems used
- ▶ Configuration of firewalls/filtering systems

ICMP Types

We are familiar with the basic ICMP messages

- ▶ 8 Echo
- ▶ 13 Timestamp - time in milliseconds since midnight GMT
- ▶ 15 Information request - outdated
- ▶ 17 Subnet mask

Useful tools to manipulate ICMP probing are Sing, Nmap and ICMPScan

Using SING

A basic ping request

```
gavin@dellbot:~$ sing -echo 192.168.1.2
SINGing to 192.168.1.2 (192.168.1.2): 16 data bytes
16 bytes from 192.168.1.2: seq=0 ttl=64 TOS=0 time=64.755 ms
16 bytes from 192.168.1.2: seq=1 ttl=64 TOS=0 time=1.576 ms
```

A timestamp request

```
gavin@dellbot:~$ sing -tstamp 192.168.1.2
SINGing to 192.168.1.2 (192.168.1.2): 20 data bytes
20 bytes from 192.168.1.2: seq=0 ttl=64 TOS=0 diff=8
20 bytes from 192.168.1.2: seq=1 ttl=64 TOS=0 diff=8
```

There was no response by default to a mask request on my linux machine, it has been removed from the 2.6 kernel.

Nmap for a ping scan

Nmap can be used to send an echo request to each machine on a subnet. An alternative tool is ICMPScan

```
gavin@dellbot:~$ sudo nmap -sP -PI 192.168.1.0/24
```

```
Starting Nmap 4.62 ( http://nmap.org ) at 2009-02-18 00:03 GMT
Host 192.168.1.1 appears to be up.
MAC Address: 00:A0:C5:F2:3E:5B (Zyxel Communication)
Host 192.168.1.2 appears to be up.
MAC Address: 00:50:BF:E3:08:91 (Mototech)
Host 192.168.1.33 appears to be up.
MAC Address: 00:0C:6E:91:A0:F0 (Asustek Computer)
Host 192.168.1.34 appears to be up.
MAC Address: 00:15:AF:66:F7:CA (AzureWave Technologies)
Host 192.168.1.36 appears to be up.
MAC Address: 00:04:20:05:09:5B (Slim Devices)
Host 192.168.1.42 appears to be up.
Host 192.168.1.45 appears to be up.
MAC Address: 00:0C:76:24:0E:DF (Micro-star International CO.)
Nmap done: 256 IP addresses (7 hosts up) scanned in 3.035 seconds
```

Finding Subnets

- ▶ If you recall, when a broadcast address is pinged, all nodes in that subnet will respond.
- ▶ We can use this to find subnets within an IP range
- ▶ Ping a network range using nmap to iterate through each IP address
- ▶ Any address that returns more than one ping is a subnet broadcast address
- ▶ As above, `nmap -sP range/mask`

Finding private IP addresses

- ▶ If a ping scan is sent to a public IP, that host may be using Network Address Translation and passing packets to an internal machine
- ▶ This internal machine may respond to the ping request. However, tools like Nmap or SING will not recognise these responses as they will have an unexpected source address
- ▶ One solution is to run a stateful firewall. This firewall will display any incoming connections, so pickup these rogue responses.
- ▶ ICMPScan will also pick up the responses.
- ▶ This can be a useful way to discover internal IP addresses.

OS Probing

```

gavin@dellbot:~$ sudo xprobe2 192.168.1.2
[..]
[+] Following modules are loaded:
[x] [1] ping:icmp_ping - ICMP echo discovery module
[x] [2] ping:tcp_ping - TCP-based ping discovery module
[x] [3] ping:udp_ping - UDP-based ping discovery module
[x] [4] infogather:ttl_calc - TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan - TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo - ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp - ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask - ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_port_unreach - ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake - TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst - TCP RST fingerprinting module
[x] [12] fingerprint:smb - SMB fingerprinting module
[x] [13] fingerprint:snmp - SNMPv2c fingerprinting module
[..]
[-] No distance calculation. 192.168.1.2 appears to be dead or no ports known
[+] Host: 192.168.1.2 is up (Guess probability: 50%)
[+] Target: 192.168.1.2 is alive. Round-Trip Time: 0.00155 sec
[+] Selected safe Round-Trip Time value is: 0.00311 sec
[..]
[+] Primary guess:
[+] Host 192.168.1.2 Running OS: "Linux Kernel 2.4.22" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.1.2 Running OS: "Linux Kernel 2.4.23" (Guess probability: 100%)
[..]
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
gavin@dellbot:~$ ssh 192.168.1.2 uname -a
Linux stinger 2.6.20-17-generic #2 SMP Wed Aug 20 16:47:34 UTC 2008 i686 GNU/Linux

```

TCP Connection scanning

A TCP port scan is a technique for locating TCP services running on a machine. It is a brute force method of attempting to connect to each port on a machine.

Realistically, most port scanners will not connect to all ports by default, but rather a list of commonly used ones. There are three main approaches to port scanning

- ▶ Standard connections
- ▶ Stealthy connections
- ▶ Third party/Spoofed connection

Standard TCP Connections

The most straightforward way to determine if a service is running is to simply connect to the server with a standard TCP/IP three way handshake.

- ▶ If the service is running, the connection will be established via a SYN, SYN/ACK and ACK.
- ▶ If there is no service running on the port, a RST packet will be sent in response to the SYN.
- ▶ If there is no response at all, it implies that the port is being filtered
- ▶ This method is reliable, but very obvious. Full connections will be logged by either the application or possible a firewall.
- ▶ `nmap -sT` and an IP range will perform this scan

Standard TCP Connections

A modification to the standard connection attempt is to send a SYN packet, receive the SYN/ACK response and reply with a RST packet, cancelling the connection. This is half open Syn scanning.

- ▶ It is more discreet than a standard connection as no full connection is made. The application will probably not log it
- ▶ It can be faster, less packets are sent and received.
- ▶ It does require access to Raw sockets, so admin access on Linux and Windows
- ▶ Most firewalls and Intrusion Detection Systems should detect this however. item Nmap will do this with the -sS flag. Another very fast tool is Scanrand from Dan Kaminsky

Stealthy TCP Connections

CA645

Gavin O' Gorman

IP Scanning

TCP Probing

UDP Probing

Firewalls

Intrusion Detection
Systems

IDS Evasion

Stealthy TCP connections are not as reliable as a standard TCP connection, however they can be used to avoid detection by firewalls and IDSs.

- ▶ Inverse TCP flag scanning
- ▶ Ack flag probe scanning

Inverse TCP Scanning

Inverse scanning is looking for responses from closed ports and no responses from open ports. By sending a malformed packet to a service, the TCP/IP stack will ignore it.

However, RFC793 states that if a packet is sent to a closed port, a RST/ACK packet is sent in response. So we need to generate invalid packets that a valid service will ignore.

These could be:

- ▶ A FIN probe - FIN TCP flag set
- ▶ An XMAS probe - FIN, URG and PUSH bits all set
- ▶ A Null probe - No flags set

Any closed port should respond with the RST/ACK packet. However, Windows OSs do not implement this part of the RFC, so the scan is only effective on some Unix OSs.

ACK Probing

TCP ACK probing is not actually used to identify open ports on a host, but rather if the ports are filtered.

- ▶ A TCP packet with the ACK bit set is sent to the host machine
- ▶ Both open and closed ports will respond with a RST packet
- ▶ If there is no response at all, or an ICMP message is sent, then the port is being filtered.

An extension to this probe is to monitor the TCP Window value set in any RST packets that are sent back. Some TCP/IP implementations will have a positive window size set, if the port is open and a 0 value if the port is closed. So it can be used to both check filtering and possibly discover open ports.

Spoofed connections

It would be nice if we could mask our IP address somehow when performing a scan, to help prevent identification. If a false IP is spoofed, obviously we can't see the response messages. We could spoof the IP of a machine on the same LAN perhaps and use ARP poisoning to monitor the responses, but that is still quite close to home. There are three possible solutions, one straightforward, one outdated and the other, ingenious !

- ▶ Decoy scan
- ▶ FTP Bounce
- ▶ TCP Idle scan

[IP Scanning](#)[TCP Probing](#)[UDP Probing](#)[Firewalls](#)[Intrusion Detection Systems](#)[IDS Evasion](#)

Decoy scan

This is quite straightforward. At the same time as you send your scan, spoof scans from a number of other IP addresses. The person being scanned will not know which is the real scanner

- ▶ Obviously, the other IPs chosen must be relatively intelligent
- ▶ The attacker must be careful to ensure that DNS lookups are done by all fake IPs and avoid leaking any other information
- ▶ The person being scanned can use tricks to determine who is scanning them, if the scan is live

FTP Bounce

This takes advantage of an element in the FTP protocol which allows one to connect into a FTP server and then request a transfer to a third party FTP server. It effectively means that someone can open a connection to any machine/port they want through a FTP server.

- ▶ So the attacker requests a file transfer to each port on a remote machine.
- ▶ The FTP server will attempt to connect and provide an error message. The error message will describe if the port is open or not.
- ▶ This is a fairly old attack and very few FTP servers will still offer this feature by default.

TCP Idle scan

This is the really good one. It allows an attacker to port scan a remote machine, without the remote machine seeing any packets from the attacker's real IP address. It is based on three facts

- ▶ When a SYN packet is sent to an open port it will respond with a SYN/ACK if open and a RST if closed
- ▶ A machine that receives an unsolicited SYN/ACK will respond with a RST and an unsolicited RST is ignored.
- ▶ Every IP packet has a fragmentation ID number (remember the fragmenting we looked at for IP). Many operating systems simply increment this ID value. Examining the IP ID value can tell you how many packets have been sent.

TCP Idle scan

It's a three step process involving three machines. The attacker, the victim and a zombie machine.

- ▶ Attacker sends a SYN/ACK to the zombie machine. The zombie responds with a RST packet. This RST packet contains an IP ID.
- ▶ Spoof a connection to the victim machine from the zombie machine. The victim will respond to the zombie machine with a SYN/ACK if the port is open, or a RST if the port is closed. The zombie will then respond to this SYN/ACK with a RST packet, or do nothing if it's a RST packet.
- ▶ Probe the zombie machine again for it's IP ID value. If the value has gone up by two, then we know the port on the victim is open. If the IP ID is gone up by one, then the port is either closed or filtered.

Idle scan example

Example take from the Nmap site.

```
# nmap -PN -p- -sI kiosk.adobe.com www.riaa.com
Starting Nmap ( http://nmap.org )
Idlescan using zombie kiosk.adobe.com (192.150.13.111:80); Class: Incremental
Interesting ports on 208.225.90.120:
(The 65522 ports scanned but not shown below are in state: closed)
Port      State      Service
21/tcp    open       ftp
25/tcp    open       smtp
80/tcp    open       http
111/tcp   open       sunrpc
135/tcp   open       loc-srv
443/tcp   open       https
1027/tcp  open       IIS
1030/tcp  open       iad1
2306/tcp  open       unknown
5631/tcp  open       pcananywheredata
7937/tcp  open       unknown
7938/tcp  open       unknown
36890/tcp open       unknown

Nmap done: 1 IP address (1 host up) scanned in 2594.47 seconds
```

The zombie machine needs to be chosen carefully

- ▶ The machine can't be active, or the IP ID values will increase
- ▶ The machine must be able to receive packets from the victim or it will not be able to respond
- ▶ and similarly, it must not have filtering rules preventing it from responding to the victim
- ▶ This can be an effective technique for determining potential trust relationships between IP addresses

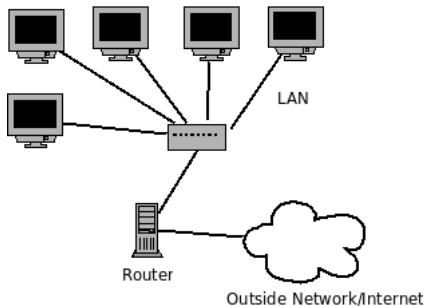
UDP Scanning

UDP scanning is not as straightforward as TCP. UDP is a connectionless protocol, there is no handshake process to validate a connections. So, scanning for a UDP service is done in two ways

- ▶ Send UDP datagram to all ports on a machine. Wait for ICMP port unreachable messages. Any port that doesn't send back an ICMP message may be running a service. This may not work, depending on how much ICMP is allowed through a firewall. Nmap can use this approach
- ▶ Use a UDP application and simply attempt to use a service on the remote machine. `scanudp` effectively does this by sending datagrams designed to look like valid clients

Routing

We will have a very brief overview of firewalls and Intrusion detection systems. A typical basic network might look at below



- ▶ This basic network has an assigned IP block, so publicly routable IP addresses
- ▶ Incoming and outgoing connections are via the multi-homed lan router/gateway. The gateway simply routes data
- ▶ So each machine on the network has complete access to outside networks, lets call it the Internet
- ▶ And of course, anything on the Internet can connect to these machines

This is how the Internet was envisaged, everything can connect to everything.

Basic Network

However. Problems start to crop up. As more application level protocols are developed, more services are being run on LAN machine. File sharing for example, NFS. IP and TCP stream spoofing starts to become a problem with the r-utilities. The Mitnick hack for example. So it becomes necessary to start restricting access.

- ▶ On the private LAN, we don't want anybody to be able to connect to the file sharing services, so we tell the router to block any traffic coming into the network from outside, to the NFS port, or the rpc port 111.
- ▶ The router will have an Access Control List. This list will have in it something of the form `deny tcp any 111` which is applied to the external interface.
- ▶ Such restrictions can also be used to restrict access within the LAN to outside. For example, refuse all outgoing connections to port 80 to disallow web access.

Rudimentary firewall

Any device which limit access is a firewall. Some trivia. Firewalls were originally brick walls built between apartment buildings to stop fires from spreading. The term is slightly confused. Firewalls now tend to mean dedicated stateful packet filters

- ▶ The previous example of using router ACLs as a firewall is a primitive approach. As we will see, such stateless rules can allow scans to enter the network from a specific source port. The example being that the router must create a permit rule allowing traffic from external port 53 into the network, so allow for DNS lookups.
- ▶ A better type of firewall is needed.

Stateful firewall

This better type of firewall is a stateful one. The firewall is more intelligent and understands how TCP/IP connections work.

- ▶ Assume that the firewall allows outgoing connections from the internal network.
- ▶ A local machine makes a request for a DNS lookup
- ▶ The stateful firewall sees the request, allows it through and at the same time updates a local state table with this information
- ▶ When the response from the DNS server arrives, the firewall looks in the state table, see that there was in fact a request for this packet, so allows it through.
- ▶ If a packet with source port 53 arrives at random, it is refused.

Stateful firewall

This better type of firewall is a stateful one. The firewall is more intelligent and understands how TCP/IP connections work.

- ▶ Assume that the firewall allows outgoing connections from the internal network.
- ▶ A local machine makes a request for a DNS lookup
- ▶ The stateful firewall sees the request, allows it through and at the same time updates a local state table with this information
- ▶ When the response from the DNS server arrives, the firewall looks in the state table, see that there was in fact a request for this packet, so allows it through.
- ▶ If a packet with source port 53 arrives at random, it is refused.

The application layer

An extra layer of security can be obtained with application level proxies

- ▶ Application layer proxies understand the application they work with and offer more control over the traffic.
- ▶ Squid is an example of a HTTP proxy which performs caching, and can also handle FTP, TLS, HTTPS.
- ▶ So rather than leave an outgoing hole of port 80 in the firewall, which could be abused, insert a HTTP proxy.
- ▶ A useful technique for achieving this subtly is transparent redirection.

Just a brief word on NAT. As home networks became common and the available IPv4 pool decreases, Network Address Translation has become more popular.

- ▶ LANS now tend to use private IP range, so basic routing will no longer work
- ▶ NAT is a technique where a single, publicly routable, IP address can be shared.
- ▶ It is a feature of most home gateway/AP/router devices. Windows implements it using its Internet Connection Sharing and Linux also offers it
- ▶ It effectively functions as a rudimentary stateful firewall
- ▶ Is it completely secure however?

Intrusion Detection Systems

A firewall works to block traffic and implement a policy. It is not designed to keep us informed of the status of a network, or detect attacks. An intrusion detection system can do this. It complements a firewall, it doesn't replace it. There are host, network and stack based IDSs

- ▶ Host based works by analysing application level logs on a specific host. Tripwire would be a rudimentary example
- ▶ Network based analyses raw packets on the wire
- ▶ Stack based is implemented in an actual network stack implementation and can allow or disallow packets from ascending up the stack.

- ▶ A network IDS consists of several components, sensors, a console and a processing engine. The console and engine tend to be located on the same device and one or more sensors are located around the network.
- ▶ Low cost. It's very easy to implement
- ▶ Low level packet analysis
- ▶ Evidence removal
- ▶ Realtime detection and response
- ▶ Malicious intent detection - extra information from outside the firewall for example

A popular opensource IDS is Snort.

- ▶ Snort is basically an advanced packet sniffer, like tcpdump, but able to examine packet contents
- ▶ *Snort decodes the application layer of a packet and can be given rules to collect traffic that has specific data contained within its application layer. This allows Snort to detect many types of hostile activity, including buffer overflows, CGI scans, or any other data in the packet payload that can be characterized in a unique detection fingerprint.*

There are several techniques possible to attempt evasion of Intrusion Detection Systems and packet filters (firewalls).

The main ones are:

- ▶ Fragmentation
- ▶ Spoofing
- ▶ Source routing
- ▶ Using specific source ports

IP Fragmentation

- ▶ There are two applications, fragroute and fragtest
- ▶ Fragtest basically tests to see if the target host will respond to heavily fragmented packets by sending test ICMP packets fragments. The target host obviously must be configured to respond to ICMP in the first instance
 - ▶ Tests are performed using fragmentation of 8 bytes
 - ▶ 8 bytes with 16byte overlap and use new data in re-construction
 - ▶ 8 bytes with 16byte overlap and use old data in re-construction
- ▶ Once fragtest confirms that one of the technique, or all of them works, fragroute can be used
- ▶ Fragroute will fragment all outgoing traffic to a particular IP
- ▶ Fragroute can perform additional segmentation at the TCP level

We've looked at source routing already, loose and strict routing. It can be a useful technique for spoofing packets from a trusted source.

- ▶ LRScan can be used to send source routed packets to hosts and check if the host will respond and if it does respond, do it use the reverse route
- ▶ Once it's verified that the target router will utilise source routing, LSRTunnel can be used to send traffic using source routing

Using source ports to bypass filtering

Packet filters/firewalls often have 'holes' in them to allow traffic through. For example, a firewall may allow internal machines to issues DNS requests to an external DNS server and so it will allow external incoming connections from source port 53, TCP and UDP. As such, it's useful when scanning an IP range through a firewall, to test different source ports. These ports might be

- ▶ TCP or UDP 53 - DNS
- ▶ TCP 20 - FTP data
- ▶ TCP Port 80 - HTTP
- ▶ TCP or UDP 888 - Kerberos

This won't work with stateful firewalls.

- ▶ Filter ICMP carefully.
- ▶ Filter outbound type 3 'unreachable messages'
- ▶ Consider using firewalls to throttle port scan attempts
This may not actually be a great idea, it leaves one open to a very easy Denial of Service attack
- ▶ Test the ability of your firewalls/gateways to handle heavily fragmented packets
- ▶ Inspect firewall rules to ensure that using specific source ports will not bypass them