Network Working Group
Request for Comments: 1858
Category: Informational

G. Ziemba
Alantec
D. Reed
Cybersource
P. Traina
cisco Systems

October 1995

Security Considerations for IP Fragment Filtering

Status of This Memo

Abstract

IP fragmentation can be used to disguise TCP packets from IP filters used in routers and hosts. This document describes two methods of attack as well as remedies to prevent them.

1. Background

System administrators rely on manufacturers of networking equipment to provide them with packet filters; these filters are used for keeping attackers from accessing private systems and information, while permitting friendly agents to transfer data between private nets and the Internet. For this reason, it is important for network equipment vendors to anticipate possible attacks against their equipment and to implement robust mechanisms to deflect such attacks.

The growth of the global Internet has brought with it an increase in "undesirable elements" manifested in antisocial behavior. Recent months have seen the use of novel attacks on Internet hosts, which have in some cases led to the compromise of sensitive data.

Increasingly sophisticated attackers have begun to exploit the more subtle aspects of the Internet Protocol; fragmentation of IP packets, an important feature in heterogeneous internetworks, poses several potential problems which we explore here.

2. Filtering IP Fragments

IP packet filters on routers are designed with a user interface that hides packet fragmentation from the administrator; conceptually, an IP filter is applied to each IP packet as a complete entity.

One approach to fragment filtering, described by Mogul [1], involves keeping track of the results of applying filter rules to the first fragment (FO==0) and applying them to subsequent fragments of the same packet. The filtering module would maintain a list of packets indexed by the source address, destination address, protocol, and IP ID. When the initial (FO==0) fragment is seen, if the MF bit is set, a list item would be allocated to hold the result of filter access checks. When packets with a non-zero FO come in, look up the list element with a matching SA/DA/PROT/ID and apply the stored result (pass or block). When a fragment with a zero MF bit is seen, free the list element.

Although this method (or some refinement of it) might successfully remove any trace of the offending whole packet, it has some difficulties. Fragments that arrive out of order, possibly because they traveled over different paths, violate one of the design assumptions, and undesired fragments can leak through as a result. Furthermore, if the filtering router lies on one of several parallel paths, the filtering module will not see every fragment and cannot guarantee complete fragment filtering in the case of packets that should be dropped.

Fortunately, we do not need to remove all fragments of an offending packet. Since "interesting" packet information is contained in the headers at the beginning, filters are generally applied only to the first fragment. Non-first fragments are passed without filtering, because it will be impossible for the destination host to complete reassembly of the packet if the first fragment is missing, and therefore the entire packet will be discarded.

The Internet Protocol allows fragmentation of packets into pieces so small as to be impractical because of data and computational overhead. Attackers can sometimes exploit typical filter behavior and the ability to create peculiar fragment sequences in order to sneak otherwise disallowed packets past the filter. In normal practice, such pathalogical fragmentation is never used, so it is safe to drop these fragments without danger of preventing normal operation.

3. Tiny Fragment Attack

With many IP implementations it is possible to impose an unusually small fragment size on outgoing packets. If the fragment size is made small enough to force some of a TCP packet’s TCP header fields into the second fragment, filter rules that specify patterns for those fields will not match. If the filtering implementation does not enforce a minimum fragment size, a disallowed packet might be passed because it didn’t hit a match in the filter.

STD 5, RFC 791 states:

> Every internet module must be able to forward a datagram of 68 octets without further fragmentation. This is because an internet header may be up to 60 octets, and the minimum fragment is 8 octets.

Note that, for the purpose of security, it is not sufficient to merely guarantee that a fragment contains at least 8 octets of data beyond the IP header because important transport header information (e.g., the CODE field of the TCP header) might be beyond the 8th data octet.

3.1 Example of the Tiny Fragment Attack

In this example, the first fragment contains only eight octets of data (the minimum fragment size). In the case of TCP, this is sufficient to contain the source and destination port numbers, but it will force the TCP flags field into the second fragment.

Filters that attempt to drop connection requests (TCP datagrams having SYN=1 and ACK=0) will be unable to test these flags in the first octet, and will typically ignore them in subsequent fragments.

FRAGMENT 1

```
IP HEADER
+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+
|     | ... | Fragment Offset = 0 | ... |     |
+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+

TCP HEADER
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Source Port            |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
FRAGMENT 2

IP HEADER
+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+
|     | ... | Fragment Offset = 1 | ... |     |
+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+

TCP HEADER
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2 Prevention of the Tiny Fragment Attack

In a router, one can prevent this sort of attack by enforcing certain limits on fragments passing through, namely, that the first fragment be large enough to contain all the necessary header information.

There are two ways to guarantee that the first fragment of a "passed" packet includes all the required fields, one direct, the other indirect.

3.2.1 Direct Method

There is some number TMIN which is the minimum length of a transport header required to contain "interesting" fields (i.e., fields whose values are significant to packet filters). This length is measured from the beginning of the transport header in the original unfragmented IP packet.

Note that TMIN is a function of the transport protocol involved and also of the particular filters currently configured.

The direct method involves computing the length of the transport header in each zero-offset fragment and comparing it against TMIN. If the transport header length is less than TMIN, the fragment is discarded. Non-zero-offset fragments need not be checked because if the zero-offset fragment is discarded, the destination host will be unable to complete reassembly. So far we have:

```
if FO=0 and TRANSPORTLEN < tmin then
        DROP PACKET
```

However, the "interesting" fields of the common transport protocols, except TCP, lie in the first eight octets of the transport header, so it isn't possible to push them into a non-zero-offset fragment. Therefore, as of this writing, only TCP packets are vulnerable to tiny-fragment attacks and the test need not be applied to IP packets carrying other transport protocols. A better version of the tiny fragment test might therefore be:

```
if FO=0 and PROTOCOL=TCP and TRANSPORTLEN < tmin then
        DROP PACKET
```

As discussed in the section on overlapping fragments below, however, this test does not block all fragmentation attacks, and is in fact unnecessary when a more general technique is used.

3.2.2 Indirect Method

The indirect method relies on the observation that when a TCP packet is fragmented so as to force "interesting" header fields out of the zero-offset fragment, there must exist a fragment with FO equal to 1.

If a packet with FO==1 is seen, conversely, it could indicate the presence, in the fragment set, of a zero-offset fragment with a transport header length of eight octets Discarding this one-offset fragment will block reassembly at the receiving host and be as effective as the direct method described above.

4. Overlapping Fragment Attack

RFC 791, the current IP protocol specification, describes a reassembly algorithm that results in new fragments overwriting any overlapped portions of previously-received fragments.

Given such a reassembly implementation, an attacker could construct a series of packets in which the lowest (zero-offset) fragment would contain innocuous data (and thereby be passed by administrative packet filters), and in which some subsequent packet having a non-zero offset would overlap TCP header information (destination port, for instance) and cause it to be modified. The second packet would be passed through most filter implementations because it does not have a zero fragment offset.

RFC 815 outlines an improved datagram reassembly algorithm, but it concerns itself primarily with filling gaps during the reassembly process. This RFC remains mute on the issue of overlapping fragments.

Thus, fully-compliant IP implementations are not guaranteed to be immune to overlapping-fragment attacks. The 4.3 BSD reassembly implementation takes care to avoid these attacks by forcing data from lower-offset fragments to take precedence over data from higher-offset fragments. However, not all IP implementations are based on the original BSD code, and it is likely that some of them are vulnerable.

4.1 Example of the Overlapping Fragment Attack

In this example, fragments are large enough to satisfy the minimum size requirements described in the previous section. The filter is configured to drop TCP connection request packets.

The first fragment contains values, e.g., SYN=0, ACK=1, that enable it to pass through the filter unharmed.

The second fragment, with a fragment offset of eight octets, contains TCP Flags that differ from those given in the first fragment, e.g., SYN=1, ACK=0. Since this second fragment is not a 0-offset fragment, it will not be checked, and it, too will pass through the filter.

The receiving host, if it conforms fully to the algorithms given in RFC 791, will reconstitute the packet as a connection request because the "bad" data arrived later.

```
      FRAGMENT 1

      IP HEADER
      +-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+
      |     | ... | Fragment Offset = 0 | ... |     |
      +-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+

      TCP HEADER
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |        Source Port            |       Destination Port        |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Sequence Number                         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                    Acknowledgment Number                      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |  Data |           |U|A|P|R|S|F|                               |
      | Offset| Reserved  |R|C|S|S|Y|I|            Window             |
      |       |           |G|K|H|T|N|N|                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   .
                                   .
                                   .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                        (Other data)                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


      FRAGMENT 2

      IP HEADER
      +-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+
      |     | ... | Fragment Offset = 1 | ... |     |
      +-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+

      TCP HEADER
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                    Acknowledgment Number                      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |  Data |           |U|A|P|R|S|F|                               |
      | Offset| Reserved  |R|C|S|S|Y|I|            Window             |
      |       |           |G|K|H|T|N|N|                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   .
                                   .
                                   .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                        (Other data)                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If the receiving host has a reassembly algorithm that prevents new data from overwriting data received previously, we can send Fragment 2 first, followed by Fragment 1, and accomplish the same successful attack.

4.2 Prevention of the Overlapping Fragment Attack

Since no standard requires that an overlap-safe reassembly algorithm be used, the potential vulnerability of hosts to this attack is quite large.

By adopting a better strategy in a router's IP filtering code, one can be assured of blocking this "attack". If the router's filtering module enforces a minimum fragment offset for fragments that have non-zero offsets, it can prevent overlaps in filter parameter regions of the transport headers.

In the case of TCP, this minimum is sixteen octets, to ensure that the TCP flags field is never contained in a non-zero-offset fragment. If a TCP fragment has FO==1, it should be discarded because it starts only eight octets into the transport header. Conveniently, dropping FO==1 fragments also protects against the tiny fragment attack, as discussed earlier.

RFC 791 demands that an IP stack must be capable of passing an 8 byte IP data payload without further fragmentation (fragments sit on 8 byte boundaries). Since an IP header can be up to 60 bytes long (including options), this means that the minimum MTU on a link should be 68 bytes.

A typical IP header is only 20 bytes long and can therefore carry 48 bytes of data. No one in the real world should EVER be generating a TCP packet with FO=1, as it would require both that a previous system fragmenting IP data down to the 8 byte minimum and a 60 byte IP header.

A general algorithm, then, for ensuring that filters work in the face of both the tiny fragment attack and the overlapping fragment attack is:

```
IF FO=1 and PROTOCOL=TCP then
        DROP PACKET
```

If filtering based on fields in other transport protocol headers is provided in a router, the minimum could be greater, depending on the position of those fields in the header. In particular, if filtering is permitted on data beyond the sixteenth octet of the transport header, either because of a flexible user interface or the implementation of filters for some new transport protocol, dropping packets with FO==1 might not be sufficient.

5. Security Considerations

This memo is concerned entirely with the security implications of filtering fragmented IP packets.

6. Acknowledgements

The attack scenarios described above grew from discussions that took place on the firewalls mailing list during May of 1995. Participants included: Darren Reed <avalon@coombs.anu.edu.au>, Tom Fitzgerald <fitz@wang.com>, and Paul Traina <pst@cisco.com>.

7. References

[1] Mogul, J., "Simple and Flexible Datagram Access Controls for Unix-based Gateways", Digital Equipment Corporation, March 1989.

[2] Postel, J., Editor, "Internet Protocol - DARPA Internet Program Protocol Specification", STD 5, RFC 791, USC/Information Sciences Institute, September 1981.

[3] Postel, J., Editor, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, USC/Information Sciences Institute, September 1981.

[4] Clark, D., "IP Datagram Reassembly Algorithms", RFC 815, MIT Laboratory for Computer Science/Computer Systems and Communications Group, July 1982.

Authors’ Addresses

G. Paul Ziemba
Alantec
2115 O’Nel Drive
San Jose, CA 95131

EMail: paul@alantec.com

Darren Reed
Cybersource
1275A Malvern Rd
Melbourne, Vic 3144
Australia

EMail: darrenr@cyber.com.au

Paul Traina
cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, CA 95028

EMail: pst@cisco.com