Network Intrusion Detection of Third Party Effects

by
Richard Bejtlich, TaoSecurity
richard@bejtlich.net
www.bejtlich.net
v1.0, 26 August 2000

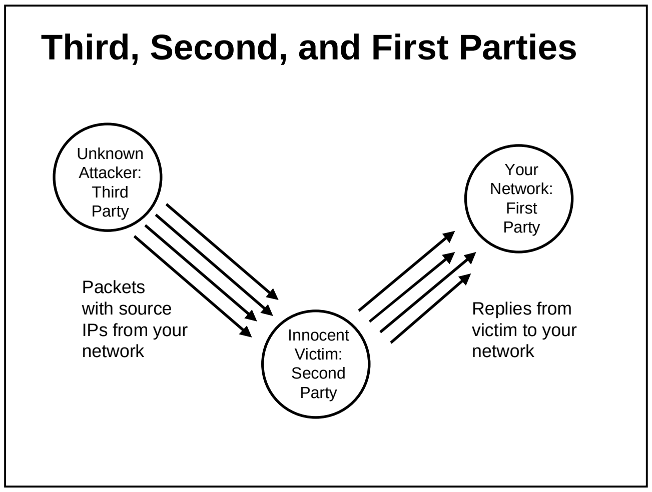
Introduction

Have you ever mingled in a crowded room during a party? Perhaps during the course of the evening someone bumped rather suddenly against you, spilling your drink and making you angry. Turning to confront your "attacker," you see him raise his hands to plead forgiveness, saying "*She* pushed me. Sorry!" Jabbing a thumb towards the truly guilty party, your assumed assailant continues to thread his way through the crowd. The guilty woman who started the whole incident cannot be found. You return to your business and dance the night away.

This scenario plays out hundreds or thousands of times per day in the world of network intrusion detection. Unfortunately, the crucial conversation between the first party (you) and the second party (the owner of the host who "bumps" your network) may never occur. Furthermore, neither you nor the second party can identify the third party (the "truly guilty" adversary who began the affair). We intrusion detectors find ourselves dealing with "third party effects" on a daily basis. We may never be sure if we are accurately identifying and responding to traffic from attackers or victims. The term "third party effects" is used to describe network events similar to the crowded party scenario. The purpose of this paper is to explain the technical characteristics of third party effects to help intrusion detectors improve their decision making processes. Hopefully, we will spend more time dancing and less time confronting innocent party-goers!

General Theory

Third party effects are almost always caused by spoofing. The third party (the attacker) assumes the identity of the first party (you) and assaults the second party (the true "victim"). The following diagram explains this situation:


Third, Second, and First Parties
Unknown Attacker: Third Party
Packets with source IPs from your network
Innocent Victim: Second Party
Replies from victim to your network
Your Network: First Party

Why does the attacker assume the identity of the first party? Generally, the first party wants to conceal herself, which in some cases is crucial to the success of her attack, or at least to hiding her identity and preserving her freedom. This is true for two categories of malicious activity: (1) reconnaissance and (2) denial of service. For reconnaissance, an attacker may gain some cover for her actions by mingling her true source address among a dozen or more spoofed source addresses. This will confuse the victim, making him think multiple hosts are conducting coordinated reconnaissance against his network. For denial of service, attacks which rely upon non-existent source addresses demand spoofing to succeed. A low-skilled attacker who chooses active addresses, such as your own, may not succeed in conducting a truly devastating assault upon a victim. Furthermore, some methods of defense against DoS rely upon identifying and blocking offensive IPs. Constantly changing, randomly selected addresses are difficult to defend against. The process by which addresses are selected is explained in the following sections.

Reconnaissance

Reconnaissance, or network mapping, is one simple event where spoofed source addresses can be used against a target. Essentially, the attacker hopes to divert attention and confuse the victim by hiding her packets amongst those with spoofed source addresses. This strategy is not without merit. Most intrusion detection operations centers have trouble properly handling packets from a multitude of source addresses. Selecting at least four false addresses should be enough to confuse most intrusion detectors. Given five sources to process, many intrusion detectors will log the event under the category “multiple sources,” and won’t bother to pursue *any* of the apparent offenders. In a time-critical and manning-depleted work environment, many intrusion detectors will record the event and press on.

Reconnaissance using false source addresses can be done using the -D (decoy) option in nmap, for example. The attacker provides the decoy addresses. Nmap’s man pages caution against selecting hosts which do not exist or are unreachable. An astute defender could isolate the attacker’s address from decoys if he finds the decoy addresses are patently false. What happens when the attacker chooses *your* IP address as a decoy? We now witness the first example of “third party effects.”

Reconnaissance-based third party effects are responses by the second party (the victim of the scanning activity) to the presumed scanner. While the second party will reply to the third party, he will also reply to your host and any other addresses chosen as decoys. For TCP-based scans, responses from live victims will appear as SYN ACK packets for open ports or RST ACK packets for closed ports, assuming the attacker is conducting some variation of a SYN scan. Non-SYN TCP-based scans, such as FIN scans, will elicit varying responses. Any scan which sets the ACK bit, such as an ACK sweep (the so-called “TCP ping”) can not generally be used to find open ports. Scans with a set ACK bit *can* be used to locate hosts. Hosts which do not exist cannot reply, and their upstream router may return ICMP host unreachable messages. For UDP-based scans, responses from live victims may appear as UDP packets for open ports or ICMP destination unreachable messages for closed ports. For ICMP-based scans, responses from live victims will take the form of some sort of ICMP response or error message. As with TCP-based probes to nonexistent hosts, UDP or ICMP-based probes to nonexistent hosts may elicit ICMP host unreachable messages from upstream routers.

Given the wide variety of possible responses to decoyed source addresses, innocent owners of those addresses (first parties) can potentially see dozens of packets of varying natures. This makes the first party intrusion detector's job very difficult. Should your address be spoofed as a decoy, you could see all of the replies directed at your address, apparently "out of the blue" (or ether?) This fact alone causes many experienced intrusion detectors to cast a suspicious eye when seeing odd packets. Fortunately, decoy scans do not seem to be as pervasive as one might expect, at least when compared to the level of spoofing employed in denial of service attacks.

Denial of Service

Denial of service attacks are among the most annoying network events imaginable. DoS is typically designed to consume a victim's bandwidth or computing resources. Spoofing can be employed in either scenario. Therefore, even though your network may not be the perpetrator or victim of a DoS event, you may become involved if the attacker spoofs your addresses.

Bandwidth consumption attacks can be implemented via two main methods. First, an attacker using a single source host with superior throughput may overwhelm a victim hosted on a narrow pipe. Second, an attacker coordinating the actions of multiple "slaves" or "agents" may overpower a victim through strength in numbers. This scenario is typically called "distributed denial of service," or DDoS. Spoofing may be employed in either case to hide the perpetrator's or slaves' identities, but it is not crucial to a bandwidth consumption attack. Spoofing may also be used to complicate the victim's defensive measures. For example, randomly changing source addresses cannot be easily blocked at the perimeter. Some defensive devices may dynamically block IPs via automatic updates of router access control lists or firewall rulesets. Unless the screening device times out older ACL or ruleset entries, it is possible to create an unworkable ACL or ruleset, throttling throughput more effectively than the DDoS!

Computing resource consumption attacks are known to much of the networked world as "SYN floods," although other variations are possible. Effective SYN floods rely upon spoofing nonexistent source addresses. While the victim forever waits for an ACK in response to his SYN ACK, he consumes memory maintaining the SYN_RECEIVED state. Beyond SYN floods, computing resource consumption attacks can involve fragmentation, whereby an attacker stretches a victim's ability to dynamically reassemble fragmented packets. This technique may use spoofing to hide the perpetrator's identity, or to force the victim to devote memory to maintaining multiple packets requiring reassembly. This activity may appear to the victim as the following example, where an attacker is attempting to deny service on the victim's telnet port:

```
11:46:14.212003 spoofed.ip.one.1053 > flood.victim.com.23:
 S 322286:322286(0) win 8192 <mss 536,nop,nop,sackOK> (DF)
11:46:14.598008 spoofed.ip.one.1054 > flood.victim.com.23:
 S 322286:322286(0) win 8192 <mss 536,nop,nop,sackOK> (DF)
11:46:14.975522 spoofed.ip.one.1055 > flood.victim.com.23:
 S 322286:322286(0) win 8192 <mss 536,nop,nop,sackOK> (DF)
```
etc... Note that here and in the traces that follow, a small subset of the activity is shown for the sake of brevity.

A victim may see dozens to hundreds of packets per second, for whatever duration the attacker chooses. Multiple source IPs can be observed, which may or may not exist. The victim will generate replies, which appear as the following:

```
11:46:14.765043 flood.victim.com.23 > spoofed.ip.one.1053:
 S 4137483508:4137483508(0) ack 322287 win 8192 <mss 1460>
11:46:14.891108 flood.victim.com.23 > spoofed.ip.one.1054:
 S 4164828806:4164828806(0) ack 322287 win 8192 <mss 1460>
11:46:15.019029 flood.victim.com.23 > spoofed.ip.one.1055:
 S 4192020032:4192020032(0) ack 322287 win 8192 <mss 1460>
```
etc...

In my professional opinion and experience, third party effects caused by SYN floods and related phenomena generate a large portion of the seemingly inexplicable traffic observed today. To that end, much of the remainder of this paper is devoted to explaining the process in detail and providing illustrative network traces.

Third Party SYN Flood Effects

Third party SYN flood effects, by definition, involve three parties. To recap, they are:

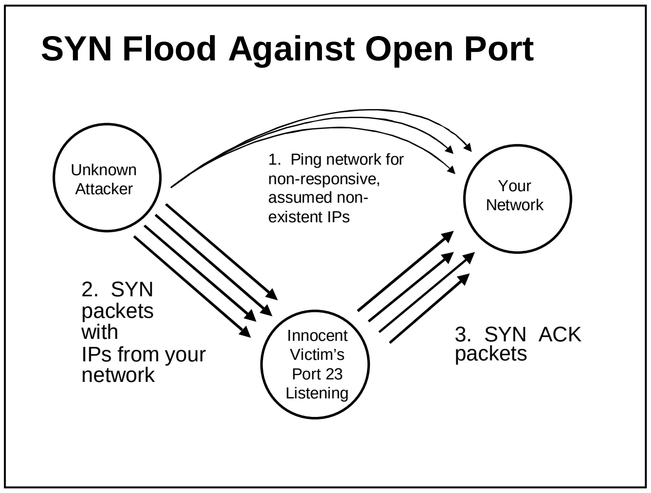
- Third party: the attacker (assumed to be female in this paper)
- Second party: the victim (assumed to be male in this paper)
- First party: you and your network (assumed to be whatever you are!)

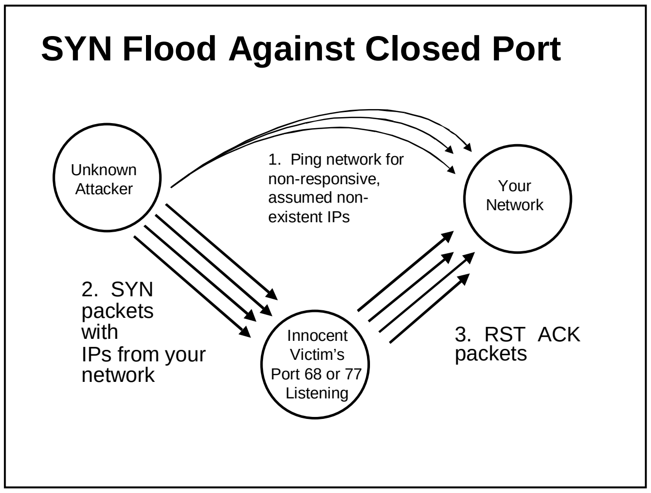
Attacks which *do not* involve the spoofing of addresses you own are *not visible to you*. Occasionally you can post your traces in the Incidents forum at securityfocus.com or on the SANS GIAC (www.sans.org/giac.htm) and correlate traffic with other intrusion detectors. Often you will see only a small and confusing subset of a massive event. Attacks where you are the victim are easy enough to recognize, although identifying the true perpetrator can be difficult, as this paper describes.

Now that we have limited this discussion to activity where your addresses are being spoofed, a question naturally follows: how are your addresses chosen? Three possibilities exist. First, the attacker deliberately chooses your addresses. She may do this to "frame" you, causing the victim to assume you are malicious. This may happen in the corporate world, whereby a saboteur tries to damage relations between business allies. Second, the attacker may choose addresses randomly; your network may be chosen by simple misfortune. Commonly used SYN flooders "shaft" and "syn4k" allow for random selection of source addresses. In fact, chance appears to play a large role in the third party effects universe, which may not be surprising, given the growth of the Internet and the millions of users riding the electronic lightning. Third, the attacker may look for nonexistent hosts, which is a requirement for effective SYN floods. If your hosts are live, how can your addresses be chosen for an attack?

In some cases, a SYN flood tool will allow the attacker to select a range of IPs for the spoofed source, or it will generate its own list. The utility will ping that range, trying to determine if any hosts exist. If no ICMP echo replies are heard, the SYN flooder assumes the IPs *do not exist* and are ideal spoofed sources. If those hosts are protected by a router or firewall denying ICMP echo requests, they *will not respond* with ICMP echo replies. This "flaw" in choosing good spoofable IPs causes a substantial amount of third party traffic. Essentially, your network becomes a third party to a SYN flood by virtue of having blocked ICMP echo requests. In other words, by trying to protect your hosts from network reconnaissance via ICMP, you have created a set of "spoofable" IPs. When your IPs are chosen by an attacker, you see an immense volume of traffic you did not solicit.

Notionally, a SYN flood may appear as either of the following:


SYN Flood Against Open Port
Unknown Attacker
1. Ping network for non-responsive, assumed non-existent IPs
Your Network
2. SYN packets with IPs from your network
Innocent Victim's Port 23 Listening
3. SYN ACK packets


SYN Flood Against Closed Port
Unknown Attacker
1. Ping network for non-responsive, assumed non-existent IPs
Your Network
2. SYN packets with IPs from your network
Innocent Victim's Port 68 or 77 Listening
3. RST ACK packets

In the first case, the traffic the first party (you) will see appears as the following:

```
11:46:14.765043 flood.victim.com.23 > spoofed.ip.one.1053:
 S 4137483508:4137483508(0) ack 322287 win 8192 <mss 1460>
11:46:14.891108 flood.victim.com.23 > spoofed.ip.one.1054:
 S 4164828806:4164828806(0) ack 322287 win 8192 <mss 1460>
11:46:15.019029 flood.victim.com.23 > spoofed.ip.one.1055:
 S 4192020032:4192020032(0) ack 322287 win 8192 <mss 1460>
```
etc...

These are the replies shown earlier. Look at them now with the thought that *you* own spoofed.ip.one. Without knowing that flood.victim.com is really a victim of a SYN flood, you could naturally assume that he is performing some sort of activity against you! Watch out, somebody bumped into you at the party, and it's not his fault.

In the second case, the traffic the first party (you) will see appears as the following:

```
20:31:15.794717 victim.isp.net.68 > spoofed.ip.one.29470:
 R 0:0(0) ack 723645348 win 0 (ttl 242, id 40923)
20:31:20.190800 victim.isp.net.68 > spoofed.ip.one.48926:
 R 0:0(0) ack 960212644 win 0 (ttl 242, id 56829)
```
more of the same follow...
```
20:31:17.754903 victim.isp.net.77 > spoofed.ip.two.44376:
 R 0:0(0) ack 1861342051 win 0 (ttl 242, id 25377)
20:31:22.054453 victim.isp.net.77 > spoofed.ip.two.13400:
 R 0:0(0) ack 454770019 win 0 (ttl 242, id 40905)
```
more of the same follow…

Here we assume you own both spoofed.ip.one and spoofed.ip.two. Note the packets have been reordered for readability. Traffic in the wild can be more difficult to interpret in its raw format, and sorting by IP address or other values can occasionally be enlightening.

In the first case, we notice a SYN flood against a listening port, as it responds with SYN ACK packets. In the second case, it appears as though ports 68 and 77 are not listening, as they reply with RST ACK packets. Why would anyone bother SYN flooding a closed port? What's the point of DoS against a non-active service? The answer is that many adversaries may attack one, two, or all 65,535 ports when conducting denial of service. Some inexperienced attackers may assume "more is better" and believe SYN flooding every port is more effective than a concentrated strike. There is a second reason why RST ACK packets may be observed by the first party. A listening port may initially respond to incoming SYN packets with SYN ACK responses. As the assault progresses, the port may become "congested" and reply with RST ACK packets until its queue clears. For example, the following pattern was created in a lab setting, via a SYN flood against an open port 139 tcp (netbios-ssn) on c1instructor. These packets are the responses generated by the flood victim:

```
10:31:50.017282 c1instructor.netbios-ssn > 197.121.183.58.1545: S
154146803:154146803(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
45009)
10:31:50.027612 c1instructor.netbios-ssn > 94.16.12.187.1455: S
154146814:154146814(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
45265)
```
...now we see a change! ...
```
10:31:50.407277 c1instructor.netbios-ssn > 20.141.170.225.2034: R 0:0(0) ack
674719802 win 0 (ttl 128, id 52945)
10:31:54.657074 c1instructor.netbios-ssn > 121.111.27.14.1067: R 0:0(0) ack
674719802 win 0 (ttl 128, id 42707)
10:31:54.677318 c1instructor.netbios-ssn > 171.158.197.16.2036: R 0:0(0) ack
674719802 win 0 (ttl 128, id 43219)
```

...continues, then the port “reopens”...

```
10:31:59.017093 c1instructor.netbios-ssn > 194.110.96.191.2245: S
154146853:154146853(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
31445)
10:31:59.017161 c1instructor.netbios-ssn > 94.16.12.187.1455: S
154146814:154146814(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
31701)
10:31:59.017255 c1instructor.netbios-ssn > 144.63.182.189.2424: S
154146834:154146834(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
31957)
10:31:59.017768 c1instructor.netbios-ssn > mac-lab3.psychologie.uni-
greifswald.de.1625: S 154146883:154146883(0) ack 674719802 win 8576 <mss
1460> (DF) (ttl 128, id 33237)
10:31:59.697104 c1instructor.netbios-ssn > 174.168.28.139.1695: R 0:0(0) ack
674719802 win 0 (ttl 128, id 50133)
10:31:59.707059 c1instructor.netbios-ssn > 71.64.113.12.1606: R 0:0(0) ack
674719802 win 0 (ttl 128, id 50389)
```

...this pattern of SYN ACK, then RST ACK, then SYN ACK continues...

This attack was generated using a tool similar to syn4k, using randomly generated source IP addresses. Some of the addresses to which c1instructor replies are clearly forged, such as 94.16.12.187 and 71.64.113.12. (These addresses are currently reserved by IANA.) Note that one address is shown by its hostname, mac-lab3.psychologie.uni-greifswald.de. This indicates a live host for which a DNS PTR record existed; its IP address is 141.53.95.66. If you owned this host, you would see an unsolicited SYN ACK packet from c1instructor and wonder what was happening. As the trace represents an edited snapshot of time, you could potentially see hundreds of SYN ACK and/or RST ACK packets from c1instructor to your mac-lab3.psychologie.uni-greifswald.de machine, if the SYN flooding tool chose to spoof 141.53.95.66 repeatedly. In fact, even in this abbreviated snapshot, we see two replies to 94.16.12.187, at 10:31:50.027612 and 10:31:59.017161!

Making Educated Guesses

At this point, we can see that SYN floods involving spoofed source addresses offer certain recognizable clues, such as nonexistent source addresses like 94.16.12.187 and 71.64.113.12. The second party (the victim of the attack) can therefore be certain no one owning those two IP addresses is legitimately at fault. What about the owner of 141.53.95.66 (mac-lab3.psychologie.uni-greifswald.de)? From his point of view, as the third party, he only sees the following arrive on his ether-doorstep:

```
10:31:59.017768 c1instructor.netbios-ssn > mac-lab3.psychologie.uni-
greifswald.de.1625: S 154146883:154146883(0) ack 674719802 win 8576 <mss
1460> (DF) (ttl 128, id 33237)
```

Fortunately, there are techniques he can employ to make an educated guess as to the type of activity he has witnessed. The first may be the least obvious but most effective: contacting the owner of the offending IP and asking for assistance.

An example of this occurred during 1999, while I was performing intrusion detection work. I observed the following pattern, pseudoanonymized to protect the innocent:

```
06:20:51.570058 firstclass.server.edu.510 > spoofed.ip.one.7002:
 R 0:0(0) ack 674711610 win 0 (ttl 116, id 48680)
13:55:27.737433 firstclass.server.edu.510 > spoofed.ip.three.6666:
 R 0:0(0) ack 674711610 win 0 (ttl 118, id 54468)
23:30:53.567215 firstclass.server.edu.510 > spoofed.ip.two.32771:
 R 0:0(0) ack 674711610 win 0 (ttl 117, id 25440)
```

My organization owned the IPs designated by spoofed.ip.one, .three, and .two. I was initially puzzled by the timing of the packets, as they were separated by hours. This could be the result of a wide variety of spoofed sources; perhaps I saw only a few? I guessed firstclass.server.edu to be a target host. These packets looked like responses, where port 510 was closed or flooded.

Researching port 510, I found it is the “firstclass” service, registered by SoftArc. SoftArc sells a product called the FirstClass Intranet Server, which can provide email, collaboration, and other services. The source IP belonged to a university, and the hostname included the word “firstclass.” It seemed that if a malicious Internet user wanted to perform a denial of service against this university, it might make sense to target port 510 tcp on the school's FirstClass server. Given the presence of RST ACK packets from port 510 to multiple IPs, it seemed the host’s buffer for port 510 was flooded and the port was now closed.

I contacted the school and confirmed their FirstClass server had been under a denial of service attack at the time and date noted in the packets sent to my hosts. The attacker was SYN flooding ports 68 (bootp) and 510 (firstclass). The firstclass.server.edu system was not compromised and it was not originating the packets sent to my hosts. It was an innocent victim, or the second party to a SYN flood perpetrated by an unknown third party. As the first party, I saw the RST ACK replies from the second party.

I employed the “contact the source technique” many other times. For example, I observed the following traces at a later date that same year:

```
10:20:52.097570 commercial.web.server.21 > spoofed.ip.one.1485:
 R 0:0(0) ack 674719802 win 0 (ttl 50, id 1034)
10:22:28.994103 commercial.web.server.23 > spoofed.ip.one.1485:
 R 0:0(0) ack 674719802 win 0 (ttl 50, id 38438)
10:25:43.004888 commercial.web.server.53 > spoofed.ip.one.1485:
 R 0:0(0) ack 674719802 win  (ttl 50, id 43626)
```

more of the same follow...

```
10:20:40.594667 commercial.web.server.21 > spoofed.ip.two.2104:
 R 0:0(0) ack 674719802 win 0 (ttl 45, id 14598)
10:22:17.576229 commercial.web.server.23 > spoofed.ip.two.2104:
 R 0:0(0) ack 674719802 win 0 (ttl 45, id 11298)
10:25:31.402693 commercial.web.server.53 > spoofed.ip.two.2104:
R 0:0(0) ack 674719802  0 (ttl 45, id 33894)
```

more of the same follow...

This source IP belonged to a commercial web site. While the three "source" ports, 21 (ftp), 23 (telnet), and 53 (dns) made little sense as true source ports, they might be good candidates as targets of a SYN flood. (They could possibly make sense as source ports if an attacker was trying to evade packet filtering rules, but that's a story for another day.) Sure enough, after contacting the web site, the system administrator told me a hired security consultant had tested the web server with a denial of service attack at the exact date and time indicated by my logs.

We have established that contacting the source can be a valuable source of information. Unfortunately, this technique is not always prudent or productive. If the source addresses belong to a source with which you would prefer not to have contact, calling or emailing them is not a good idea. For example, you may be wary of contacting owners of hosts in "rogue states." You may not believe the owner of the apparently offensive host is trustworthy. Furthermore, the owner may not speak your language, or he may ignore your requests for help.

Beyond contacting the source, two other techniques may be helpful. First, let's look at two more traces where SYN ACK packets are sent to IPs you own (spoofed.ip.one and .two):

```
05:41:36.772836 major.irc.host.6666 > spoofed.ip.one.1578:
 S 1822395560:1822395560(0) ack 674711610 win 4096 <mss 1460> (DF)
05:41:53.834459 major.irc.host.6666 > spoofed.ip.two.1578:
 S 311457256:311457256(0) ack 674711610 win 4096 <mss 1460> (DF)
05:42:00.765914 major.irc.host.6667 > spoofed.ip.three.1433:
 S 1074583123:1074583123(0) ack 674711610 win 61440 <mss 1460> (DF)

22:25:46.030135 biology.web.com.23 > spoofed.ip.one.2154:
 S 4154715243:4154715243(0) ack 674719802 win 8192 <mss 152>
22:26:24.456103 biology.web.com.23 > spoofed.ip.one.2026:
 S 159261598:159261598(0) ack 674719802 win 8192 <mss 32>
22:29:38.265734 biology.web.com.23 > spoofed.ip.one.1838:
 S 1866996756:1866996756(0) ack 674719802 win 8192 <mss 152>
```

Now, observe two traces involving RST ACK packets:

```
12:52:10.879563 auction.this.com.23 > spoofed.ip.one.1985:
 R 0:0(0) ack 674711610 win 0
12:54:37.882708 auction.this.com.23 > spoofed.ip.one.1554:
 R 0:0(0) ack 674711610 win 0
12:55:38.961649 auction.this.com.23 > spoofed.ip.one.1409:
 R 0:0(0) ack 674711610 win 0

22:34:47.629194 van.smack.net.21 > spoofed.ip.two.2031:
 R 0:0(0) ack 674719802 win 0
22:36:01.282720 van.smack.net.21 > spoofed.ip.two.1071:
 R 0:0(0) ack 674719802 win 0
22:36:11.483963 van.smack.net.21 > spoofed.ip.two.2143:
 R 0:0(0) ack 674719802 win 0
```

Do you notice anything significant among these packets? Each set offers packets with repeating ACK sequence numbers, seen in our two “contacting the source examples”: 674711610 and 674719802. Given the way the three-way handshake occurs, we can reasonably conclude that SYN packets with 674711609 and 674719801 generated their respective SYN ACK or RST ACK replies. In fact, tools do exist which generate the expected sequence numbers. A tool called “synk4” creates SYN 674719801 packets, according to DDoS guru David Dittrich, who made this discovery by analyzing the following section of code:

`#define SEQ 0x28376839` (0x28376839 is decimal 674719801)

According to Dave, “synk4 takes a source address on the command line for outgoing packets, and if zero, it generates them randomly using this code”:

```
...
        for (i=1;i>0;i++)
          {
             srandom((time(0)+i));
             srcport = getrandom(1, max)+1000;
             for (x=lowport;x<=highport;x++)
               {
                  if ( urip == 1 )
                    {
                       a = getrandom(0, 255);
                       b = getrandom(0, 255);
                       c = getrandom(0, 255);
                       d = getrandom(0, 255);
                       sprintf(junk, "%i.%i.%i.%i", a, b, c, d);
                       me_fake = getaddr(junk);
                    }
...
```

The source code is here:

http://packetstorm.securify.com/spoof/unix-spoof-code/synk4.zip

SYN packets with SYN 674711609 may be created with a tool called “shaft.” An analysis can be found here:

http://packetstorm.securify.com/distributed/shaft_analysis.txt

Interestingly, both tools allow the attacker to select random source IP generation.

To verify the operation of tools which used specific SYN sequence numbers, I tested syn4k in a lab environment. I generated the following packets while SYN flooding the open port 139 tcp on c1instructor:

```
10:31:49.206938 ppp-125-232.infonie.fr.1276 > c1instructor.netbios-ssn: S
674719801:674719801(0) win 65535 (ttl 30, id 27207)
10:31:49.216946 92.137.210.105.1186 > c1instructor.netbios-ssn: S
674719801:674719801(0) win 65535 (ttl 30, id 23737)
10:31:49.226939 245.33.39.234.1097 > c1instructor.netbios-ssn: S
674719801:674719801(0) win 65535 (ttl 30, id 20266)
10:31:49.236936 142.184.124.107.1007 > c1instructor.netbios-ssn: S
674719801:674719801(0) win 65535 (ttl 30, id 16796)
10:31:49.246998 39.80.209.236.2066 > c1instructor.netbios-ssn: S
674719801:674719801(0) win 65535 (ttl 30, id 13325)
10:31:49.256936 192.231.38.109.1976 > c1instructor.netbios-ssn: S
674719801:674719801(0) win 65535 (ttl 30, id 9855)
```

Again, we notice an IP which appears to exist but is chosen randomly by the tool: ppp-125-232.infonie.fr, a dial-up account in France. More importantly, we do indeed see SYN packets with 674719801 set as the initial sequence number. How did the c1instructor host respond?

```
10:31:49.207761 c1instructor.netbios-ssn > ppp-125-232.infonie.fr.1276:
 S 154145993:154145993(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
26577)
10:31:49.217259 c1instructor.netbios-ssn > 92.137.210.105.1186:
 S 154146003:154146003(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
26833)
10:31:49.237210 c1instructor.netbios-ssn > 142.184.124.107.1007:
 S 154146023:154146023(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
27089)
10:31:49.247279 c1instructor.netbios-ssn > 39.80.209.236.2066:
 S 154146033:154146033(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
27345)
10:31:49.257185 c1instructor.netbios-ssn > 192.231.38.109.1976:
 S 154146043:154146043(0) ack 674719802 win 8576 <mss 1460> (DF) (ttl 128, id
27601)
```

These SYN ACK 674719802 packets match the patterns we observed earlier from commercial.web.server, biology.web.com, and van.smack.net. How does a closed port appear? From the second party's (victim's) view:

```
09:36:03.778833 1Cust196.tnt52.dfw5.da.uu.net.1002 > c1instructor.login:
 S 674719801:674719801(0) win 65535
09:36:03.786883 216.187.222.69.2412 > c1instructor.login:
 S 674719801:674719801(0) win 65535
09:36:03.796771 113.83.51.198.2323 > c1instructor.login:
 S 674719801:674719801(0) win 65535
09:36:03.806770 10.234.136.71.1881 > c1instructor.login:
 S 674719801:674719801(0) win 65535
09:36:03.816852 SAINS.sapmed.ac.jp.1792 > c1instructor.login:
 S 674719801:674719801(0) win 65535
09:36:03.826850 60.25.50.73.1702 > c1instructor.login:
 S 674719801:674719801(0) win 65535
```

From the first party's (your) view:

```
09:36:03.779074 c1instructor.login > 1Cust196.tnt52.dfw5.da.uu.net.1002:
 R 0:0(0) ack 674719802 win 0
09:36:03.787107 c1instructor.login > 216.187.222.69.2412:
 R 0:0(0) ack 674719802 win 0
09:36:03.796996 c1instructor.login > 113.83.51.198.2323:
 R 0:0(0) ack 674719802 win 0
09:36:03.806996 c1instructor.login > 10.234.136.71.1881:
 R 0:0(0) ack 674719802 win 0
09:36:03.817074 c1instructor.login > SAINS.sapmed.ac.jp.1792:
 R 0:0(0) ack 674719802 win 0
09:36:03.827073 c1instructor.login > 60.25.50.73.1702:
 R 0:0(0) ack 674719802 win 0
```

Again, notice the randomly chosen addresses which do seem to exist, namely 1Cust196.tnt52.dfw5.da.uu.net and SAINS.sapmed.ac.jp.

Unfortunately, this technique of recognizing certain sequence numbers is not 100% foolproof. For example, certain SYN flooding tools do not use specific sequence numbers. The very first SYN flood example in this paper used random sequence numbers. Also, a crafty attacker could try to masquerade truly malicious activity directly against your network by employing these "universally known" sequence numbers. A cunning adversary could attempt network reconnaissance using SYN ACK 674711610 or RST ACK 674719802 packets to locate hosts through "inverse mapping." (Since the ACK bit is set in each case, she could not check for open ports on well-behaved target TCP/IP stacks.)

Beyond the SYN Flood

Earlier we mentioned the use of an ACK sweep or "TCP ping" to locate hosts. While most hosts are vulnerable to SYN floods, some TCP/IP stacks can be vulnerable to "ACK floods." While an ACK sweep is employed to find live hosts scattered across a domain, an ACK flood is used to perform a denial of service against a specific host. As with a SYN flood, an ACK flood may employ random source addresses. Therefore, the conditions exist to create third party ACK flood effects. The second party (the victim, or c1instructor) might see the following:

```
11:48:56.261221 226.245.223.19.2529 > c1instructor.netbios-ssn:
 . ack 1161178473 win 16384 (ttl 255, id 43068)
11:48:56.262008 231.159.128.79.11380 > c1instructor.netbios-ssn:
 . ack 2404935014 win 16384 (ttl 255, id 27789)
11:48:56.262055 53.116.113.78.4239 > c1instructor.netbios-ssn:
 . ack 3891874576 win 16384 (ttl 255, id 51072)
11:48:56.262100 100.85.51.103.17952 > c1instructor.netbios-ssn:
 . ack 1842262121 win 16384 (ttl 255, id 59153)
11:48:56.262145 16.85.242.75.2814 > c1instructor.netbios-ssn:
 . ack 2235884921 win 16384 (ttl 255, id 25130)
```

What do you notice about these packets? You may see that I enjoy performing denial of service attacks against instructors' machines (c1instructor here.) More importantly, you see packets with only the ACK flag set. Again, random source IPs are present, some of which clearly could not exist. 100.85.51.103, for example, is owned by IANA, while 231.159.128.79 is a multicast address and not valid for unicast traffic. How does the victim respond?

```
11:48:51.105468 c1instructor.netbios-ns > 192.168.1.2.netbios-ns:
 udp 68 (ttl 128, id 19159)
11:48:51.106415 192.168.1.2 > c1instructor: icmp: 192.168.1.2
 udp port netbios-ns unreachable (DF) (ttl 254, id 19323)
```

... Where are the responses to the initial packets? ...

```
11:49:01.044912 c1instructor.netbios-ssn > 203.48.38.105.14083:
 R 2319094109:2319094109(0) win 0 (ttl 128, id 20183)
11:49:10.283963 c1instructor.netbios-ssn > 132.83.95.12.3538:
 R 1104418348:1104418348(0) win 0 (ttl 128, id 20695)
11:49:24.140438 c1instructor.netbios-ssn > 164.251.210.2.4009:
 R 2105212682:2105212682(0) win 0 (ttl 128, id 21207)
```

First you notice a UDP packet from c1instructor to 192.168.1.2, followed by an ICMP unreachable message. While these are not directly related to the ACK flood, I included them to demonstrate that my packet capture utility (TCPDump) was active and recording traffic. Although the flood commenced at 11:48:56, we do not see a RST response until 11:49:01. Perhaps this flood was effective against the target, a Windows box. Nmap confirms the victim's identity:

```
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on c1instructor (192.168.4.13):
Port    State       Protocol  Service
139     open        tcp        netbios-ssn

TCP Sequence Prediction: Class=trivial time dependency
                         Difficulty=65 (Easy)
Remote operating system guess: Windows NT4 / Win95 / Win98

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

In any event, we see that the proper response to an ACK packet is simply a RST response -- not a RST ACK. This response is consistent with the requirements set forth in RFC 793: Transmission Control Protocol.

From the third party effect point of view, you may see a series of RST packets from a host which you may initially consider malicious. Indeed, certain tools, such as "resetter," can produce RST packets which are designed to kill connections. In this case, the RST packets from c1instructor were the result of a third party ACK flooding c1instructor. Although an individual owner would only see response RST packets destined for his machine, he might be able to imagine the larger picture and realize third party effects were at work.

Many other sorts of activities involving spoofing can cause unexpected packets to arrive at your network's front door. For example, a malicious third party could perform an ICMP echo flood against a victim second party. If your IP address is spoofed, the victim could reply to your addresses. Similarly, if an attacker conducts a UDP flood and spoofs your addresses, you may see the victim's replies. Keep in mind that replies can take the form of ICMP error messages for certain sorts of traffic.

Summary

The main goal of this paper was to familiarize the reader with reactions and responses from innocent victims, who may be subject to reconnaissance or denial of service. If a perpetrator spoofs your address space, you may see unsolicited traffic from an innocent second party. While it's tempting to hunt down the woman who pushed some guy into your space, sometimes it's best to keep on dancing!

Acknowledgements

I'd like to thank Karen Frederick for reviewing this paper and making editing suggestions. I'd also like to thank Dave Dittrich for some astute recognition of sequence numbers used in DoS tools.

This paper contains some material and concepts originally published under the title "Interpreting Network Traffic." I plan to publish new documents addressing topics included in that paper but not mentioned here. I will always maintain the most current version of this paper at bejtlich.net.