Global Information Assurance Certification Paper

Spoof Bounce

Kevin Van Dixon
19 February 2001

Introduction

By now every security professional must be aware of hacker Kevin Mitnick and the classic attack carried out against Tsutomu Shimomura on Christmas day 1994. The attack used IP address spoofing and TCP sequence number prediction to gain unauthorized root access to a diskless workstation from which further attacks were carried out [1][2].

While non-random TCP sequence numbers can lead to a vulnerability to session hijacking [3] and IP address spoofing leads to trust exploitation this paper will detail an attack called (in some places)[4] "spoof bounce" that exploits predictable IP IDs to allow nearly untraceable port scans.

What is "spoof bounce"?

The identification field of the IP protocol helps with re-assembly of packet data by remote routers and host systems. Its purpose is to give a unique value to packets so if fragmentation occurs along a route, they can be accurately re-assembled [5]. Many current IP stacks assure IP identification field uniqueness by simply incrementing the IP ID field by 1 or some other fixed number every time they send a packet.

The methods outlined here rely on the fact that the IP ID field must be unique and on the way that different IP implementations respond to packets with different characteristics. Most current IP implementations will respond to an unsolicited ACK with an RST and then increment the next IP ID by some fixed number, they don't respond to an unsolicited RST at all, and thus will not increment the IP ID field.

The essence of the spoof bounce technique is to find a system (called a sensor) with an IP address that you can spoof that sees little traffic and has predictable IP ID numbers. You then send traffic to a target, but spoof the source address of the sensor host in the packet you send. You can then examine the IP ID field of packets from the sensor host to determine if the target responded to the spoofed packet that was sent.

The first reference to this type of scan I can find is a 1998 BugTraq post by "antirez" [6] in which he says:

> *" I have uncovered a new tcp port scan method. Instead all others it allows you to scan using spoofed packets, so scanned hosts can't see your real address. In order to perform this i use three well known tcp/ip implementation peculiarities of*

most OS:

*(1) * hosts reply SYN|ACK to SYN if tcp target port is open, reply RST|ACK if tcp target port is closed.*

*(2) * You can know the number of packets that hosts are sending using id ip header field. See my previous posting 'about the ip header' in this ml.*

*(3) * hosts reply RST to SYN|ACK, reply nothing to RST."*

One year latter in another BugTraq post [7] LiquidK introduces the "idlescan" tool, attributing the original "antirez" post for inspiration and saying:

"By using this type of scanner, an attacker is able to fake portscans that appear as coming from the sensors, and is able to do it with a large network of distributed sensors, thus appearing to the target, that the attack is coming from a lot of different machines."

One day latter a similar tool called ipidscan was also announced [8] on BugTraq.

I find nothing further until November 2000 when Teri Bidwell posted the following [9] on BugTraq:

"For a year or so the scan tool has been unidentified and referred to by one Bugtaq poster as "mystery tool number 11". At least, I couldn't find any correlations that identified it anywhere on the web. If there have been postings about this and I just didn't find them, then please chalk this message up to IDS neophyte-ness on my part.

I decided to tackle finding the tool as a research project for GIAC certification, and I now believe the tool being used is a slight variant of Idlescan."

And most recently David Thibault's "PRACTICAL ASSIGNMENT FOR GIAC GCIA CERTIFICATION Network Security 2000" [4] has a very nice explanation of this technique.

My own experimentation with the idlescan tool (v0.1alpha3) shows that it may still be under development, because my examination of the traffic shows the spoofed requests being sent with random source ports, only the SYN flag being set and random IDs .

(tcpdump output from "./idlescan probe target –p 50-51")

```
03:49:18.442358 eth0 > sensor.20738 > target.50: S 1681692777:1681692777(0) win 1024 (ttl 64, id 44710)
03:49:18.450906 eth0 > sensor .47937 > target.50: S 424238335:424238335(0) win 1024 (ttl 64, id 48946)
03:49:18.458889 eth0 > sensor.38357 > target.50: S 596516649:596516649(0) win 1024 (ttl 64, id 50101)
03:49:20.398503 eth0 > sensor.9158 > target.51: S 783368690:783368690(0) win 1024 (ttl 64, id 24754)
03:49:20.406592 eth0 > sensor.48786 > target.51: S 1967513926:1967513926(0) win 1024 (ttl 64, id 20955)
03:49:20.414719 eth0 > sensor .9521 > target.51: S 304089172:304089172(0) win 1024 (ttl 64, id 30121)
```

In [4] and [9] the request were SYN/FIN's, source port = destination port and ID = 39426). And, LiquidK's TODO list in idlescan.c include adding probe types (icmp echo, different flag combinations) and adding scan types (fin/ack scan).

This type of scanning can also be done, more flexibly, using tools such as hping2 [6].

What can this technique be used for?

Now that we know what spoof bouncing is we can go on to what it might be used for. These uses are only those which I found during the research for this paper and some of the more obvious uses I could think of.

Stealth OS finger printing – You could use a tool such as hping2 to test for open ports and also to test for unique IP stack response ala Nmap [10].

Stealth Scanning - See "idlescan" and above references.

Firewalking – See antirez, "more about IP ID", BugTraq, 20 Nov 1999,URL: http://www.securityfocus.com/archive/1/35862 (19 Feb 2001)

Use it to map anti-spoofing rules – If you have a target that you know has an open port and an appropriate sensor that you can ping directly, you can test indirectly if your spoofed traffic was allowed outbound. By moving your sensor around the internet (or the network you are testing) you can find out what anti-spoofing rules are in place and where they are enforced.

How could someone use spoof bounce to evade detection?

There are several ways that this technique could be made more stealthy.

Find a sensor that is idle, but is actually running a server of some sort, (web server, mail backup, etc) and then hide your sensor probes in legitimate looking traffic. I.e. use a low traffic web server and legitimate web request to poll the sensor.

Use multiple sensors to distribute the scan, as idlescan does.

Use statistical techniques for noise reduction and filtering and you might be able to use sensors that actually have traffic on them.

Stopping and detecting spoof bounce attacks

The only real hope in stopping the spoof bounce attacks are for networks to tighten up anti-spoofing rules (push the rules further into the networks), and/or patch the current crop of IP implementations to use random IP IDs.

IDS hosts on each local subnet looking for traffic such as source IP != local subnet AND destination IP != local subnet and there exists some other recent traffic where destination IP = source IP above would detect this attack at the origin.

Another indication that this type of attack might be going on would be a higher than normal number of connection requests that are simply Reset after my hosts send a SYN/ACK.

The only way to verify this type of attack is at the sensor site. A few honey pots strewn around watching specifically for this type of traffic might catch some of the less cautious attempts. Here you are not looking for port scans or direct attacks, but rather for SYN/ACK and RST traffic from the target to the sensor unpaired with any outbound SYN, and probing traffic from the attacker at the same time.

Future implications.

What future implications can we deduce from all of this… should we start to ignore portscans because we can never truly be sure of where they are coming from? Should we be even more worried about portscans because we now know that what looked like a dead end when we were trying to track the source of a scan was a dead end for us but not for the bad guy?

I'm not sure. But I do know that I expect the tools for this type of scanning to become much more sophisticated. Anonymity is too valuable to hackers for this not to become a bigger issue.

I also know that we will be setting up systems to watch for this type of traffic from the perspective of all three (attacker, sensor, and target) players in this attack. I also have a much better idea of what to ask my counterparts to look for at the suspected sensor site "look for SYN/ACKs from *target to sensor* that don't correspond to any valid connections, and if found then find out what hosts are talking to the sensor and what kind of traffic they are sending"

Footnotes

[1] Shimomura, Tsutomu, "Technical details of the attack described by Markoff in NYT.", URL: http://www.netsys.com/firewalls/firewalls-9501/0900.html (18 Feb. 2001).

[2] "CERT® Advisory CA-1995-01 IP Spoofing Attacks and Hijacked Terminal Connections", 23 Sept 1997, URL: http://www.cert.org/advisories/CA-1995-01.html (18 Feb. 2001).

[3] Joncheray, Laurent, "Simple Active Attack Against TCP", URL: http://www.insecure.org/stf/iphijack.txt (18 Feb. 2001).

[4] Thibault, David, "PRACTICAL ASSIGNMENT FOR GIAC GCIA CERTIFICATION Network Security 2000", 17 Nov 2000, URL: http://www.sans.org/y2k/practical/David_Thibault_GCIA.html (18 Feb. 2001).

[5] D. E. Comer, 1995. Internetworking with TCP/IP, vol. 1. Upper Saddle River, N. J.: Prentice Hall.

[6] antirez, "new tcp scan method",BugTraq, 18 Dec 1998, URL: http://www.securityfocus.com/archive/1/11581 (18 Feb. 2001).

[7] LiquidK, "idlescan (ip.id portscanner)", BugTraq, 3 Dec 1999, URL: http://www.securityfocus.com/archive/1/37272 (18 Feb. 2001).

[8] marvin@nss.nu, "idlescan (ip.id portscanner)", BugTraq,4 Dec 1999, URL: http://www.securityfocus.com/archive/1/37428 (18 Feb. 2001).

[9] Bidwell, Teri, "mystery tool number 11", BugTraq, 13 Nov 2000, URL: http://www.securityfocus.com/archive/75/144723 (19 Feb. 2001).

[10] Fyodor,"Remote OS detection via TCP/IP Stack FingerPrinting",10 Apr 1999, URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html (19 Feb 2001)