IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 2460 (if approved)
Intended status: Standards Track
Expires: July 13, 2013

F. Gont
SI6 Networks / UTN-FRH

January 9, 2013

Security Implications of Predictable Fragment Identification Values
draft-gont-6man-predictable-fragment-id-03

Abstract

IPv6 specifies the Fragment Header, which is employed for the fragmentation and reassembly mechanisms. The Fragment Header contains an "Identification" field which, together with the IPv6 Source Address and the IPv6 Destination Address of the packet, identifies fragments that correspond to the same original datagram, such that they can be reassembled together at the receiving host. The only requirement for setting the "Identification" value is that it must be different than that of any other fragmented packet sent recently with the same Source Address and Destination Address. Some implementations simply use a global counter for setting the Fragment Identification field, thus leading to predictable values. This document analyzes the security implications of predictable Identification values, and updates RFC 2460 specifying additional requirements for setting the Fragment Identification, such that the aforementioned security implications are mitigated.

Table of Contents

1. Introduction

IPv6 specifies the Fragment Header, which is employed for the fragmentation and reassembly mechanisms. The Fragment Header contains an "Identification" field which, together with the IPv6 Source Address and the IPv6 Destination Address of the packet, identifies fragments that correspond to the same original datagram, such that they can be reassembled together at the receiving host. The only requirement for setting the "Identification" value is that it must be different than that of any other fragmented packet sent recently with the same Source Address and Destination Address.

The most trivial algorithm to avoid reusing Fragment Identification values too quickly is to maintain a global counter that is incremented for each fragmented packet that is sent. However, this trivial algorithm leads to predictable Identification values, which can be leveraged for performing a variety of attacks.

Section 2 of this document analyzes the security implications of predictable Identification values. Section 3 updates RFC 2460 by adding the requirement that Identification values not be predictable by an off-path attacker. Section 4 discusses constraints in the possible algorithms for selecting Fragment Identification values. Section 5 specifies a number of algorithms that could be used for generating Identification values. Finally, Appendix B contains a survey of the Fragment Identification algorithms employed by popular IPv6 implementations.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Security Implications of Predictable Fragment Identification values

Predictable Identification values result in an information leakage that can be exploited in a number of ways. Among others, they may potentially be exploited to:

- determine the packet rate at which a given system is transmitting information,
- perform stealth port scans to a third-party,
- uncover the rules of a number of firewalls,
- count the number of systems behind a middle-box, or,
- perform a Denial of Service (DoS) attack

[CPNI-IPv6] contains a detailed analysis of possible vulnerabilities introduced by predictable Fragment Identification values. In summary, their security implications are very similar to those of predictable Identification values in IPv4.

> [Sanfilippo1998a] originally pointed out how the IPv4 Identification field could be examined to determine the packet rate at which a given system is transmitting information. Later, [Sanfilippo1998b] described how a system with such an implementation could be used to perform a stealth port scan to a third (victim) host. [Sanfilippo1999] explained how to exploit this implementation strategy to uncover the rules of a number of firewalls. [Bellovin2002] explains how the IPv4 Identification field can be exploited to count the number of systems behind a NAT. [Fyodor2004] is an entire paper on most (if not all) the ways to exploit the information provided by the Identification field of the IPv4 header (and these results apply in a similar way to IPv6). [RFC6274] covers the security implications of IPv4 in detail.

One key difference between the IPv4 case and the IPv6 case is that in IPv4 the Identification field is part of the fixed IPv4 header (and thus usually set for all packets), while in IPv6 the Identification field is set only in those packets that employ a Fragment Header. As a result, successful exploitation of the Identification field against communication instances with arbitrary destinations depends on two different factors:

- IPv6 implementations using predictable Identification values, and,

- the ability of the attacker to cause the victim host to fragment packets destined to other nodes

As noted in the previous section, some implementations are known to use predictable identification values.

> For example, Linux 2.6.38-8 sets the Identification field according to a global counter that is incremented by one for each datagram that is sent with a fragment header (either a single fragment or as multiple fragments).

Finally, we note that an attacker could cause a victim host to fragment its outgoing packets by sending it a forged ICMPv6 'Packet Too Big' error message advertising a Next-Hop MTU smaller than 1280 bytes.

> RFC 1981 [RFC1981] states that when an ICMPv6 Packet Too Big error message with an MTU smaller than 1280 bytes is received, the receiving host is not required to reduce the Path-MTU for the corresponding destination address, but must simply include a Fragment Header in all subsequent packets sent to that destination. In order to make sure that the forged ICMPv6 Packet Too Big error message triggers fragmentation at the victim host, the attacker could set the MTU field of the error message to a value smaller than 1280 bytes. Since the minimum IPv6 MTU is 1280 bytes, such value would always be smaller than the Path-MTU in use for that destination.

There are a few issues that should be considered, though:

- In all the implementations the author is aware of, an attacker can only cause the victim to enable fragmentation on a per-destination basis. That is, the victim will use fragmentation only for those packets sent to the Source Address of IPv6 packet embedded in the payload of the ICMPv6 Packet Too Big error message.

  > Section 5.2 of [RFC1981] notes that an implementation could maintain a single system-wide PMTU value to be used for all packets originating from that nodes. Clearly, such an implementations would exacerbate the problem of any attacks based on PMTUD [RFC5927] or IPv6 fragmentation.

- If the victim node implements some of the counter-measures for ICMP attacks described in RFC 5927 [RFC5927], it might be difficult for an attacker to cause the victim node to use fragmentation for its outgoing packets.

Some implementations do not incorporate countermeasures for attacks based on ICMPv6 error messages. For example, Linux 2.6.38-8 does not even require received ICMPv6 error messages to correspond to ongoing communication instances.

Implementations that employ predictable Identification values and also fail to include countermeasures against attacks based on ICMPv6 error messages will be vulnerable to attacks similar to those based on the IPv4 Identification field for IPv4 networks, such as the stealth port-scanning technique described in [Sanfilippo1998b].

One possible way in which predictable Identification values could be leveraged for performing a Denial of Service (DoS) attack is as follows: once the Identification value currently in use at the victim host has been learned, the attacker would send a forged ICMPv6 Packet Too Big error message to the victim host, with the IPv6 Destination Address of the embedded IPv6 packet set to the IPv6 address of a third-party host with which the victim is communicating. This ICMPv6 Packet Too Big error message would cause any packets sent from the victim to the third-party host to include a Fragment Header. The attacker would then send forged IPv6 fragments to the third-party host, with their IPv6 Source Address set to that of the victim host, and with the Identification field of the forged fragments set to values that would result in collisions at the third-party host. If the third-party host discards fragments that result in collisions of Identification values, the attacker could simply trash the Identification space by sending multiple forged fragments with different Identification values, such that any subsequent packets from the victim host are discarded at the third-party host as a result of the malicious fragments sent by the attacker.

For example, Linux 2.6.38-10 is vulnerable to the aforementioned issue.

[I-D.ietf-6man-ipv6-atomic-fragments] describes an improved processing of these packets that would eliminate this specific attack vector, at least in the case of TCP connections that employ the Path-MTU Discovery mechanism.

The aforementioned attack scenario is simply included to illustrate the problem of employing predictable fragment Identification values, rather than to indicate a specific attack vector that needs to be mitigated.

We note that regardless of the attacker's ability to cause a victim host to employ fragmentation when communicating with third-parties, use of predictable Identification values makes communication flows that employ fragmentation vulnerable to any fragmentation-based

attacks.

3. Updating RFC 2460

Hereby we update RFC 2460 [RFC2460] as follows:

The Identification value of the Fragment Header MUST NOT be predictable by an off-path attacker.

4. Constraints for the selection of Fragment Identification Values

the "Identification" field of the Fragmentation Header is 32-bits long. However, when translators [RFC6145] are employed, the "effective" length of the IPv6 Fragment Identification field is 16 bits.

> [RFC6145] notes that, when translating in the IPv6-to-IPv4 direction, "if there is a Fragment Header in the IPv6 packet, the last 16 bits of its value MUST be used for the IPv4 identification value". This means that the high-order 16 bits are effectively ignored.

As a result, at least during the IPv6/IPv4 transition/co-existence phase, it is probably safer to assume that only the last 16 bits of the IPv6 Fragment Identification may be used in some cases.

Regarding the selection of Fragment Identification values, the only requirement specified in [RFC2460] is that the Fragment Identification must be different than that of any other fragmented packet sent recently with the same Source Address and Destination Address.

> Failure to comply with that requirement might lead to the interoperability problems discussed in [RFC4963].

From a security standpoint, unpredictable Identification values are desirable. However, this is somewhat at odds with the "re-use" requirements specified in [RFC2460].

Finally, since Fragment Identification values need to be selected for each outgoing datagram that requires fragmentation, the performance aspect should be considered when choosing an algorithm for the selection of Fragment Identification values.

5. Algorithms for Selecting Fragment Identification Values

This section specifies a number of algorithms that MAY be used for selecting Fragment Identification values.

5.1. Per-destination counter (initialized to a random value)

1. Whenever a packet must be sent with a Fragment Header, the sending host should perform a look-up in the Destinations Cache an entry corresponding to the intended Destination Address.

2. If such an entry exists, it contains the last Fragment Identification value used for that Destination. Therefore, such value should be incremented by 1, and used for setting the Fragment Identification value of the outgoing packet. Additionally, the updated value should be recorded in the corresponding entry of the Destination Cache.

3. If such an entry does not exist, it should be created, and the "Identification" value for that destination should be initialized with a random value (e.g., with a pseudorandom number generator), and used for setting the Identification field of the Fragment Header of the outgoing packet.

The advantages of this algorithm are:

- It is simple to implement, with the only complexity residing in the Pseudo-Random Number Generator (PRNG) used to initialize the "Identification" value contained in each entry of the Destinations Cache.

- The "Identification" re-use frequency will typically be lower than that achieved by a global counter (when sending traffic to multiple destinations), since this algorithm uses per-destination counters (rather than a single system-wide counter).

- It has good performance properties (once the corresponding entry in the Destinations Cache has been created, each subsequent "Identification" value simply involves the increment of a counter).

The possible drawbacks of this algorithm are:

- If as a result of resource management an entry of the Destinations Cache must be removed, the last Fragment Identification value used for that Destination is obviously lost. Thus, if subsequent traffic to that destination causes the aforementioned entry to be re-created, the Fragment Identification value will be randomized,

thus possibly leading to Fragment Identification "collisions".

- Since the Fragment Identification values are predictable by the destination host, a vulnerable host might possible leak to third-parties the Fragment Identification values used by other hosts to send traffic to it (i.e., Host B could leak to Host C the Fragment Identification values that Host A is using to send packets to Host B).

  Appendix A describes a scenario in which that information leakage could take place.

5.2. Randomized Identification values

Clearly, use of a Pseudo-Random Number Generator for selecting the Fragment Identification could be desirable from a security standpoint. With such a scheme, the Fragment Identification of each fragmented datagram would be selected as:

```
Identification = random()
```

where "random()" is the PRNG.

The specific properties of such scheme would clearly depend on the specific PRNG algorithm used. For example, some PRNGs may result in higher Fragment Identification reuse frequencies than others, in the same way as some PRNGs may be more expensive (in terms of processing requirements and/or implementation complexity) than others.

Discussion of the properties of possible PRNGs is considered out of the scope of this document. However, we do note that some PRNGs employed in the past by some implementations have been found to be predictable [Klein2007]. Please see [RFC4086] for randomness requirements for security.

5.3. Hash-based Fragment Identification selection algorithm

Another alternative is to implement a hash-based algorithm similar to that specified in for the selection of transport port numbers. With such a scheme, the Fragment Identification value of each fragment datagram would be selected with the expression:

```
Identification = F(Src IP, Dst IP, secret1)  +
                 counter[G(src IP, Dst Pref, secret2)]
```

where:

Identification:
Identification value to be used for the fragmented datagram

F():
Hash function

Src IP:
IPv6 Source Address of the datagram to be fragmented

Dst IP:
IPv6 Destination Address of the datagram to be fragmented

secret1:
Secret data unknown to the attacker

counter[]:
System-wide array of 32-bit counters (e.g. with 8K elements or more)

G():
Hash function. May or may not be the same hash function as that used for F()

Dst Pref:
IPv6 "Destination Prefix" of datagram to be fragmented (can be assumed to be the first eight bytes of the Destination Address of such packet). Note: the "Destination Prefix" (rather than Destination Address) is used, such that the ability of an attacker of searching the "increments" space by using multiple addresses of the same subnet is reduced.

secret1:
Secret data unknown to the attacker

Note: counter[G(src IP, Dst Pref, secret2)] should be incremented by one each time an Identification value is selected.

The advantages of this algorithm are:

- The "Identification" re-use frequency will typically be lower than that achieved by a global counter (when sending traffic to multiple destinations), since this algorithm uses multiple system-wide counters (rather than a single system-wide counter). The extent to which the re-use frequency will be lower will depend on the number of elements in counter[], and the number of other active flows that result in the same value of G() (and hence cause the same counter to be incremented for each fragmented datagram that is sent).

- It is possible to implement the algorithm such that good performance is achieved. For example, the result of F() could be stored in the Destinations Cache (such that it need not be recomputed for each packet that must be sent) along with computed *index* for counter[].

  It should be noted that if this implementation approach is followed, and an entry of the Destinations Cache must be removed as a result of resource management, the last Fragment Identification value used for that Destination will *not* lost. This is an improvement over the algorithm specified in Section 5.1.

The possible drawbacks of this algorithm are:

- Since the Fragment Identification values are predictable by the destination host, a vulnerable host could possibly leak to third-parties the Fragment Identification values used by other hosts to send traffic to it (i.e., Host B could leak to Host C the Fragment Identification values that Host A is using to send packets to Host B).

  Appendix A describes a scenario in which that information leakage could take place. We note, however, that this algorithm makes the aforementioned attack less reliable for the attacker, since each counter could be possibly shared by multiple traffic flows (i.e., packets destined to other destinations might cause the counter to be incremented).

This algorithm might be preferable (over the one specified in Section 5.1) in those scenarios in which a node is expected to communicate with a large number of destinations, and thus it is desirable to limit the amount of information to be maintained in memory.

In such scenarios, if the algorithm specified in Section 5.1 were implemented, entries from the Destinations Cache might need to be pruned frequently, thus increasing the risk of fragment Identification collisions.

6. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

7. Security Considerations

This document discusses the security implications of predictable Fragment Identification values, and updates RFC 2460 such that Fragment Identification values are required to be unpredictable by off-path attackers, hence mitigating the aforementioned security implications.

A number of possible algorithms are specified, to provide some implementation alternatives to implementers. However, the selection of an specific algorithm that complies with Section 3 is left to implementers. We note that the selection of such an algorithm usually implies a number of trade-offs (security, performance, implementation complexity, interoperability properties, etc.).

8. Acknowledgements

The author would like to thank Ivan Arce for proposing the attack scenario described in Appendix A, and for providing valuable comments on earlier versions of this document.

The author would like to thank Dave Thaler for providing valuable comments on earlier versions of this document.

This document is based on the technical report "Security Assessment of the Internet Protocol version 6 (IPv6)" [CPNI-IPv6] authored by Fernando Gont on behalf of the UK Centre for the Protection of National Infrastructure (CPNI).

Fernando Gont would like to thank the UK CPNI (http://www.cpni.gov.uk) for their continued support.

9. References

9.1. Normative References

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, December 2009.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

9.2. Informative References

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, July 2010.

[RFC6274] Gont, F., "Security Assessment of the Internet Protocol Version 4", RFC 6274, July 2011.

[I-D.ietf-6man-ipv6-atomic-fragments]
Gont, F., "Processing of IPv6 "atomic" fragments", draft-ietf-6man-ipv6-atomic-fragments-03 (work in progress), December 2012.

[Bellovin2002]
Bellovin, S., "A Technique for Counting NATted Hosts", IMW'02 Nov. 6-8, 2002, Marseille, France, 2002.

[CPNI-IPv6]
Gont, F., "Security Assessment of the Internet Protocol version 6 (IPv6)", UK Centre for the Protection of National Infrastructure, (available on request).

[Fyodor2004]
Fyodor, "Idle scanning and related IP ID games", 2004,

<http://www.insecure.org/nmap/idlescan.html>.

[Klein2007]
Klein, A., "OpenBSD DNS Cache Poisoning and Multiple O/S Predictable IP ID Vulnerability", 2007, <http://www.trusteer.com/files/OpenBSD_DNS_Cache_Poisoning_and_Multiple_OS_Predictable_IP_ID_Vulnerability.pdf>.

[Sanfilippo1998a]
Sanfilippo, S., "about the ip header id", Post to Bugtraq mailing-list, Mon Dec 14 1998, <http://www.kyuzz.org/antirez/papers/ipid.html>.

[Sanfilippo1998b]
Sanfilippo, S., "Idle scan", Post to Bugtraq mailing-list, 1998, <http://www.kyuzz.org/antirez/papers/dumbscan.html>.

[Sanfilippo1999]
Sanfilippo, S., "more ip id", Post to Bugtraq mailing-list, 1999, <http://www.kyuzz.org/antirez/papers/moreipid.html>.

[SI6-IPv6]
"SI6 Networks' IPv6 toolkit", <http://www.si6networks.com/tools/ipv6toolkit>.

Appendix A. Information leakage produced by vulnerable implementations

Section 2 provides a number of references describing a number of ways in which the information leakage produced by a vulnerable implementation could be leveraged by an attacker. This section describes a specific network scenario in which a vulnerable implementation could possibly leak the current Fragment Identification value in use by a third-party host to send fragmented datagrams to the vulnerable implementation.

For the most part, this section is included to illustrate how a vulnerable implementation might be leveraged to leak-out the Fragment Identification value of an otherwise secure implementation. This section might be removed in future revisions of this document.

The following scenarios assume:

A:
Is an IPv6 host that implements the recommended Fragment Identification algorithm (Section 5.1), implements [RFC5722], but does not implement [I-D.ietf-6man-ipv6-atomic-fragments].

B:
Victim node. Selected the Fragment Identification values from a global counter.

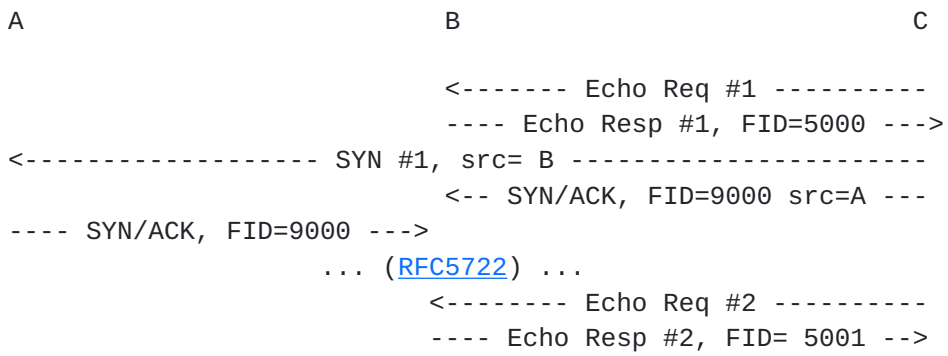
C:
Attacker. Can forge the IPv6 Source Address of his packets at will.

If the attacker sends forged SYN packets to a closed TCP port, and then fails when trying to produce a collision of Fragment Identifications (see line #4), the following packet exchange might take place:

```
      A                          B                              C

   #1                             <------ Echo Req #1 ----------
   #2                             --- Echo Resp #1, FID=5000 --->
   #3  <------------------- SYN #1, src= B ----------------------
   #4                             <---- SYN/ACK, FID=42 src = A---
   #5  ---- SYN/ACK, FID=9000 --->
   #6  <---- RST, FID= 5001 -----
   #7  <---- RST, FID= 5002 -----
   #8                            <-------- Echo Req #2 ----------
   #9                            ---- Echo Resp #2, FID= 5003 -->
```

On the other hand, if the attacker succeeds to produce a collision of Fragment Identification values, the following packet exchange could take place:



Clearly, the Fragment Identification value sampled by from the second ICMPv6 Echo Response packet ("Echo Resp #2") implicitly indicates whether the Fragment Identification in the forged SYN/ACK (see line #4 in both figures) was the current Fragment Identification in use by Host A.

As a result, the attacker could employ this technique to learn the current Fragment Identification value used by host A to send packets to host B.

Appendix B. Survey of Fragment Identification selection algorithms employed by popular IPv6 implementations

This section includes a survey of the Fragment Identification selection algorithms employed in some popular operating systems.

The survey was produced with the SI6 Networks IPv6 toolkit [SI6-IPv6].

Operating System	Algorithm
FreeBSD 9.0	Unpredictable (Random)
Linux 3.0.0-15	Predictable (Global Counter, Init=0, Incr=1)
Linux-current	Unpredictable (Per-dest Counter, Init=random, Incr=1)
NetBSD 5.1	Unpredictable (Random)
OpenBSD-current	Random (SKIP32)
Solaris 10	Predictable (Per-dst Counter, Init=0, Incr=1)
Windows XP SP2	Predictable (Global Counter, Init=0, Incr=2)
Windows Vista (Build 6000)	Predictable (Global Counter, Init=0, Incr=2)
Windows 7 Home Premium	Predictable (Global Counter, Init=0, Incr=2)

Table 1: Fragment Identification algorithms employed by different OSes

In the text above, "predictable" should be taken as "easily guessable by an off-path attacker, by sending a few probe packets".

Author's Address

Fernando Gont
SI6 Networks / UTN-FRH
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: http://www.si6networks.com