IPv6 maintenance Working Group (6man)
Internet-Draft
Intended status: Informational
Expires: February 26, 2016

F. Gont
SI6 Networks / UTN-FRH

August 25, 2015

Security Implications of Predictable Fragment Identification Values
draft-ietf-6man-predictable-fragment-id-09

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 a packet, identifies fragments that correspond to the same original datagram, such that they can be reassembled together by the receiving host. The only requirement for setting the "Identification" field is that the corresponding value must be different than that employed for any other fragmented packet sent recently with the same Source Address and Destination Address. Some implementations use a simple global counter for setting the Identification field, thus leading to predictable Identification values. This document analyzes the security implications of predictable Identification values, and provides implementation guidance for selecting the Identification field of the Fragment Header, such that the aforementioned security implications are mitigated.

Status of This Memo

Copyright Notice

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 a 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 employed for 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 transmitted. However, this trivial algorithm leads to predictable Identification values, which can be leveraged to perform a variety of attacks.

Section 3 of this document analyzes the security implications of predictable Identification values. 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.

2. Terminology

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].

3. 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:

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

The security implications introduced by predictable Fragment Identification values 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] explains 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). [Zalewski2003] originally envisioned the exploitation of IP fragmentation/reassembly for performing data injection attacks against upper-layer protocols. [Herzberg2013] explores the use of IPv4/IPv6 fragmentation and predictable Identification values for performing DNS cache poisoning attacks in great detail. [RFC6274] covers the security implications of the IPv4 case 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 present only in those packets that carry a Fragment Header. As a result, successful exploitation of the IPv6 Fragment Identification field depends on two different factors:

o vulnerable IPv6 Fragment Identification generators, and,

o the ability of an attacker to trigger the use of IPv6 fragmentation for packets sent from/to the victim node

The scenarios in which an attacker may successfully perform the aforementioned attacks depend on the specific attack type. For example, in order to DoS communications between two hosts, an attacker would need to know the IPv6 addresses employed by the aforementioned two nodes. Such knowledge may be readily available if the target of the attack is the communication between two specific BGP peers, two specific SMTP servers, or one specific primary DNS server and one of its secondary DNS servers, but may not be easily available if goal of the attack is to DoS all communications between arbitrary IPv6 hosts (e.g. the goal was to DoS all communications involving one specific node with arbitrary/unknown hosts). Other attacks, such as performing stealth port scans to a third-party or determining the packet rate at which a given system is transmitting information, only require the attacker to know the IPv6 address of a vulnerable implementation.

As noted in the previous section, some implementations have been known to use predictable Fragment Identification values. For instance, Appendix B of this document shows that recent versions of a number of popular IPv6 implementations employ predictable values for the IPv6 Fragment Identification.

Additionally, we note that [RFC2460] states that when an ICMPv6 Packet Too Big error message advertising 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. This triggers the use of the so-called IPv6 "atomic fragments" [RFC6946]: IPv6 fragments with a Fragment Offset equal to 0, and the "M" ("More fragments") bit clear. [I-D.ietf-6man-deprecate-atomfrag-generation] aims at deprecating the generation of IPv6 atomic fragments.

Thus, an attacker can usually cause a victim host to "fragment" its outgoing packets by sending it a forged ICMPv6 'Packet Too Big' (PTB) error message that advertises an MTU smaller than 1280 bytes.

There are a number of aspects that should be considered, though:

o All the implementations the author is aware of record the Path-MTU information on a per-destination basis. Thus, an attacker can only cause the victim to enable fragmentation for those packets sent to the Source Address of IPv6 packet embedded in the payload of the ICMPv6 PTB message. However, we note that Section 5.2 of [RFC1981] notes that an implementation could maintain a single system-wide PMTU value to be used for all packets sent to that node. Clearly, such implementations would exacerbate the problem of any attacks based on PMTUD [RFC5927] or IPv6 fragmentation.

o 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 employ fragmentation for its outgoing packets. However, many current implementations fail to enforce these validation checks. For example, Linux 2.6.38-8 does not even require received ICMPv6 error messages to correspond to an ongoing communication instance.

o Some implementations (notably Linux) have already been updated according to [I-D.ietf-6man-deprecate-atomfrag-generation] such that ICMPv6 PTB messages do not result in the generation of IPv6 atomic fragments.

Implementations that employ predictable Identification values and also fail to enforce validation checks on ICMPv6 error messages

become vulnerable to the same type of attacks that can be exploited with IPv4 fragmentation, discussed earlier in this section.

One possible way in which predictable Identification values could be leveraged for performing a Denial of Service (DoS) attack is as follows: Let us assume that Host A is communicating with Host B, and that an attacker wants to DoS attack such communication. The attacker would learn the the Identification value currently in use by Host A, possibly by sending any packet that would elicit a fragmented response (e.g., an ICPMv6 echo request with a large payload). The attacker would then send a forged ICMPv6 Packet Too Big error message to Host A (with the IPv6 Destination Address of the embedded IPv6 packet set to the IPv6 address of a Host B), such that any subsequent packets sent by Host A to Host B include a Fragment Header. Finally, the attacker would send forged IPv6 fragments to Host B, with their IPv6 Source Address set to that of Host A, and Identification values that would result in collisions with the Identification values employed for the legitimate traffic sent by Host A to Host B. If Host B discards fragments that result in collisions of Identification values (e.g., such fragments overlap, and the host implements [RFC5722]), the attacker could simply trash the Identification space by sending multiple forged fragments with different Identification values, such that any subsequent packets from Host A to Host B are discarded at Host B as a result of the malicious fragments sent by the attacker.

NOTES:

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

[RFC6946] 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. 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.

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 high-order 16 bits of the Identification field are effectively ignored.

NOTES: [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".

Additionally, Section 3.3 of [RFC6052] encourages operators to use a Network-Specific Prefix (NSP) that maps the IPv4 address space into IPv6. Thus, when an NSP is being used, IPv6 addresses representing IPv4 nodes (reached through a stateless translator) are indistinguishable from native IPv6 addresses.

Thus, when translators are employed, the "effective" length of the IPv6 Fragment Identification field is 16 bits and, as a result, at least during the IPv6/IPv4 transition/co-existence phase, it is probably safer to assume that only the low-order 16 bits of the IPv6 Fragment Identification are of use to the destination system.

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 this requirement could 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], that specifies that an Identification value must be different than that of any other fragment sent recently.

Finally, since Fragment Identification values need to be selected for each outgoing datagram that requires fragmentation, the performance impact 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 look-up in the Destinations Cache an entry corresponding to the Destination Address of the packet.

2. If such an entry exists, it contains the last Fragment Identification value used for that Destination Address.

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 [RFC4861].

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 fragmented datagram.

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 and initialized, 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 will be lost. Thus, subsequent traffic to that destination would cause that entry to be re-created and re-initialized to random value, thus possibly leading to Fragment Identification "collisions".

- Since the Fragment Identification values are predictable by the destination host, a vulnerable host might 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 one possible scenario for such leakage in detail.

5.2. Randomized Identification values

Clearly, use of a Pseudo-Random Number Generator for selecting the Fragment Identification would 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 employed. For example, some PRNGs may result in higher Fragment Identification reuse frequencies than others, in the same way that 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 [RFC6056] 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. This value can be initialized to a pseudo-random value during the system bootstrapping sequence. It should remain constant at least while there could be previously-sent fragments still in the network or at the fragment reassembly buffer of the corresponding destination system(s).

counter[]:
System-wide array of 32-bit counters (e.g. with 8K elements or more). Each counter should be initialized to a pseudo-random value during the system bootstrapping sequence.

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.

secret2:
Secret data unknown to the attacker. This value can be initialized to a pseudo-random value during the system bootstrapping sequence. It should remain constant at least while there could be previously-sent fragments still in the network or at the fragment reassembly buffer of the corresponding destination system(s).

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

The output of F() will be constant for each (Src IP, Dst IP) pair. Similarly, the output of G() will be constant for each (Src IP, Dst Pref) pair. Thus, the resulting "Identification" value will be the result of a random offset plus a linear function (provided by counter[]), therefore resulting in a monotonically-increasing sequence of "Identification" values for each (src IP, Dst IP) pair.

NOTE:
F() essentially provides the unpredictability (by off-path attackers) of the resulting "Identification" values, while counter[] provides a linear function such that the "Identification" values are different for each fragmented packet while the "Identification" reuse frequency is minimized.

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 the computed index/argument for counter[].

  NOTE:
  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* be 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 possible 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 same 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.

NOTE: 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 provides implementation guidance such that the aforementioned security implications can be mitigated.

A number of possible algorithms are described, to provide some implementation alternatives 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.

The author would like to thank Ivan Arce, Stephen Bensley, Ron Bonica, Tassos Chatzithomaoglou, Brian Haberman, Bob Hinden, Tatuya Jinmei, Merike Kaeo, Will Liu, Juan Antonio Matos, Simon Perreault, Hosnieh Rafiee, Mark Smith, and Dave Thaler for providing valuable comments on earlier versions of this document.

This document is based on work performed by Fernando Gont on behalf of the UK Centre for the Protection of National Infrastructure (CPNI).

9. References

9.1. Normative References

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, DOI 10.17487/RFC5722, December 2009, <http://www.rfc-editor.org/info/rfc5722>.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <http://www.rfc-editor.org/info/rfc6052>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <http://www.rfc-editor.org/info/rfc6056>.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, DOI 10.17487/RFC6145, April 2011, <http://www.rfc-editor.org/info/rfc6145>.

[RFC6946] Gont, F., "Processing of IPv6 "Atomic" Fragments", RFC 6946, DOI 10.17487/RFC6946, May 2013, <http://www.rfc-editor.org/info/rfc6946>.

9.2. Informative References

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <http://www.rfc-editor.org/info/rfc4963>.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <http://www.rfc-editor.org/info/rfc5927>.

[RFC6274] Gont, F., "Security Assessment of the Internet Protocol Version 4", RFC 6274, DOI 10.17487/RFC6274, July 2011, <http://www.rfc-editor.org/info/rfc6274>.

[I-D.ietf-6man-deprecate-atomfrag-generation]
Gont, F., LIU, S., and T. Anderson, "Deprecating the Generation of IPv6 Atomic Fragments", draft-ietf-6man-deprecate-atomfrag-generation-03 (work in progress), July 2015.

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

[Fyodor2004]
Fyodor, , "Idle scanning and related IP ID games", 2004, <http://www.insecure.org/nmap/idlescan.html>.

[Herzberg2013]
Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", Technical Report 13-03, March 2013, <http://u.cs.biu.ac.il/~herzbea/security/13-03-frag.pdf>.

[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>.

[Zalewski2003]
Zalewski, M., "A new TCP/IP blind data injection technique?", Post to Bugtraq mailing-list, Thu, 11 Dec 2003 00:28:28 +0100 (CET), 2003, <http://lcamtuf.coredump.cx/ipfrag.txt>.

Appendix A. Information leakage produced by vulnerable implementations

Section 3 provides a number of references describing a number of ways in which a vulnerable implementation may reveal the Fragment Identification values to be used in subsequent packets, thus opening the door to a number of attacks. In all of those scenarios, a vulnerable implementation leaks/reveals its own Identification number.

This section presents a different attack scenario, in which a vulnerable implementation leaks/reveals the Identification number of a non-vulnerable implementation. That is, a vulnerable implementation (Host A) leaks the current Fragment Identification value in use by a third-party host (Host B) to send fragmented datagrams from Host B to Host A.

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 non-vulnerable implementation.

The following scenarios assume:

Host A:
An IPv6 host that implements the the algorithm specified in Section 5.1, implements [RFC5722], but does not implement [RFC6946].

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

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

In the following scenarios, large ICMPv6 Echo Request packets are employed to "sample" the Fragment Identification value of a host. We note that while the figures show only one packet for the ICMPv6 Echo Request and the ICMPv6 Echo Response, each of those packets will typically comprise two fragments, such that the corresponding unfragmented datagram is larger than the MTU of the networks to which

Host B and Host C are attached. Additionally, the following scenarios assume that Host A employs a fragment header when sending traffic to Host B (typically the so-called "IPv6 atomic fragments" [RFC6946]): this behavior may be triggered by forged ICMPv6 PTB messages that advertise an MTU smaller than 1280 bytes (assumming the victim does not implement [I-D.ietf-6man-deprecate-atomfrag-generation]).

In lines #1-#2 (and lines #8-#9), the attacker samples the current Fragment Identification value at Host B. In line #3, the attacker sends a forged TCP SYN segment to Host A. If corresponding TCP port is closed, and the attacker 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 --->
```

The two RST segments are elicited by the SYN/ACK segment from line #4, and the (illegitimately elicited by the SYN in line #3) SYN/ACK segment from line #5.

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

```
    A                          B                              C

#1                             <------- Echo Req #1 ----------
#2                             --- Echo Resp #1, FID=5000 --->
#3  <------------------- SYN #1, src= B ----------------------
#4                                  <-- SYN/ACK, FID=9000 src=A ---
#5  ---- SYN/ACK, FID=9000 --->
                        ... (RFC5722) ...
#6                             <------- Echo Req #2 ----------
#7                             ---- Echo Resp #2, FID=5001 -->
```

Clearly, the Fragment Identification value sampled 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, even when Host A itself has a non-vulnerable implementation.

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
Cisco IOS 15.3	Predictable (Global Counter, Init=0, Incr=1)
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 XP Professional 32bit, SP3	Predictable (Global Counter, Init=0, Incr=2)
Windows Vista (Build 6000)	Predictable (Global Counter, Init=0, Incr=2)

Windows Vista Business 64bit, SP1	Predictable (Global Counter, Init=0, Incr=2)
Windows 7 Home Premium	Predictable (Global Counter, Init=0, Incr=2)
Windows Server 2003 R2 Standard 64bit, SP2	Predictable (Global Counter, Init=0, Incr=2)
Windows Server 2008 Standard 32bit, SP1	Predictable (Global Counter, Init=0, Incr=2)
Windows Server 2008 R2 Standard 64bit, SP1	Predictable (Global Counter, Init=0, Incr=2)
Windows Server 2012 Standard 64bit	Predictable (Global Counter, Init=0, Incr=2)
Windows 7 Home Premium 32bit, SP1	Predictable (Global Counter, Init=0, Incr=2)
Windows 7 Ultimate 32bit, SP1	Predictable (Global Counter, Init=0, Incr=2)
Windows 8 Enterprise 32 bit	Unpredictable (Alg. from Section 5.3)

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