

Understanding IP Fragmentation

spoonfork

Introduction

While Fragmentation has been every hacker's wet dream, it is a nightmare for Intrusion Detection System, packet filters, and routers. Techniques on how fragmentation has been used to evade IDS are documented everywhere, and fragmentation has been used as an effective method to penetrate a network's perimeter defenses, especially firewall that does not support stateful packet inspection.

But what is fragmentation? In this first installment of a few series of articles that I'm writing for HITB, I will be introducing fragmentation, and how fragmented IP datagram looks like. Next I will explain how to create fragmented packets using available and customized tools to evade IDS. Finally, I will explain how fragmented packets are detected by IDS sensor with bias towards Snort, a lightweight network Intrusion Detection engine. It is assumed the reader has familiarity with TCP/IP concepts.

The purpose of this article is to enable the hacker to choose attack mechanisms effectively. This assumes that reconnaissance have been done, and vulnerable servers, services or web application has been identified. Once the primary steps have been completed, fragmentation techniques will be used to attack the system. By employing fragmentation techniques, the hacker wishes to evade Intrusion Detection system, and at the same time, launch her attack with elegance and finesse.

Fragmentation

Why does fragmentation occur? Fragmentation happens when an IP datagram has to travel through a network with a maximum transmission unit (MTU) that is smaller than the size of the IP datagram. Thus, if I send an IP datagram that is bigger than 1500 bytes to an Ethernet network, the datagram needs to be fragmented. The packets are then assembled at the receiving host. Fragmentation can either at the originating sending host or at an intermediate router.

How are the packets reassembled? Note that with IP fragmentation, packets are not reassembled until they reach the final destination. It is reassembled at the IP layer at the receiving end. This is make fragmentation and reassembly transparent to the protocol layer (TCP and UDP). If one of the packets is lost, the whole packets needs to be transmitted again.

Packets are reassembled at the receiving host by associating each fragment with an identical fragment identification number, or frag id for short. The frag ID is actually a copy of the ID field (IP identification number) in the IP header. Besides that, each fragment must carry its "position" or "offset" in the original unfragmented packet. Thus the first fragment will have an offset of 0, since its seat is at the front row and counting starts from 0. Each fragments must also tell the length of data that it carries. This is like the compartments in a train. And finally, each fragment must flag the MF (more fragments) bit if it is not the last fragment.

Fragmenting a Packet

Here is a hypothetical example. Suppose that we want to send a 110 bytes ICMP packet on a network with MTU of 40 (well that's damn small, but this is for illustration purposes). This is a diagram of the original packet:

```
+-----+-----+-----+
| IP | ICMP | Data |
| header | header | |
+-----+-----+-----+
20 8 82 (bytes)
```

The packet will be fragmented as shown below.

Understanding IP Fragmentation

spoonfork

```
+-----+-----+-----+
Packet 1 | IP header (20) | ICMP (8) | Data (12) | ID=88, Len=20, Off=0, MF=1
+-----+-----+-----+
Packet 2 | IP header (20) | Data (20) | ID=88, Len=20, Off=20, MF=1
+-----+-----+-----+
Packet 3 | IP header (20) | Data (20) | ID=88, Len=20, Off=40, MF=1
+-----+-----+-----+
Packet 4 | IP header (20) | Data (20) | ID=88, Len=20, Off=60, MF=1
+-----+-----+-----+
Packet 5 | IP header (20) | Data (10) | ID=88, Len=10, Off=80, MF=0
+-----+-----+-----+
```

- ID - IP identification number
- Len - Data Length (data length does not include IP header)
- Off - Offset
- MF - More Fragment

Notice that the second packet and subsequent packets contains IP header that is copied from the original packet. There are no ICMP headers, except in the first packet. In a nutshell, the 110 ICMP packet is broken into 5 packet, with total lengths of 40, 40, 40, 40 and 30 bytes each. The ICMP data is broken into lengths of 12, 20, 20, 20, and 10 bytes each.

Using tcpdump, the packet logs will look like this:

```
31337.com > 14m3r.com: icmp: echo request (frag 88:20@0+)
31337.com > 14m3r.com: (frag 88:20@20+)
31337.com > 14m3r.com: (frag 88:20@40+)
31337.com > 14m3r.com: (frag 88:20@60+)
31337.com > 14m3r.com: (frag 88:20@80)
```

Notice anything yet? Attackers, here is the catch. The second packet does not contain what type of ICMP packet it is. Only the first packet contains the ICMP code. For the subsequent packets, we can determine that it is an ICMP packet because the IP header is there. This type of fragmented packet can easily beat a stateless router, firewall or IDS. An example would be a stateless packet filter that blocks inbound ICMP echo request. Upon seeing the first packet, it will drop it, whereas the rest of the packets creep in successfully.

How to Discover MTU?

It is very easy to detect the MTU of a particular network. Just send various sizes of ICMP echo requests to the target network with the don't fragment (DF) bit set. If the packet exceeds the MTU, the receiving end will return with an ICMP error "fragmentation needed but don't fragment bit set" message. Lower the size of the packet, and keep sending until you don't get any ICMP error messages. Available tools like traceroute and nemesys can be modified and used to discover MTUs.

Funky Frags

So what sorts of things that we can do with fragmented packets? Well, denial of service of course. One fine example is to send fragmented packets to a packet filter that drops inbound ICMP echo request. The rest of the packets will get in, and the upon receiving the packets without the initial fragment, the receiving host will pening kepala doing reassembly.

And how about sending fragmented packets will identical offsets? This will be a basis for a denial of service as well.

Understanding IP Fragmentation

spoonfork

Another cool Denial of Service attack is Teardrop. Teardrop works by sending packets with odd offset. For example, my first offset is 0, and data length is 40. The next packet's data length is 10, and offset 20. This is a headache to the host because in order to reassemble this packet, it has to rewind back by 20. Old and unpatched operating system does not deal with negative numbers well, and negatives numbers can translate to very large positive numbers.

Another attack that used fragmentation effectively is the Ping of Death. Ping of Death works by sending a very large echo packet (65535 bytes) which is fragmented and reassembled at the receiving end. The last packet however, is crafted in such a way that the offset plus data length is greater than 65535. This will cause a possible internal overflow of 16-bit internal variable on the system, resulting in blue screen of death, reboot, etc.

Last but not least, a denial of service on routers can be also be done by never sending the last fragment - just keep on sending packets with the promise of more fragments (MF bit set). The router, in the hope of receiving more packets for reassembly, will not timeout. Sooner or later, it will become overloaded. Of course, you can also try to send fragmented packets to the router without the initial first fragment.

Another trick is to send fragmented TCP headers to the target source to avoid detection. NMAP for example, can send fragmented SYN packet in multiple fragments.

Summary

In this article I've discussed fragmentation theory. I've also introduce attacks that use fragmentation methods. In the next article, we'll go through the steps of creating fragmented packets in the hope to evade intrusion detection system. Comments are welcome at spoonfork@onebox.com (c) 2001 spoonfork, <http://ini2.net/mel>