Subverting Intrusion Detection Systems

> This page is a direct copy/paste of this Nmap documentation
>
> > My favorite section is DNS proxying

Firewalls are not the only obstacle that modern attackers face. Intrusion detection and prevention systems can be problematic as well. Network administration staff do not always take well to a flood of 2:00 A.M. intrusion alert pages from the IDS. Considerate hackers take pains to prevent their actions from causing all of these alerts in the first place. A first step is to detect whether an IDS is even present—many small companies do not use them. If an IDS is suspected or detected, there are many effective techniques for subverting it. They fall into three categories that vary by intrusiveness: avoiding the IDS as if the attacker is not there, confusing the IDS with misleading data, and exploiting the IDS to gain further network privilege or just to shut it down. Alternatively, attackers who are not concerned with stealth can ignore the IDS completely as they pound away at the target network.

Intrusion Detection System Detection

Early on in the never-ending battle between network administrators and malicious hackers, administrators defended their turf by hardening systems and even installing firewalls to act as a perimeter barrier. Hackers developed new tools to penetrate or sneak around the firewalls and exploit vulnerable hosts. The arms race escalated with administrators introducing intrusion detection systems that constantly watch for devious activity. Attackers responded, of course, by devising systems for detecting and deceiving the IDS. While intrusion detection systems are meant to be passive devices, many can be detected by attackers over the network.

The least conspicuous IDS is one that passively listens to network traffic without ever transmitting. Special network tap hardware devices are available to ensure that the IDS *cannot* transmit, even if it is compromised by attackers. Despite the security advantages of such a setup, it is not widely deployed due to practical considerations. Modern IDSs expect to be able send alerts to central management consoles and the like. If this was all the IDS transmitted, the risk would be minimal. But to provide more extensive data on the alert, they often initiate probes that may be seen by attackers.

Reverse probes

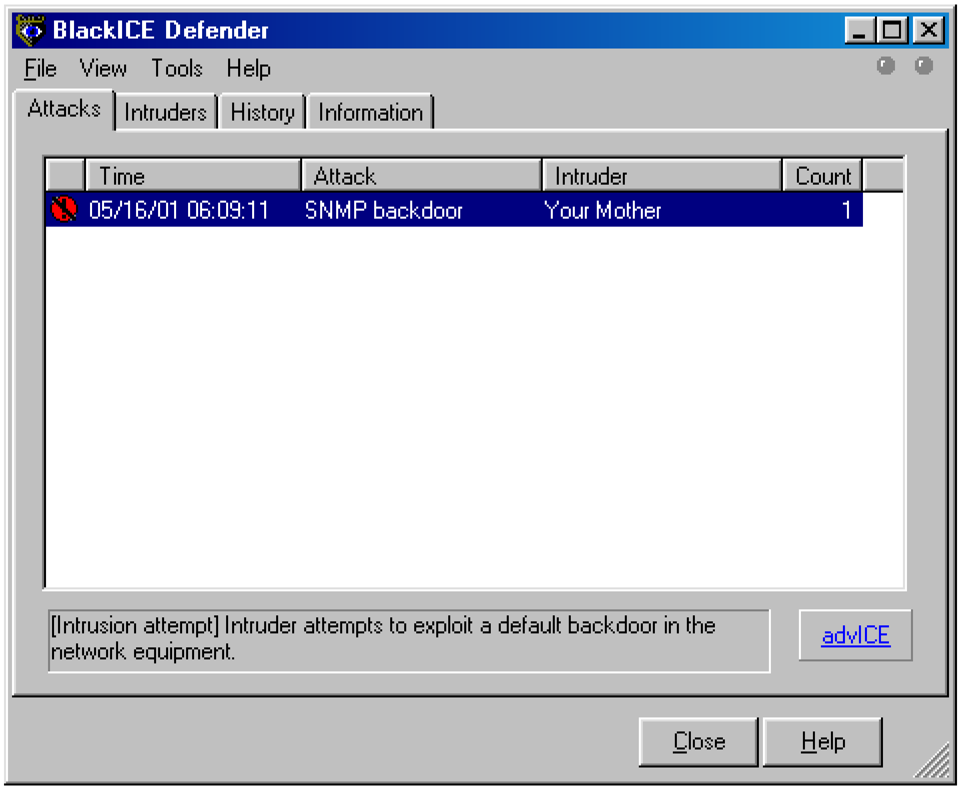
One probe commonly initiated by IDSs is reverse DNS query of the attacker's IP address. A domain name in an alert is more valuable than just an IP address, after all. Unfortunately, attackers who control their own rDNS (quite common) can watch the logs in real time and learn that they have been detected. This is a good time for attackers to feed misinformation, such as bogus names and cache entries to the requesting IDS.

Some IDSs go much further and send more intrusive probes to the apparent attackers. When an attacker sees his target scan him back, there is no question that he has set off alarms. Some IDSs send Windows NetBIOS information requests back to the attacker. ISS BlackICE Defender is one vendor that does (or at least did) this by default. I wrote a small tool called icepick which sends a simple packet that generates an alert from listening BlackICE instances. Then it watches for telltale NetBIOS queries and reports any

BlackICE installations found. One could easily scan large networks looking for this IDS and then attempt to exploit them using holes discussed later in this chapter.

Not content with simply locating BlackICE installations or detecting them during penetration tests, I wrote a simple Unix program called windentd which replies to the probe with misinformation. Figure 10.1 shows a BlackICE console where the Intruder is listed as “Your Mother” thanks to windentd and icepick. Those simple tools are available from `https://nmap.org/presentations/CanSecWest01/` , though they are not supported.

Figure 10.1. BlackICE discovers an unusual intruder



Time	Attack	Intruder	Count
05/16/01 06:09:11	SNMP backdoor	Your Mother	1

[Intrusion attempt] Intruder attempts to exploit a default backdoor in the network equipment.

Sudden firewall changes and suspicious packets

Many intrusion detection systems have morphed into what marketing departments label intrusion prevention systems. Some can only sniff the network like a normal IDS and send triggered packet responses. The best IPS systems are inline on the network so that they can restrict packet flow when suspicious activity is detected. For example, an IPS may block any further traffic from an IP address that they believe has port scanned them, or that has attempted a buffer overflow exploit. Attackers are likely to notice this if they port scan a system, then are unable to connect to the reported open ports. Attackers can confirm that they are blocked by trying to connect from another IP address.

Suspicious response packets can also be a tip-off that an attacker's actions have been flagged by an IDS. In particular, many IDSs that are *not* inline on the network will forge RST packets in an attempt to tear down connections. Ways to determine that these packets are forged are covered in the section called “Detecting Packet Forgery by Firewall and Intrusion Detection Systems”.

Naming conventions

Naming conventions can be another giveaway of IDS presence. If an Nmap list scan returns host names such as realsecure, ids-monitor, or dragon-ids, you may have found an intrusion detection system. The administrators might have given away that information inadvertently, or they may think of it like the alarm stickers on house and car windows. Perhaps they think that the script kiddies will be scared away by IDS-related names. It could also be misinformation. You can never fully trust DNS names. For example, you might assume that bugzilla.securityfocus.com is a web server running the popular Bugzilla web-based bug tracking software. Not so. The Nmap scan in Example 10.16 shows that it is probably a Symantec Raptor firewall instead. No web server is accessible, though there may be one hidden behind the Raptor.

Example 10.16. Host names can be deceiving

```
nmap -sS -sV -T4 -p1-24 bugzilla.securityfocus.com
```

```
Starting Nmap ( https://nmap.org )
Nmap scan report for 205.206.231.82
Not shown: 21 closed ports
PORT STATE SERVICE VERSION
21/tcp open ftp-proxy Symantec Enterprise Firewall FTP proxy
22/tcp open ssh?
23/tcp open telnet Symantec Raptor firewall secure gateway telnetd

Nmap done: 1 IP address (1 host up) scanned in 0.94 seconds
```

Unexplained TTL jumps

One more way to detect certain IDSs is to watch for unexplained gaps (or suspicious machines) in traceroutes. While most operating systems include a **traceroute** command (it is abbreviated to **tracert** on Windows), Nmap offers a faster and more effective alternative with the `--traceroute` option. Unlike standard **traceroute**, Nmap sends its probes in parallel and is able to determine what sort of probe will be most effective based on scan results. In Example 10.17, which was contrived for simplicity, traceroute locates nothing at hop five. That may be an inline IDS or firewall protecting the target company. Of course, this can only detect inline IDSs as opposed to those which passively sniff the network without being part of the route. Even some inline devices may not be spotted because they fail to decrement the TTL or refuse to pass ICMP ttl-exceeded messages back from the protected network.

Example 10.17. Noting TTL gaps with traceroute

```
nmap --traceroute www.target.com
```

```
Interesting ports on orestes.red.target.com (10.0.0.6)
Not shown: 996 filtered ports
PORT STATE SERVICE
22/tcp open ssh
53/tcp open domain
80/tcp open http
113/tcp closed auth

TRACEROUTE (using port 22/tcp)
HOP RTT ADDRESS
1 1.10 gw (205.217.153.49)
```

```
2 10.40 metro1-ge-152.pa.meer.net (205.217.152.1)
3 12.02 208.185.168.171 (208.185.168.171)
4 14.74 p4-2-0-0.r06.us.bb.verio.net (129.250.9.129)
5 ...
6 15.07 orestes.red.target.com (10.0.0.6)

Nmap done: 1 IP address (1 host up) scanned in 4.35 seconds
```

While traceroute is the best-known method for obtaining this information, it isn't the only one. IPv4 offers an obscure option called record route for gathering this information. Due to the maximum IP header size, a maximum of nine hops can be recorded. In addition, some hosts and routers drop packets with this option set. It is still a handy trick for those times when traditional traceroute fails. This option can be specified with Nmap using `--ip-options R` to set the option and `--packet-trace` to read it from the response. It is generally used in conjunction with an ICMP ping scan (`-sn -PE`). Most operating systems offer an `-R` option to their ping command, which is easier to use than Nmap for this purpose. An example of this technique is provided in Example 10.18.

Example 10.18. Using the IP record route option

```
ping -R 151.164.184.68
PING 151.164.184.68 (151.164.184.68) 56(124) bytes of data.
64 bytes from 151.164.184.68: icmp_seq=1 ttl=126 time=11.7 ms
NOP
RR: 192.168.0.100
    69.232.194.10
    192.168.0.6
    192.168.0.100

--- 151.164.184.68 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 11.765/11.765/11.765/0.000 ms
```

Avoiding Intrusion Detection Systems

The most subtle way to defeat intrusion detection systems is to avoid their watchful gaze entirely. The reality is that rules governing IDSs are pretty brittle in that they can often be defeated by manipulating the attack slightly. Attackers have dozens of techniques, from URL encoding to polymorphic shellcode generators for escaping IDS detection of their exploits. This section focuses on stealthy port scanning, which is even easier than stealthily exploiting vulnerabilities.

Slow down

When it comes to avoiding IDS alerts, patience is a virtue. Port scan detection is usually threshold based. The system watches for a given number of probes in a certain timeframe. This helps prevent false positives from innocent users. It is also essential to save resources—saving connection probes forever would consume memory and make real-time list searching too slow. The downside to this threshold approach is that attackers can evade it by keeping their scan rate just below the threshold. Nmap offers several canned timing modes that can be selected with the `-T` option to accomplish this. For example, the `-T paranoid` option causes Nmap to send just one probe at a time, waiting five minutes between them. A large scan may take weeks, but at least it probably will not be detected. The `-T sneaky` option is similar, but it only waits 15 seconds between probes.

Rather than specify canned timing modes such as `sneaky`, timing variables can be customized precisely with options such as `--max-parallelism`, `--min-rtt-timeout`, and `--scan-delay`. Chapter 6, *Optimizing Nmap Performance* describes these in depth.

A practical example: bypassing default Snort 2.2.0 rules

Examining the handy open-source Snort IDS provides a lesson on sneaking under the radar. Snort has had several generations of port scan detectors. The Flow-Portscan module is quite formidable. A scan that slips by this is likely to escape detection by many other IDSs as well.

Flow-portscan is made up of two detection systems that can work in concert (or be enabled individually) to detect port scanners. The system and its dozens of configuration variables are documented in `docs/README.flow-portscan` in the Snort distribution, but I'll provide a quick summary.

The simpler detection method in Flow-portscan is known as the *fixed time scale*. This simply watches for `scanner-fixed-threshold` probe packets in `scanner-fixed-window` seconds. Those two variables, which are set in `snort.conf`, each default to 15. Note that the counter includes any probes sent from a single machine to any host on the protected network. So quickly scanning a single port on each of 15 protected machines will generate an alert just as surely as scanning 15 ports on a single machine.

If this were the only detection method, the solution would be pretty easy. Pass the `--scan-delay 1075ms` option to ensure that Nmap waits 1.075 seconds between sending probes. The intuitive choice might be a one second wait between packets to avoid 15 packets in 15 seconds, but that is not enough. There are only 14 waits between sending the first packet and the fifteenth, so the wait must be at least 15/14, or 1.07143 seconds. Some poor sap who chooses `--scan-delay 1000ms` would slow the scan down dramatically, while still triggering the alarm. If multiple hosts on the network are being probed, they must be scanned separately to avoid triggering the alarm. The option `--max-hostgroup 1` would ensure that only one host at a time is scanned, but is not completely safe because it will not enforce the `--scan-delay` between the last probe sent to one host, and the first sent to the next. As long as at least 15 ports per host are being scanned, you could compensate by making the `--scan-delay` at least 1155 ms, or simply start single-target Nmap instances from a shell script, waiting 1075 ms between them. Example 10.19 shows such a stealthy scan of several machines on a network. Multiple Nmap instances are handled using the Bash shell syntax. Here the IPs are specified manually. If many targets were desired, they could be enumerated into a file with the `-sL` (list scan) option, then Nmap started against each using a normal shell loop. The reason these scans took more than 1.075 seconds per port is that retransmissions were required for the filtered ports to ensure that they were not dropped due to network congestion.

Example 10.19. Slow scan to bypass the default Snort 2.2.0 Flow-portscan fixed time scan detection method

```
felix~# for target in 205.217.153.53 205.217.153.54 205.217.153.62; \ do nmap --scan-delay 1075ms -p21,22,23,25,53 $target; \ usleep 1075000; \ done

Starting Nmap ( https://nmap.org )
Nmap scan report for insecure.org (205.217.153.53)
```

```
PORT STATE SERVICE
21/tcp filtered ftp
22/tcp open ssh
23/tcp filtered telnet
25/tcp open smtp
53/tcp open domain

Nmap done: 1 IP address (1 host up) scanned in 10.75 seconds

Starting Nmap ( https://nmap.org )
Nmap scan report for lists.insecure.org (205.217.153.54)
PORT STATE SERVICE
21/tcp filtered ftp
22/tcp open ssh
23/tcp filtered telnet
25/tcp open smtp
53/tcp open domain

Nmap done: 1 IP address (1 host up) scanned in 10.78 seconds

Starting Nmap ( https://nmap.org )
Nmap scan report for scanme.nmap.org (205.217.153.62)
PORT STATE SERVICE
21/tcp filtered ftp
22/tcp open ssh
23/tcp filtered telnet
25/tcp open smtp
53/tcp open domain

Nmap done: 1 IP address (1 host up) scanned in 10.80 seconds
```

Unfortunately for port scanning enthusiasts, defeating Snort is not so simple. It has another detection method, known as *sliding time scale*. This method is similar to the fixed-window method just discussed, except that it increases the window whenever a new probe from a host is detected. An alarm is raised if `scanner-sliding-threshold` probes are detected during the window. The window starts at `scanner-sliding-window` seconds, and increases for each probe detected by the amount of time elapsed so far in the window times `scanner-sliding-scale-factor`. Those three variables default to 40 probes, 20 seconds, and a factor of 0.5 in `snort.conf`.

The sliding scale is rather insidious in the way it grows continually as new packets come in. The simplest (if slow) solution would be to send one probe every 20.1 seconds. This would evade both the default fixed and sliding scales. This could be done just as in Example 10.19, but using a higher value. You could speed this up by an order of magnitude by sending 14 packets really fast, waiting 20 seconds for the window to expire, then repeating with another 14 probes. You may be able to do this with a shell script controlling Nmap, but writing your own simple SYN scanning program for this custom job may be preferable.

Scatter probes across networks rather than scanning hosts consecutively

As discussed in the previous section, IDSs are often programmed to alarm only after a threshold of suspicious activity has been reached. This threshold is often global, applying to the whole network protected by the IDS rather than just a single host. Occasionally they specifically watch for traffic from a given source address to consecutive hosts. If a host sends a SYN packet to port 139 of host 10.0.0.1, that isn't too suspicious by itself. But if that probe is followed by similar packets to 10.0.0.2, .3, .4, and .5, a port scan is clearly indicated.

One way to avoid triggering these alarms is to scatter probes among a large number of hosts rather than scanning them consecutively. Sometimes you can avoid scanning very many hosts on the same network. If you are only conducting a research survey, consider scattering probes across the whole Internet with `-iR` rather than scanning one large network. The results are likely to be more representative anyway.

In most cases, you want to scan a particular network and Internet-wide sampling isn't enough. Avoiding the consecutive-host probe alarms is easy. Nmap offers the `--randomize-hosts` option which splits up the target networks into blocks of 16384 IPs, then randomizes the hosts in each block. If you are scanning a huge network, such as class B or larger, you may get better (more stealthy) results by randomizing larger blocks. You can achieve this by increasing `PING_GROUP_SZ` in `nmap.h` and then recompiling. The block size used in a `--randomize-hosts` scan is four times the value of `PING_GROUP_SZ`. Note that higher values of `PING_GROUP_SZ` eat up more host memory. An alternative solution is to generate the target IP list with a list scan (`-sL -n -oN _`<filename>`_`), randomize it with a Perl script, then provide the whole list to Nmap with `-iL`. You will probably have to use this approach if you are scanning a huge network such as 10.0.0.0/8 and want all 16 million IP addresses randomized.

Fragment packets

IP fragments can be a major problem for intrusion detection systems, particularly because the handling of oddities such as overlapping fragments and fragmentation assembly timeouts are ambiguous and differ substantially between platforms. Because of this, the IDS often has to guess at how the remote system will interpret a packet. Fragment assembly can also be resource intensive. For these reasons, many intrusion detection systems still do not support fragmentation very well. Specify the `-f` to specify that a port scan use tiny (8 data bytes or fewer) IP fragments. See the section called “Fragmentation” for more important details.

Evade specific rules

Most IDS vendors brag about how many alerts they support, but many (if not most) are easy to bypass. The most popular IDS among Nmap users is the open-source Snort. Example 10.20 shows all of the default rules in Snort 2.0.0 that reference Nmap.

Example 10.20. Default Snort rules referencing Nmap

```
felix~/src/snort-2.0.0/rules> grep -i nmap *
icmp.rules:alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING NMAP";
dsize:0;itype: 8;reference:arachnids,162;
classtype:attempted-recon;sid:469;rev:1;)
scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap XMAS";
flags:FPU;reference:arachnids,30;classtype:attempted-recon; sid:1228;rev:1;)
scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP";
flags:A;ack:0;reference:arachnids,28;classtype:attempted-recon;sid:628;rev:1;)
scan.rules:alert tcp $EXTERNAL_NET any ->
```

$HOME_NET any (msg:"SCAN nmap fingerprint attempt";
flags:SFPU;reference:arachnids,05;classtype:attempted-recon; sid:629; rev:1;)
web-attacks.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-ATTACKS nmap command attempt";
flow:to_server,established;content:"nmap%20";
nocase;sid:1361;classtype:web-application-attack; rev:4;)

Now let us look at these rules through the eyes of an attacker. The first rule looks for an ICMP ping packet without any payload (`dsize:0`). Simply specifying a non-zero `--data-length` option will defeat that rule. Or the user could specify a different type of ping scan entirely, such as TCP SYN ping.

The next rule searches for TCP packets with the FIN, PSH, and URG flags set (flags:FPU) and signals an Nmap Xmas scan alert. Adding the option `--scanflags FINPSH` to the Xmas scan options will remove the URG flag. The scan will still work as expected, but the rule will fail to trigger.

The third rule in the list looks for TCP packets with the ACK bit set but an acknowledgment number of zero (flags:A;ack:0). Ancient versions of Nmap had this behavior, but it was fixed in 1999 in response to the Snort rule.

Rule number four looks for TCP packets with the SYN, FIN, PSH, and URG flags set (flags:SFPU). It then declares an Nmap OS fingerprinting attempt. An attacker can avoid flagging this by omitting the `-O` flag. If he really wishes to do OS detection, that single test can be commented out in `osscan2.cc` . The OS detection will still be quite accurate, but the IDS alert will not flag.

The final rule looks for people sending the string " `nmap` " to web servers. They are looking for attempts to execute commands through the web server. An attacker could defeat this by renaming Nmap, using a tab character instead of a space, or connecting with SSL encryption if available.

Of course there are other relevant rules that do not have Nmap in the name but could still be flagged by intrusive port scans. Advanced attackers install the IDS they are concerned with on their own network, then alter and test scans in advance to ensure that they do not trigger alarms.

Snort was only chosen for this example because its rules database is public and it is a fellow open-source network security tool. Commercial IDSs suffer from similar issues.

Avoid easily detected Nmap features

Some features of Nmap are more conspicuous than others. In particular, version detection connects to many different services, which will often leave logs on those machines and set off alarms on intrusion detection systems. OS detection is also easy to spot by intrusion detection systems, because a few of the tests use rather unusual packets and packet sequences. The Snort rules shown in Example 10.20, "Default Snort rules referencing Nmap" demonstrate a typical Nmap OS detection signature.

One solution for pen-testers who wish to remain stealthy is to skip these conspicuous probes entirely. Service and OS detection are valuable, but not essential for a successful attack. They can also be used on a case-by-case basis against machines or ports that look interesting, rather than probing the whole target network with them.

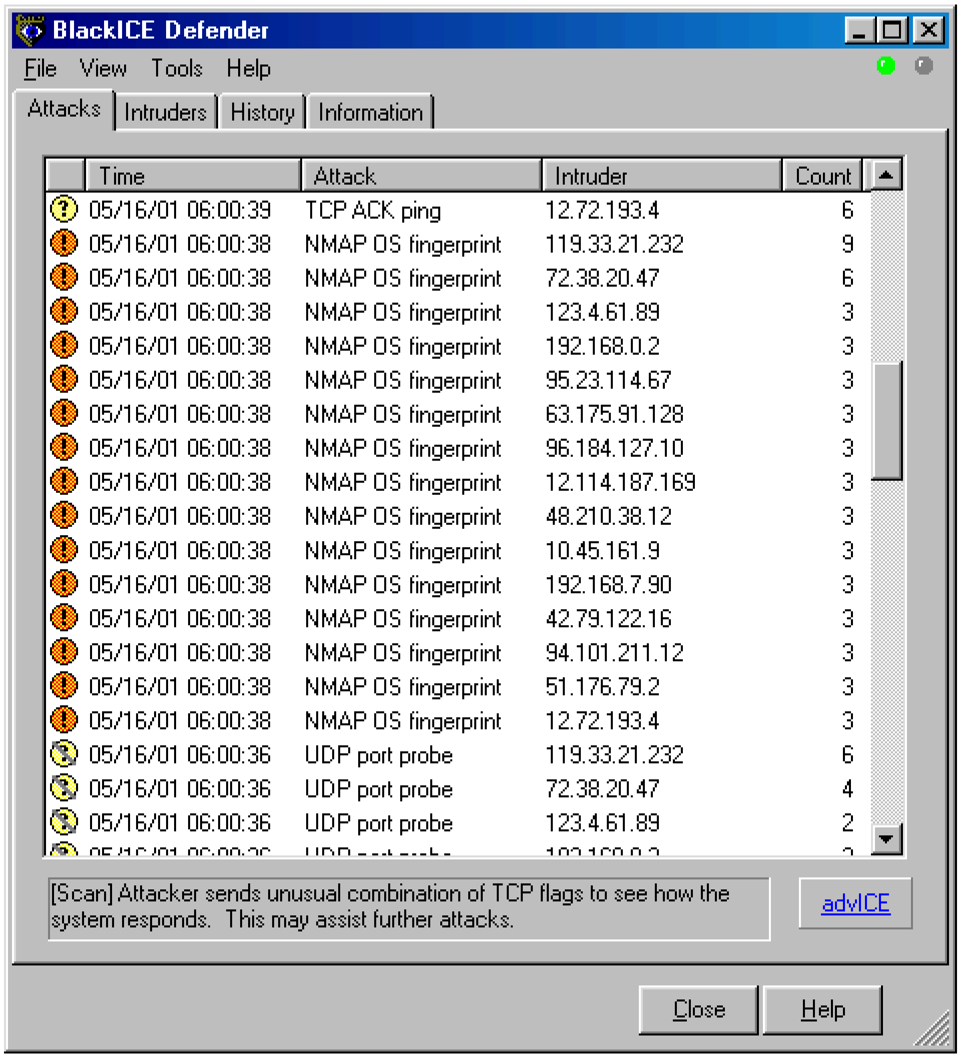
Misleading Intrusion Detection Systems

The previous section discussed using subtlety to avoid the watchful eye of intrusion detection systems. An alternative approach is to actively mislead or confuse the IDS with packet forgery. Nmap offers numerous options for effecting this.

Decoys

Street criminals know that one effective means for avoiding authorities after a crime is to blend into any nearby crowds. The police may not be able to tell the purse snatcher from all of the innocent passersby. In the network realm, Nmap can construct a scan that appears to be coming from dozens of hosts across the world. The target will have trouble determining which host represents the attackers, and which ones are innocent decoys. While this can be defeated through router path tracing, response-dropping, and other active mechanisms, it is generally an effective technique for hiding the scan source. Figure 10.2 shows a BlackICE report screen that is inundated with decoys. The administrator cannot complain to the providers for every ISP on the list. It would take a long time, and all but one of the hosts are innocent.

Figure 10.2. An attacker masked by dozens of decoys



Time	Attack	Intruder	Count
05/16/01 06:00:39	TCP ACK ping	12.72.193.4	6
05/16/01 06:00:38	NMAP OS fingerprint	119.33.21.232	9
05/16/01 06:00:38	NMAP OS fingerprint	72.38.20.47	6
05/16/01 06:00:38	NMAP OS fingerprint	123.4.61.89	3
05/16/01 06:00:38	NMAP OS fingerprint	192.168.0.2	3
05/16/01 06:00:38	NMAP OS fingerprint	95.23.114.67	3
05/16/01 06:00:38	NMAP OS fingerprint	63.175.91.128	3
05/16/01 06:00:38	NMAP OS fingerprint	96.184.127.10	3
05/16/01 06:00:38	NMAP OS fingerprint	12.114.187.169	3
05/16/01 06:00:38	NMAP OS fingerprint	48.210.38.12	3
05/16/01 06:00:38	NMAP OS fingerprint	10.45.161.9	3
05/16/01 06:00:38	NMAP OS fingerprint	192.168.7.90	3
05/16/01 06:00:38	NMAP OS fingerprint	42.79.122.16	3
05/16/01 06:00:38	NMAP OS fingerprint	94.101.211.12	3
05/16/01 06:00:38	NMAP OS fingerprint	51.176.79.2	3
05/16/01 06:00:38	NMAP OS fingerprint	12.72.193.4	3
05/16/01 06:00:36	UDP port probe	119.33.21.232	6
05/16/01 06:00:36	UDP port probe	72.38.20.47	4
05/16/01 06:00:36	UDP port probe	123.4.61.89	2

[Scan] Attacker sends unusual combination of TCP flags to see how the system responds. This may assist further attacks.

Warning: Many retail (dialup, cable modem, DSL, etc.) ISPs filter out most spoofed packets, though spoofed packets from the same network range as yours may get through. Do some tests first against some machine you control across the Internet, or you could even test this against 3rd party servers using IP ID tricks similar to those discussed in the section called “IP ID Tricks”.

Decoys are added with the `-D` option. The argument is a list of hosts, separated by commas. The string `ME` can be used as one of the decoys to represent where the true source host should appear in the scan order. Otherwise it will be a random position. Including `ME` in the 6th position or further in the list prevents some common port scan detectors from reporting the activity. For example, Solar Designer's excellent Scanlogd only reports the first five scan sources to avoid flooding its logs with decoys.

You can also use `RND` to request a random, non-reserved IP address, or `` RND:_`<number>`_ `` to generate *`<number>`* random addresses.

Note that the hosts used as decoys should be up and running. It would be pretty easy to determine which host is scanning if only one is actually up on the network. Using too many down decoys can also cause target ports to become temporarily unresponsive, due to a condition known as a SYN flood. Using IP addresses instead of names is advised to avoid appearing in the decoy networks' nameserver logs. The targets themselves should ideally be expressed by IP addresses too.

Decoys are used both in the initial ping scan (using ICMP, SYN, ACK, or whatever) and during the actual port scanning phase. Decoys are also used during remote OS detection. They are not used for DNS queries or service/version detection, so you will give yourself away if you use options such as `-sV` or `-A`. Using too many decoys can slow a scan dramatically, and sometimes even make it less accurate.

Port scan spoofing

While a huge group of decoys is quite effective at hiding the true source of a port scan, the IDS alerts will make it obvious that someone is using decoys. A more subtle, but limited, approach is to spoof a port scan from a single address. Specify `-S` followed by a source IP, and Nmap will launch the requested port scan from that given source. No useful Nmap results will be available since the target will respond to the spoofed IP, and Nmap will not see those responses. IDS alarms at the target will blame the spoofed source for the scan. You may have to specify `` -e _`<interfacename>`_ `` to select the proper interface name (such as eth0, ppp0, etc.) for Nmap to send the spoofed packets through. This can be useful for framing innocent parties, casting doubt in the administrator's mind about the accuracy of his IDS, and denial of service attacks that will be discussed in the section called “DoS Attacks Against Reactive Systems”.

Idle scan

Idle scan is a clever technique that allows for spoofing the source IP address, as discussed in the previous section, while still obtaining accurate TCP port scan results. This is done by abusing properties of the IP identification field as implemented by many systems. It is described in much more depth in the section called “TCP Idle Scan (`-sI`)”.

DNS proxying

Even the most carefully laid plans can be foiled by one little overlooked detail. If the plan involves ultra-stealth port scanning, that little detail can be DNS. As discussed in the section called “DNS Resolution”, Nmap performs reverse-DNS resolution by default against every responsive host. If the target network administrators are the paranoid log-everything type or they have an extremely sensitive IDS, these DNS lookup probes could be detected. Even something as unintrusive as a list scan (`-sL`) could be detected this way. The probes will come from the DNS server configured for the machine running Nmap. This is usually a separate machine maintained by your ISP or organization, though it is sometimes your own system.

The most effective way to eliminate this risk is to specify `-n` to disable all reverse DNS resolution. The problem with this approach is that you lose the valuable information provided by DNS. Fortunately, Nmap offers a way to gather this information while concealing the source. A substantial percentage of DNS servers on the Internet are open to recursive queries from anyone. Specify one or more of those name servers to the `--dns-servers` option of Nmap, and all rDNS queries will be proxied through them. Example 10.21 demonstrates this technique by conducting a list scan of some SecurityFocus IPs while using the public recursive DNS servers `4.2.2.1` and `4.2.2.2` to cover any tracks. Keep in mind that forward DNS still uses your host's configured DNS server, so specify target IP addresses rather than domain names to prevent even that tiny potential information leak. For this reason, Example 10.21 first shows the Linux host command being used to look up `www.securityfocus.com` rather than specifying that host name in the Nmap command line. To avoid IDS thresholds based on the number of requests from a single DNS server, you may specify dozens of comma-separated DNS servers to `--dns-servers` and Nmap will round-robin its requests among them.

Example 10.21. Using DNS Proxies (Recursive DNS) for a Stealth List Scan of SecurityFocus

```
host www.securityfocus.com 4.2.2.1
Using domain server:
Address: 4.2.2.1#53

www.securityfocus.com has address 205.206.231.12
www.securityfocus.com has address 205.206.231.15
www.securityfocus.com has address 205.206.231.13
```

```
nmap --dns-servers 4.2.2.1,4.2.2.2 -sL 205.206.231.12/28

Starting Nmap ( https://nmap.org )
Host 205.206.231.0 not scanned
Host mail2.securityfocus.com (205.206.231.1) not scanned
Host ns1.securityfocus.com (205.206.231.2) not scanned
Host sgs1.securityfocus.com (205.206.231.3) not scanned
Host sgs2.securityfocus.com (205.206.231.4) not scanned
Host 205.206.231.5 not scanned
Host adserver.securityfocus.com (205.206.231.6) not scanned
Host datafeeds.securityfocus.com (205.206.231.7) not scanned
Host sfcm.securityfocus.com (205.206.231.8) not scanned
Host mail.securityfocus.com (205.206.231.9) not scanned
Host www.securityfocus.com (205.206.231.10) not scanned
Host www1.securityfocus.com (205.206.231.11) not scanned
Host www2.securityfocus.com (205.206.231.12) not scanned
Host www3.securityfocus.com (205.206.231.13) not scanned
Host media.securityfocus.com (205.206.231.14) not scanned
Host www5.securityfocus.com (205.206.231.15) not scanned
Nmap done: 16 IP addresses (0 hosts up) scanned in 0.27 seconds
```

DoS Attacks Against Reactive Systems

Many vendors are pushing what they call intrusion *prevention* systems. These are basically IDSs that can actively block traffic and reset established connections that are deemed malicious. These are usually inline on the network or host-based, for greater control over network activity. Other (non-inline) systems listen promiscuously and try to deal with suspicious connections by forging TCP RST packets. In addition to the traditional IPS vendors that try to block a wide range of suspicious activity, many popular small programs such as Port Sentry are designed specifically to block port scanners.

While blocking port scanners may at first seem like a good idea, there are many problems with this approach. The most obvious one is that port scans are usually quite easy to forge, as previous sections have demonstrated. It is also usually easy for attackers to tell when this sort of scan blocking software is in place, because they will not be able to connect to purportedly open ports after doing a port scan. They will try again from another system and successfully connect, confirming that the original IP was blocked. Attackers can then use the host spoofing techniques discussed previously (`-S` option) to cause the target host to block any systems the attacker desires. This may include important DNS servers, major web sites, software update archives, mail servers, and the like. It probably would not take long to annoy the legitimate administrator enough to disable reactive blocking. While most such products offer a whitelist option to prevent blocking certain important hosts, enumerating them all is extraordinarily difficult. Attackers can usually find a new commonly used host to block, annoying users until the administrator determines the problem and adjusts the whitelist accordingly.

Exploiting Intrusion Detection Systems

The most audacious way to subvert intrusion detection systems is to hack them. Many commercial and open source vendors have pitiful security records of product exploitability. Internet Security System's flagship RealSecure and BlackICE IDS products had a vulnerability which allowed the Witty worm to compromise more than ten thousand installations, then disabled the IDSs by corrupting their filesystems. Other IDS and firewall vendors such as Cisco, Checkpoint, Netgear, and Symantec have suffered serious remotely exploitable vulnerabilities as well. Open source sniffers have not done much better, with exploitable bugs found in Snort, Wireshark, tcpdump, FakeBO, and many others. Protocol parsing in a safe and efficient manner is extremely difficult, and most of the applications need to parse hundreds of protocols. Denial of service attacks that crash the IDS (often with a single packet) are even more common than these privilege escalation vulnerabilities. A crashed IDS will not detect any Nmap scans.

Given all of these vulnerabilities, exploiting the IDS may be the most viable way into the target network. A nice aspect of this approach is that you do not even have to find the IDS. Sending a rogue packet to any “protected” machine on the network is usually enough to trigger these IDS bugs.

Ignoring Intrusion Detection Systems

While advanced attackers will often employ IDS subversion techniques described in this chapter, the much more common novice attackers (script kiddies) rarely concern themselves with IDSs. Many companies do not even deploy an IDS, and those that do often have them misconfigured or pay little attention to the alerts. An Internet-facing IDS will see so many attacks from script kiddies and worms that a few Nmap scans to locate a vulnerable service are unlikely to raise any flags.

Even if such an attacker compromises the network, is detected by a monitored IDS, and then kicked out of the systems, that is a small loss. Hacking is often a numbers game for them, so losing one compromised network out of thousands is inconsequential. Such a well-patrolled network would have likely quickly noticed their usage (such as denial of service attacks, mass scanning, or spam sending) and shut them down anyway. Hackers want to compromise negligently administered and poorly monitored networks that will provide long-lasting nodes for criminal activity.

Being tracked down and prosecuted is rarely a concern of the IDS-ignoring set. They usually launch attacks from other compromised networks, which are often several globe-spanning hops away from their true location. Or they may use anonymous connectivity such as provided by some Internet cafes, school computer labs, libraries, or the prevalent open wireless access points. Throwaway dialup accounts are also commonly used. Even if they get kicked off, signing up again with another (or the same) provider takes only minutes. Many attackers come from Romania, China, South Korea, and other countries where prosecution is highly unlikely.

Internet worms are another class of attack that rarely bothers with IDS evasion. Shameless scanning of millions of IP addresses is preferred by both worms and script kiddies as it leads to more compromises per hour than a careful, targeted approach that emphasizes stealth.

While most attacks make no effort at stealth, the fact that most intrusion detection systems are so easily subverted is a major concern. Skilled attackers are a small minority, but are often the greatest threat. Do not be lulled into complacency by the large number of alerts spewed from IDSs. They cannot detect everything, and often miss what is most important.

Even skilled hackers sometimes ignore IDS concerns for initial reconnaissance. They simply scan away from some untraceable IP address, hoping to blend in with all of the other attackers and probe traffic on the Internet. After analyzing the results, they may launch more careful, stealthy attacks from other systems.