Internet Engineering Task Force (IETF)
Request for Comments: 6274
Category: Informational
ISSN: 2070-1721

F. Gont
UK CPNI

July 2011

Security Assessment of the Internet Protocol Version 4

Abstract

This document contains a security assessment of the IETF specifications of the Internet Protocol version 4 and of a number of mechanisms and policies in use by popular IPv4 implementations. It is based on the results of a project carried out by the UK's Centre for the Protection of National Infrastructure (CPNI).

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6274.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Preface

1.1. Introduction

The TCP/IP protocols were conceived in an environment that was quite different from the hostile environment in which they currently operate. However, the effectiveness of the protocols led to their early adoption in production environments, to the point that, to some extent, the current world's economy depends on them.

While many textbooks and articles have created the myth that the Internet protocols were designed for warfare environments, the top level goal for the Defense Advanced Research Projects Agency (DARPA) Internet Program was the sharing of large service machines on the ARPANET [Clark1988]. As a result, many protocol specifications focus only on the operational aspects of the protocols they specify and overlook their security implications.

While the Internet technology evolved since its inception, the Internet's building blocks are basically the same core protocols adopted by the ARPANET more than two decades ago. During the last twenty years, many vulnerabilities have been identified in the TCP/IP stacks of a number of systems. Some of them were based on flaws in some protocol implementations, affecting only a reduced number of systems, while others were based on flaws in the protocols themselves, affecting virtually every existing implementation [Bellovin1989]. Even in the last couple of years, researchers were still working on security problems in the core protocols [RFC5927] [Watson2004] [NISCC2004] [NISCC2005].

The discovery of vulnerabilities in the TCP/IP protocols led to reports being published by a number of CSIRTs (Computer Security Incident Response Teams) and vendors, which helped to raise awareness about the threats and the best mitigations known at the time the

reports were published. Unfortunately, this also led to the documentation of the discovered protocol vulnerabilities being spread among a large number of documents, which are sometimes difficult to identify.

For some reason, much of the effort of the security community on the Internet protocols did not result in official documents (RFCs) being issued by the IETF (Internet Engineering Task Force). This basically led to a situation in which "known" security problems have not always been addressed by all vendors. In addition, in many cases, vendors have implemented quick "fixes" to protocol flaws without a careful analysis of their effectiveness and their impact on interoperability [Silbersack2005].

The lack of adoption of these fixes by the IETF means that any system built in the future according to the official TCP/IP specifications will reincarnate security flaws that have already hit our communication systems in the past.

Nowadays, producing a secure TCP/IP implementation is a very difficult task, in part because of the lack of a single document that serves as a security roadmap for the protocols. Implementers are faced with the hard task of identifying relevant documentation and differentiating between that which provides correct advisory and that which provides misleading advisory based on inaccurate or wrong assumptions.

There is a clear need for a companion document to the IETF specifications; one that discusses the security aspects and implications of the protocols, identifies the possible threats, discusses the possible countermeasures, and analyzes their respective effectiveness.

This document is the result of an assessment of the IETF specifications of the Internet Protocol version 4 (IPv4), from a security point of view. Possible threats were identified and, where possible, countermeasures were proposed. Additionally, many implementation flaws that have led to security vulnerabilities have been referenced in the hope that future implementations will not incur the same problems. Furthermore, this document does not limit itself to performing a security assessment of the relevant IETF specifications, but also provides an assessment of common implementation strategies found in the real world.

Many IP implementations have also been subject of the so-called "packet-of-death" vulnerabilities, in which a single specially crafted packet causes the IP implementation to crash or otherwise misbehave. In most cases, the attack packet is simply malformed; in

other cases, the attack packet is well-formed, but exercises a little used path through the IP stack. Well-designed IP implementations should protect against these attacks, and therefore this document describes a number of sanity checks that are expected to prevent most of the aforementioned "packet-of-death" attack vectors. We note that if an IP implementation is found to be vulnerable to one of these attacks, administrators must resort to mitigating them by packet filtering.

Additionally, this document analyzes the security implications from changes in the operational environment since the Internet Protocol was designed. For example, it analyzes how the Internet Protocol could be exploited to evade Network Intrusion Detection Systems (NIDSs) or to circumvent firewalls.

This document does not aim to be the final word on the security of the Internet Protocol (IP). On the contrary, it aims to raise awareness about many security threats based on the IP protocol that have been faced in the past, those that we are currently facing, and those we may still have to deal with in the future. It provides advice for the secure implementation of the Internet Protocol (IP), but also provides insights about the security aspects of the Internet Protocol that may be of help to the Internet operations community.

Feedback from the community is more than encouraged to help this document be as accurate as possible and to keep it updated as new threats are discovered.

This document is heavily based on the "Security Assessment of the Internet Protocol" [CPNI2008] released by the UK Centre for the Protection of National Infrastructure (CPNI), available at http://www.cpni.gov.uk/Products/technicalnotes/3677.aspx.

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].

1.2. Scope of This Document

While there are a number of protocols that affect the way in which IP systems operate, this document focuses only on the specifications of the Internet Protocol (IP). For example, routing and bootstrapping protocols are considered out of the scope of this project.

The following IETF RFCs were selected as the primary sources for the assessment as part of this work:

- RFC 791, "INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION" (45 pages).
- RFC 815, "IP DATAGRAM REASSEMBLY ALGORITHMS" (9 pages).
- RFC 919, "BROADCASTING INTERNET DATAGRAMS" (8 pages).
- RFC 950, "Internet Standard Subnetting Procedure" (18 pages)
- RFC 1112, "Host Extensions for IP Multicasting" (17 pages)
- RFC 1122, "Requirements for Internet Hosts -- Communication Layers" (116 pages).
- RFC 1812, "Requirements for IP Version 4 Routers" (175 pages).
- RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers" (20 pages).
- RFC 2475, "An Architecture for Differentiated Services" (36 pages).
- RFC 3168, "The Addition of Explicit Congestion Notification (ECN) to IP" (63 pages).
- RFC 4632, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan" (27 pages).

1.3. Organization of This Document

This document is basically organized in two parts: "Internet Protocol header fields" and "Internet Protocol mechanisms". The former contains an analysis of each of the fields of the Internet Protocol header, identifies their security implications, and discusses possible countermeasures for the identified threats. The latter contains an analysis of the security implications of the mechanisms implemented by the Internet Protocol.

2. The Internet Protocol

The Internet Protocol (IP) provides a basic data transfer function for passing data blocks called "datagrams" from a source host to a destination host, across the possible intervening networks. Additionally, it provides some functions that are useful for the interconnection of heterogeneous networks, such as fragmentation and reassembly.

The "datagram" has a number of characteristics that makes it convenient for interconnecting systems [Clark1988]:

- It eliminates the need of connection state within the network, which improves the survivability characteristics of the network.
- It provides a basic service of data transport that can be used as a building block for other transport services (reliable data transport services, etc.).
- It represents the minimum network service assumption, which enables IP to be run over virtually any network technology.

3. Internet Protocol Header Fields

The IETF specifications of the Internet Protocol define the syntax of the protocol header, along with the semantics of each of its fields. Figure 1 shows the format of an IP datagram, as specified in [RFC0791].

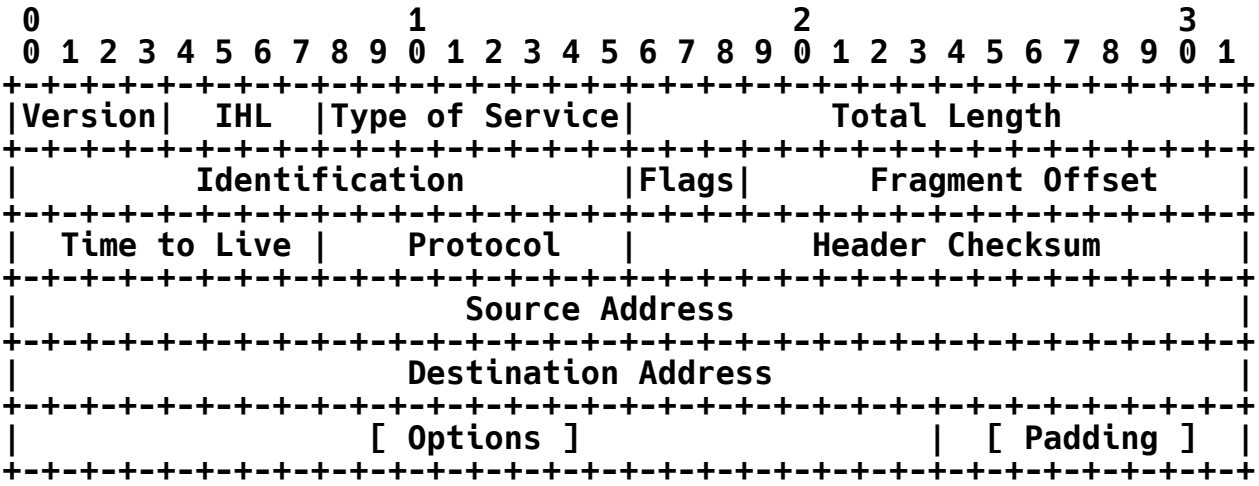

Figure 1: Internet Protocol Header Format

Even though the minimum IP header size is 20 bytes, an IP module might be handed an (illegitimate) "datagram" of less than 20 bytes. Therefore, before doing any processing of the IP header fields, the following check should be performed by the IP module on the packets handed by the link layer:

LinkLayer.PayloadSize >= 20

where LinkLayer.PayloadSize is the length (in octets) of the datagram passed from the link layer to the IP layer.

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented reflecting the packet drop).

The following subsections contain further sanity checks that should be performed on IP packets.

3.1. Version

This is a 4-bit field that indicates the version of the Internet Protocol (IP), and thus the syntax of the packet. For IPv4, this field must be 4.

When a link-layer protocol de-multiplexes a packet to an Internet module, it does so based on a Protocol Type field in the data-link packet header.

In theory, different versions of IP could coexist on a network by using the same Protocol Type at the link layer, but a different value in the Version field of the IP header. Thus, a single IP module could handle all versions of the Internet Protocol, differentiating them by means of this field.

However, in practice different versions of IP are identified by a different Protocol Type (e.g., EtherType in the case of Ethernet) number in the link-layer protocol header. For example, IPv4 datagrams are encapsulated in Ethernet frames using an EtherType of 0x0800, while IPv6 datagrams are encapsulated in Ethernet frames using an EtherType of 0x86DD [IANA_ET].

Therefore, if an IPv4 module receives a packet, the Version field must be checked to be 4. If this check fails, the packet should be silently dropped, and this event should be logged (e.g., a counter could be incremented reflecting the packet drop). If an implementation does not perform this check, an attacker could use a different value for the Version field, possibly evading NIDSs that decide which pattern-matching rules to apply based on the Version field.

If the link-layer protocol employs a specific "Protocol Type" value for encapsulating IPv4 packets (e.g., as is the case of Ethernet), a node should check that IPv4 packets are de-multiplexed to the IPv4 module when such value was used for the Protocol Type field of the link-layer protocol. If a packet does not pass this check, it should be silently dropped.

An attacker could encapsulate IPv4 packets using other link-layer "Protocol Type" values to try to subvert link-layer Access Control Lists (ACLs) and/or for tampering with NIDSs.

3.2. IHL (Internet Header Length)

The IHL (Internet Header Length) field indicates the length of the Internet header in 32-bit words (4 bytes). The following paragraphs describe a number of sanity checks to be performed on the IHL field, such that possible packet-of-death vulnerabilities are avoided.

As the minimum datagram size is 20 bytes, the minimum legal value for this field is 5. Therefore, the following check should be enforced:

IHL >= 5

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented reflecting the packet drop).

For obvious reasons, the Internet header cannot be larger than the whole Internet datagram of which it is part. Therefore, the following check should be enforced:

IHL * 4 <= Total Length

This needs to refer to the size of the datagram as specified by the sender in the Total Length field, since link layers might have added some padding (see Section 3.4).

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented reflecting the packet drop).

The above check allows for Internet datagrams with no data bytes in the payload that, while nonsensical for virtually every protocol that runs over IP, are still legal.

3.3. Type of Service (TOS)

3.3.1. Original Interpretation

Figure 2 shows the original syntax of the Type of Service field, as defined by RFC 791 [RFC0791] and updated by RFC 1349 [RFC1349]. This definition has been superseded long ago (see Sections 3.3.2.1 and 3.3.2.2), but it is still assumed by some deployed implementations.

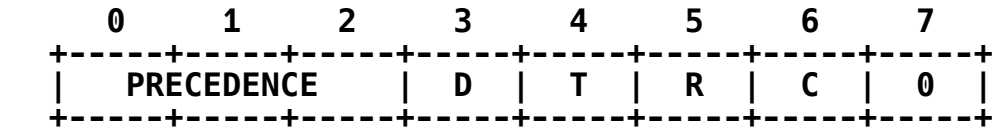

Figure 2: Type of Service Field (Original Interpretation)

Bits 0-2	Precedence
Bit 3	0 = Normal Delay, 1 = Low Delay
Bit 4	0 = Normal Throughput, 1 = High Throughput
Bit 5	0 = Normal Reliability, 1 = High Reliability
Bit 6	0 = Normal Cost, 1 = Minimize Monetary Cost
Bits 7	Reserved for Future Use (must be zero)

Table 1: Semantics of the TOS Bits

111	Network Control
110	Internetwork
101	CRITIC/ECP
100	Flash Override
011	Flash
010	Immediate
001	Priority
000	Routine

Table 2: Semantics of the Possible Precedence Field Values

The Type of Service field can be used to affect the way in which the packet is treated by the systems of a network that process it. Section 4.2.1 ("Precedence-Ordered Queue Service") and Section 4.2.2

("Weak Type of Service") of this document describe the security implications of the Type of Service field in the forwarding of packets.

3.3.2. Standard Interpretation

3.3.2.1. Differentiated Services Field

The Differentiated Services Architecture is intended to enable scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop [RFC2475]. RFC 2474 [RFC2474] redefined the IP "Type of Service" octet, introducing a Differentiated Services Field (DS Field). Figure 3 shows the format of the field.

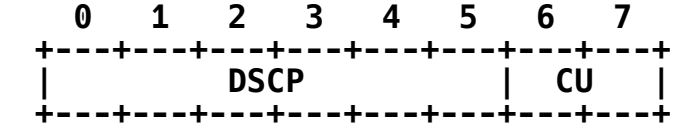

Figure 3: Revised Structure of the Type of Service Field (RFC 2474)

The DSCP ("Differentiated Services CodePoint") is used to select the treatment the packet is to receive within the Differentiated Services Domain. The CU ("Currently Unused") field was, at the time the specification was issued, reserved for future use. The DSCP field is used to select a PHB (Per-Hop Behavior), by matching against the entire 6-bit field.

Considering that the DSCP field determines how a packet is treated within a Differentiated Services (DS) domain, an attacker could send packets with a forged DSCP field to perform a theft of service or even a Denial-of-Service (DoS) attack. In particular, an attacker could forge packets with a codepoint of the type '11x000' which, according to Section 4.2.2.2 of RFC 2474 [RFC2474], would give the packets preferential forwarding treatment when compared with the PHB selected by the codepoint '000000'. If strict priority queuing were utilized, a continuous stream of such packets could cause a DoS to other flows that have a DSCP of lower relative order.

As the DS field is incompatible with the original Type of Service field, both DS domains and networks using the original Type of Service field should protect themselves by remarking the corresponding field where appropriate, probably deploying remarking boundary nodes. Nevertheless, care must be taken so that packets received with an unrecognized DSCP do not cause the handling system to malfunction.

3.3.2.2. Explicit Congestion Notification (ECN)

RFC 3168 [RFC3168] specifies a mechanism for routers to signal congestion to hosts exchanging IP packets, by marking the offending packets rather than discarding them. RFC 3168 defines the ECN field, which utilizes the CU field defined in RFC 2474 [RFC2474]. Figure 4 shows the current syntax of the IP Type of Service field, with the DSCP field used for Differentiated Services and the ECN field.

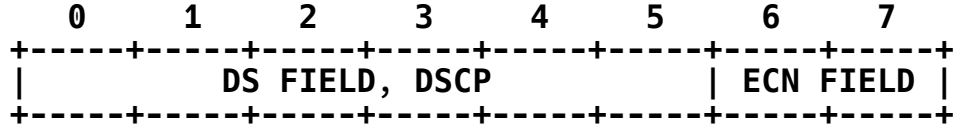

Figure 4: The Differentiated Services and ECN Fields in IP

As such, the ECN field defines four codepoints:

ECN field	Codepoint
00	Not-ECT
01	ECT(1)
10	ECT(0)
11	CE

Table 3: ECN Codepoints

ECN is an end-to-end transport protocol mechanism based on notifications by routers through which a packet flow passes. To allow this interaction to happen on the fast path of routers, the ECN field is located at a fixed location in the IP header. However, its use must be negotiated at the transport layer, and the accumulated congestion notifications must be communicated back to the sending node using transport protocol means. Thus, ECN support must be specified per transport protocol.

[RFC6040] specifies how the Explicit Congestion Notification (ECN) field of the IP header should be constructed on entry to and exit from any IP-in-IP tunnel.

The security implications of ECN are discussed in detail in a number of Sections of RFC 3168. Of the possible threats discussed in the ECN specification, we believe that one that can be easily exploited is that of a host falsely indicating ECN-Capability.

An attacker could set the ECT codepoint in the packets it sends, to signal the network that the endpoints of the transport protocol are ECN-capable. Consequently, when experiencing moderate congestion, routers using active queue management based on Random Early Detection (RED) would mark the packets (with the CE codepoint) rather than discard them. In this same scenario, packets of competing flows that do not have the ECT codepoint set would be dropped. Therefore, an attacker would get better network service than the competing flows.

However, if this moderate congestion turned into heavy congestion, routers should switch to drop packets, regardless of whether or not the packets have the ECT codepoint set.

A number of other threats could arise if an attacker was a man in the middle (i.e., was in the middle of the path the packets travel to get to the destination host). For a detailed discussion of those cases, we urge the reader to consult Section 16 of RFC 3168.

There is also ongoing work in the research community and the IETF to define alternate semantics for the CU/ECN field of IP TOS octet (see [RFC5559], [RFC5670], and [RFC5696]). The application of these methods must be confined to tightly administered domains, and on exit from such domains, all packets need to be (re-)marked with ECN semantics.

3.4. Total Length

The Total Length field is the length of the datagram, measured in bytes, including both the IP header and the IP payload. Being a 16-bit field, it allows for datagrams of up to 65535 bytes. RFC 791 [RFC0791] states that all hosts should be prepared to receive datagrams of up to 576 bytes (whether they arrive as a whole, or in fragments). However, most modern implementations can reassemble datagrams of at least 9 Kbytes.

Usually, a host will not send to a remote peer an IP datagram larger than 576 bytes, unless it is explicitly signaled that the remote peer is able to receive such "large" datagrams (for example, by means of TCP's Maximum Segment Size (MSS) option). However, systems should assume that they may receive datagrams larger than 576 bytes, regardless of whether or not they signal their remote peers to do so. In fact, it is common for Network File System (NFS) [RFC3530]

implementations to send datagrams larger than 576 bytes, even without explicit signaling that the destination system can receive such "large" datagram.

Additionally, see the discussion in Section 4.1 ("Fragment Reassembly") regarding the possible packet sizes resulting from fragment reassembly.

Implementations should be aware that the IP module could be handed a packet larger than the value actually contained in the Total Length field. Such a difference usually has to do with legitimate padding bytes at the link-layer protocol, but it could also be the result of malicious activity by an attacker. Furthermore, even when the maximum length of an IP datagram is 65535 bytes, if the link-layer technology in use allows for payloads larger than 65535 bytes, an attacker could forge such a large link-layer packet, meaning it for the IP module. If the IP module of the receiving system were not prepared to handle such an oversized link-layer payload, an unexpected failure might occur. Therefore, the memory buffer used by the IP module to store the link-layer payload should be allocated according to the payload size reported by the link layer, rather than according to the Total Length field of the IP packet it contains.

The IP module could also be handed a packet that is smaller than the actual IP packet size claimed by the Total Length field. This could be used, for example, to produce an information leakage. Therefore, the following check should be performed:

LinkLayer.PayloadSize >= Total Length

If this check fails, the IP packet should be dropped, and this event should be logged (e.g., a counter could be incremented reflecting the packet drop). As the previous expression implies, the number of bytes passed by the link layer to the IP module should contain at least as many bytes as claimed by the Total Length field of the IP header.

[US-CERT2002] is an example of the exploitation of a forged IP Total Length field to produce an information leakage attack.

3.5. Identification (ID)

The Identification field is set by the sending host to aid in the reassembly of fragmented datagrams. At any time, it needs to be unique for each set of {Source Address, Destination Address, Protocol}.

In many systems, the value used for this field is determined at the IP layer, on a protocol-independent basis. Many of those systems also simply increment the IP Identification field for each packet they send.

This implementation strategy is inappropriate for a number of reasons. Firstly, if the Identification field is set on a protocol-independent basis, it will wrap more often than necessary, and thus the implementation will be more prone to the problems discussed in [Kent1987] and [RFC4963]. Secondly, this implementation strategy opens the door to an information leakage that can be exploited in a number of ways.

[Sanfilippo1998a] describes how the Identification field can be leveraged to determine the packet rate at which a given system is transmitting information. Later, [Sanfilippo1998b] described how a system with such an implementation can 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 IP Identification field can be exploited to count the number of systems behind a NAT. [Fyodor2004] is an entire paper on most (if not all) of the ways to exploit the information provided by the Identification field of the IP header.

> Section 4.1 contains a discussion of the security implications of the IP fragment reassembly mechanism, which is the primary "consumer" of this field.

3.5.1. Some Workarounds Implemented by the Industry

As the IP Identification field is only used for the reassembly of datagrams, some operating systems (such as Linux) decided to set this field to 0 in all packets that have the DF bit set. This would, in principle, avoid any type of information leakage. However, it was detected that some non-RFC-compliant middle-boxes fragmented packets even if they had the DF bit set. In such a scenario, all datagrams originally sent with the DF bit set would all result in fragments with an Identification field of 0, which would lead to problems ("collision" of the Identification number) in the reassembly process.

Linux (and Solaris) later set the IP Identification field on a per-IP-address basis. This avoids some of the security implications of the IP Identification field, but not all. For example, systems behind a load balancer can still be counted.

3.5.2. Possible Security Improvements

Contrary to common wisdom, the IP Identification field does not need to be system-wide unique for each packet, but has to be unique for each {Source Address, Destination Address, Protocol} tuple.

> For instance, the TCP specification defines a generic send() function that takes the IP ID as one of its arguments.

We provide an analysis of the possible security improvements that could be implemented, based on whether the protocol using the services of IP is connection-oriented or connection-less.

3.5.2.1. Connection-Oriented Transport Protocols

To avoid the security implications of the information leakage described above, a pseudo-random number generator (PRNG) could be used to set the IP Identification field on a {Source Address, Destination Address} basis (for each connection-oriented transport protocol).

> [RFC4086] provides advice on the generation of pseudo-random numbers.
>
> [Klein2007] is a security advisory that describes a weakness in the pseudo-random number generator (PRNG) employed for the generation of the IP Identification by a number of operating systems.

While in theory a pseudo-random number generator could lead to scenarios in which a given Identification number is used more than once in the same time span for datagrams that end up getting fragmented (with the corresponding potential reassembly problems), in practice, this is unlikely to cause trouble.

By default, most implementations of connection-oriented protocols, such as TCP, implement some mechanism for avoiding fragmentation (such as the Path-MTU Discovery mechanism described in [RFC1191]). Thus, fragmentation will only take place if a non-RFC-compliant middle-box that still fragments packets even when the DF bit is set is placed somewhere along the path that the packets travel to get to the destination host. Once the sending system is signaled by the middle-box (by means of an ICMP "fragmentation needed and DF bit set" error message) that it should reduce the size of the packets it sends, fragmentation would be avoided. Also, for reassembly problems to arise, the same Identification value would need to be reused very frequently, and either strong packet reordering or packet loss would need to take place.

Nevertheless, regardless of what policy is used for selecting the Identification field, with the current link speeds fragmentation is already bad enough (i.e., very likely to lead to fragment reassembly errors) to rely on it. A mechanism for avoiding fragmentation (such as [RFC1191] or [RFC4821] should be implemented, instead.

3.5.2.2. Connectionless Transport Protocols

Connectionless transport protocols often have these characteristics:

- lack of flow-control mechanisms,
- lack of packet sequencing mechanisms, and/or,
- lack of reliability mechanisms (such as "timeout and retransmit").

This basically means that the scenarios and/or applications for which connection-less transport protocols are used assume that:

- Applications will be used in environments in which packet reordering is very unlikely (such as Local Area Networks), as the transport protocol itself does not provide data sequencing.
- The data transfer rates will be low enough that flow control will be unnecessary.
- Packet loss is can be tolerated and/or is unlikely.

With these assumptions in mind, the Identification field could still be set according to a pseudo-random number generator (PRNG).

> [RFC4086] provides advice on the generation of pseudo-random numbers.

In the event a given Identification number was reused while the first instance of the same number is still on the network, the first IP datagram would be reassembled before the fragments of the second IP datagram get to their destination.

In the event this was not the case, the reassembly of fragments would result in a corrupt datagram. While some existing work [Silbersack2005] assumes that this error would be caught by some upper-layer error detection code, the error detection code in question (such as UDP's checksum) might not be able to reliably detect data corruption arising from the replacement of a complete data block (as is the case in corruption arising from collision of IP Identification numbers).

In the case of UDP, unfortunately some systems have been known to not enable the UDP checksum by default. For most applications, packets containing errors should be dropped by the transport layer and not delivered to the application. A small number of applications may benefit from disabling the checksum; for example, streaming media where it is desired to avoid dropping a complete sample for a single-bit error, and UDP tunneling applications where the payload (i.e., the inner packet) is protected by its own transport checksum or other error detection mechanism.

In general, if IP Identification number collisions become an issue for the application using the connection-less protocol, the application designers should consider using a different transport protocol (which hopefully avoids fragmentation).

It must be noted that an attacker could intentionally exploit collisions of IP Identification numbers to perform a DoS attack, by sending forged fragments that would cause the reassembly process to result in a corrupt datagram that either would be dropped by the transport protocol or would incorrectly be handed to the corresponding application. This issue is discussed in detail in Section 4.1 ("Fragment Reassembly").

3.6. Flags

The IP header contains 3 control bits, two of which are currently used for the fragmentation and reassembly function.

As described by RFC 791, their meaning is:

- Bit 0: reserved, must be zero (i.e., reserved for future standardization)

- Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment

- Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments

The DF bit is usually set to implement the Path-MTU Discovery (PMTUD) mechanism described in [RFC1191]. However, it can also be exploited by an attacker to evade Network Intrusion Detection Systems. An attacker could send a packet with the DF bit set to a system monitored by a NIDS, and depending on the Path-MTU to the intended recipient, the packet might be dropped by some intervening router (because of being too big to be forwarded without fragmentation), without the NIDS being aware of it.

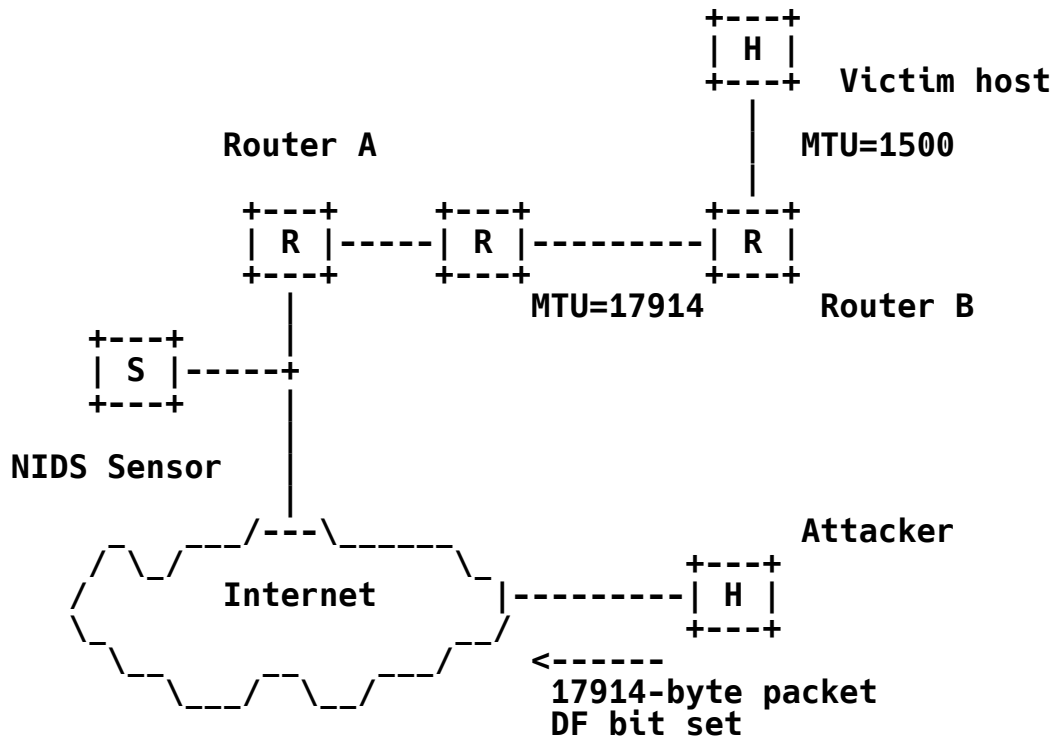

Figure 5: NIDS Evasion by Means of the Internet Protocol DF Bit

In Figure 3, an attacker sends a 17914-byte datagram meant for the victim host in the same figure. The attacker's packet probably contains an overlapping IP fragment or an overlapping TCP segment, aiming at "confusing" the NIDS, as described in [Ptacek1998]. The packet is screened by the NIDS sensor at the network perimeter, which probably reassembles IP fragments and TCP segments for the purpose of assessing the data transferred to and from the monitored systems. However, as the attacker's packet should transit a link with an MTU smaller than 17914 bytes (1500 bytes in this example), the router that encounters that this packet cannot be forwarded without fragmentation (Router B) discards the packet, and sends an ICMP "fragmentation needed and DF bit set" error message to the source host. In this scenario, the NIDS may remain unaware that the screened packet never reached the intended destination, and thus get an incorrect picture of the data being transferred to the monitored systems.

[Shankar2003] introduces a technique named "Active Mapping" that prevents evasion of a NIDS by acquiring sufficient knowledge about the network being monitored, to assess which packets will arrive at the intended recipient, and how they will be interpreted by it.

Some firewalls are known to drop packets that have both the MF (More Fragments) and the DF (Don't Fragment) bits set. While in principle such a packet might seem nonsensical, there are a number of reasons for which non-malicious packets with these two bits set can be found in a network. First, they may exist as the result of some middle-box processing a packet that was too large to be forwarded without fragmentation. Instead of simply dropping the corresponding packet and sending an ICMP error message to the source host, some middle-boxes fragment the packet (copying the DF bit to each fragment), and also send an ICMP error message to the originating system. Second, some systems (notably Linux) set both the MF and the DF bits to implement Path-MTU Discovery (PMTUD) for UDP. These scenarios should be taken into account when configuring firewalls and/or tuning NIDSs.

Section 4.1 contains a discussion of the security implications of the IP fragment reassembly mechanism.

3.7. Fragment Offset

The Fragment Offset is used for the fragmentation and reassembly of IP datagrams. It indicates where in the original datagram payload the payload of the fragment belongs, and is measured in units of eight bytes. As a consequence, all fragments (except the last one), have to be aligned on an 8-byte boundary. Therefore, if a packet has the MF flag set, the following check should be enforced:

```
(Total Length - IHL * 4) % 8 == 0
```

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented reflecting the packet drop).

Given that Fragment Offset is a 13-bit field, it can hold a value of up to 8191, which would correspond to an offset 65528 bytes within the original (non-fragmented) datagram. As such, it is possible for a fragment to implicitly claim to belong to a datagram larger than 65535 bytes (the maximum size for a legitimate IP datagram). Even when the fragmentation mechanism would seem to allow fragments that could reassemble into such large datagrams, the intent of the specification is to allow for the transmission of datagrams of up to 65535 bytes. Therefore, if a given fragment would reassemble into a datagram of more than 65535 bytes, the resulting datagram should be dropped, and this event should be logged (e.g., a counter could be incremented reflecting the packet drop). To detect such a case, the following check should be enforced on all packets for which the Fragment Offset contains a non-zero value:

Fragment Offset * 8 + (Total Length - IHL * 4) + IHL_FF * 4 <= 65535

where IHL_FF is the IHL field of the first fragment (the one with a Fragment Offset of 0).

If a fragment does not pass this check, it should be dropped.

If IHL_FF is not yet available because the first fragment has not yet arrived, for a preliminary, less rigid test, IHL_FF == IHL should be assumed, and the test is simplified to:

Fragment Offset * 8 + Total Length <= 65535

Once the first fragment is received, the full sanity check described earlier should be applied, if that fragment contains "don't copy" options.

In the worst-case scenario, an attacker could craft IP fragments such that the reassembled datagram reassembled into a datagram of 131043 bytes.

> Such a datagram would result when the first fragment has a Fragment Offset of 0 and a Total Length of 65532, and the second (and last) fragment has a Fragment Offset of 8189 (65512 bytes), and a Total Length of 65535. Assuming an IHL of 5 (i.e., a header length of 20 bytes), the reassembled datagram would be 65532 + (65535 - 20) = 131047 bytes.

Additionally, the IP module should implement all the necessary measures to be able to handle such illegitimate reassembled datagrams, so as to avoid them from overflowing the buffer(s) used for the reassembly function.

> [CERT1996c] and [Kenney1996] describe the exploitation of this issue to perform a DoS attack.

Section 4.1 contains a discussion of the security implications of the IP fragment reassembly mechanism.

3.8. Time to Live (TTL)

The Time to Live (TTL) field has two functions: to bound the lifetime of the upper-layer packets (e.g., TCP segments) and to prevent packets from looping indefinitely in the network.

Originally, this field was meant to indicate the maximum time a datagram was allowed to remain in the Internet system, in units of seconds. As every Internet module that processes a datagram must

decrement the TTL by at least one, the original definition of the TTL field became obsolete, and in practice it is interpreted as a hop count (see Section 5.3.1 of [RFC1812]).

Most systems allow the administrator to configure the TTL to be used for the packets they originate, with the default value usually being a power of 2, or 255 (e.g., see [Arkin2000]). The recommended value for the TTL field, as specified by the IANA is 64 [IANA_IP_PARAM]. This value reflects the assumed "diameter" of the Internet, plus a margin to accommodate its growth.

The TTL field has a number of properties that are interesting from a security point of view. Given that the default value used for the TTL is usually either a power of two, or 255, chances are that unless the originating system has been explicitly tuned to use a non-default value, if a packet arrives with a TTL of 60, the packet was originally sent with a TTL of 64. In the same way, if a packet is received with a TTL of 120, chances are that the original packet had a TTL of 128.

> This discussion assumes there was no protocol scrubber, transparent proxy, or some other middle-box that overwrites the TTL field in a non-standard way, between the originating system and the point of the network in which the packet was received.

Determining the TTL with which a packet was originally sent by the source system can help to obtain valuable information. Among other things, it may help in:

- Fingerprinting the originating operating system.
- Fingerprinting the originating physical device.
- Mapping the network topology.
- Locating the source host in the network topology.
- Evading Network Intrusion Detection Systems.

However, it can also be used to perform important functions such as:

- Improving the security of applications that make use of the Internet Protocol (IP).
- Limiting spread of packets.

3.8.1. Fingerprinting the Operating System in Use by the Source Host

Different operating systems use a different default TTL for the packets they send. Thus, asserting the TTL with which a packet was originally sent will help heuristics to reduce the number of possible operating systems in use by the source host. It should be noted that since most systems use only a handful of different default values, the granularity of OS fingerprinting that this technique provides is negligible. Additionally, these defaults may be configurable (system-wide or per protocol), and managed systems may employ such opportunities for operational purposes and to defeat the capability of fingerprinting heuristics.

3.8.2. Fingerprinting the Physical Device from which the Packets Originate

When several systems are behind a middle-box such as a NAT or a load balancer, the TTL may help to count the number of systems behind the middle-box. If each of the systems behind the middle-box uses a different default TTL value for the packets it sends, or each system is located at different distances in the network topology, an attacker could stimulate responses from the devices being fingerprinted, and responses that arrive with different TTL values could be assumed to come from a different devices.

> Of course, there are many other (and much more precise) techniques to fingerprint physical devices. One weakness of this method is that, while many systems differ in the default TTL value that they use, there are also many implementations which use the same default TTL value. Additionally, packets sent by a given device may take different routes (e.g., due to load sharing or route changes), and thus a given packet may incorrectly be presumed to come from a different device, when in fact it just traveled a different route.

However, these defaults may be configurable (system-wide or per protocol) and managed systems may employ such opportunities for operational purposes and to defeat the capability of fingerprinting heuristics.

3.8.3. Mapping the Network Topology

An originating host may set the TTL field of the packets it sends to progressively increasing values in order to elicit an ICMP error message from the routers that decrement the TTL of each packet to zero, and thereby determine the IP addresses of the routers on the path to the packet's destination. This procedure has been traditionally employed by the traceroute tool.

3.8.4. Locating the Source Host in the Network Topology

The TTL field may also be used to locate the source system in the network topology [Northcutt2000].

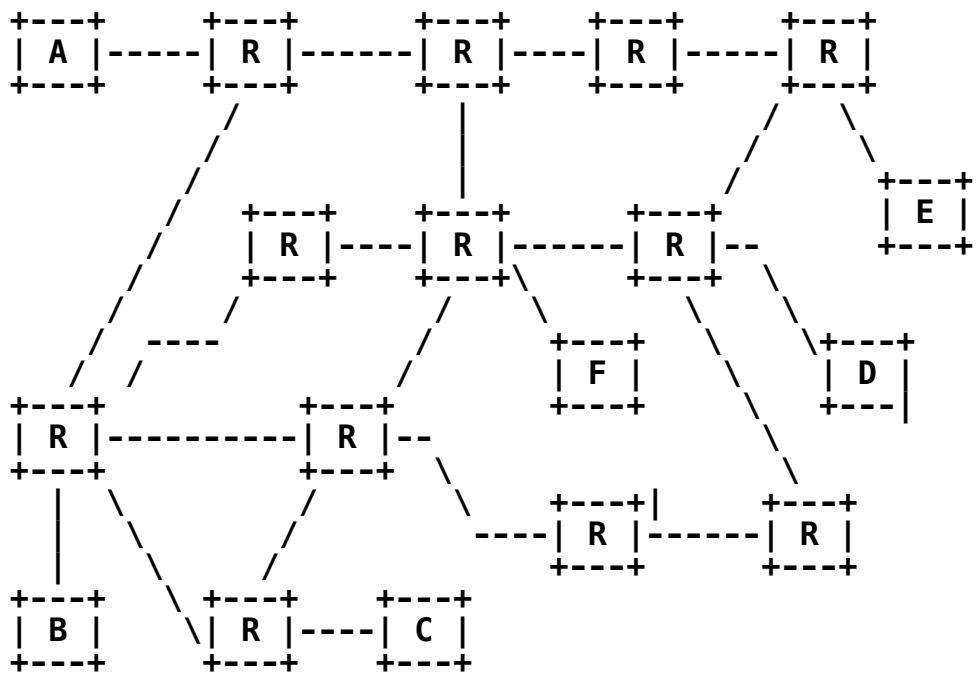

Figure 6: Tracking a Host by Means of the TTL Field

Consider network topology of Figure 6. Assuming that an attacker ("F" in the figure) is performing some type of attack that requires forging the Source Address (such as for a TCP-based DoS reflection attack), and some of the involved hosts are willing to cooperate to locate the attacking system.

Assuming that:

- All the packets A gets have a TTL of 61.
- All the packets B gets have a TTL of 61.
- All the packets C gets have a TTL of 61.
- All the packets D gets have a TTL of 62.

Based on this information, and assuming that the system's default value was not overridden, it would be fair to assume that the original TTL of the packets was 64. With this information, the number of hops between the attacker and each of the aforementioned hosts can be calculated.

The attacker is:

- Four hops away from A.
- Four hops away from B.
- Four hops away from C.
- Four hops away from D.

In the network setup of Figure 3, the only system that satisfies all these conditions is the one marked as the "F".

The scenario described above is for illustration purposes only. In practice, there are a number of factors that may prevent this technique from being successfully applied:

- Unless there is a "large" number of cooperating systems, and the attacker is assumed to be no more than a few hops away from these systems, the number of "candidate" hosts will usually be too large for the information to be useful.
- The attacker may be using a non-default TTL value, or, what is worse, using a pseudo-random value for the TTL of the packets it sends.
- The packets sent by the attacker may take different routes, as a result of a change in network topology, load sharing, etc., and thus may lead to an incorrect analysis.

3.8.5. Evading Network Intrusion Detection Systems

The TTL field can be used to evade Network Intrusion Detection Systems. Depending on the position of a sensor relative to the destination host of the examined packet, the NIDS may get a different picture from that of the intended destination system. As an example, a sensor may process a packet that will expire before getting to the destination host. A general countermeasure for this type of attack is to normalize the traffic that gets to an organizational network. Examples of such traffic normalization can be found in [Paxson2001]. OpenBSD Packet Filter is an example of a packet filter that includes TTL-normalization functionality [OpenBSD-PF]

3.8.6. Improving the Security of Applications That Make Use of the Internet Protocol (IP)

In some scenarios, the TTL field can be also used to improve the security of an application, by restricting the hosts that can communicate with the given application [RFC5082]. For example, there are applications for which the communicating systems are typically in the same network segment (i.e., there are no intervening routers). Such an application is the BGP (Border Gateway Protocol) utilized by two peer routers (usually on a shared link medium).

If both systems use a TTL of 255 for all the packets they send to each other, then a check could be enforced to require all packets meant for the application in question to have a TTL of 255.

As all packets sent by systems that are not in the same network segment will have a TTL smaller than 255, those packets will not pass the check enforced by these two cooperating peers. This check reduces the set of systems that may perform attacks against the protected application (BGP in this case), thus mitigating the attack vectors described in [NISCC2004] and [Watson2004].

> This same check is enforced for related ICMP error messages, with the intent of mitigating the attack vectors described in [NISCC2005] and [RFC5927].

The TTL field can be used in a similar way in scenarios in which the cooperating systems are not in the same network segment (i.e., multi-hop peering). In that case, the following check could be enforced:

TTL >= 255 - DeltaHops

This means that the set of hosts from which packets will be accepted for the protected application will be reduced to those that are no more than DeltaHops away. While for obvious reasons the level of protection will be smaller than in the case of directly connected peers, the use of the TTL field for protecting multi-hop peering still reduces the set of hosts that could potentially perform a number of attacks against the protected application.

This use of the TTL field has been officially documented by the IETF under the name "Generalized TTL Security Mechanism" (GTSM) in [RFC5082].

Some protocol scrubbers enforce a minimum value for the TTL field of the packets they forward. It must be understood that depending on the minimum TTL being enforced, and depending on the particular network setup, the protocol scrubber may actually help attackers to fool the GTSM, by "raising" the TTL of the attacking packets.

3.8.7. Limiting Spread

The originating host sets the TTL field to a small value (frequently 1, for link-scope services) in order to artificially limit the (topological) distance the packet is allowed to travel. This is suggested in Section 4.2.2.9 of RFC 1812 [RFC1812]. Further discussion of this technique can be found in RFC 1112 [RFC1112].

3.9. Protocol

The Protocol field indicates the protocol encapsulated in the Internet datagram. The Protocol field may not only contain a value corresponding to a protocol implemented by the system processing the packet, but also a value corresponding to a protocol not implemented, or even a value not yet assigned by the IANA [IANA_PROT_NUM].

While in theory there should not be security implications from the use of any value in the protocol field, there have been security issues in the past with systems that had problems when handling packets with some specific protocol numbers [Cisco2003] [CERT2003].

A host (i.e., end-system) that receives an IP packet encapsulating a Protocol it does not support should drop the corresponding packet, log the event, and possibly send an ICMP Protocol Unreachable (type 3, code 2) error message.

3.10. Header Checksum

The Header Checksum field is an error-detection mechanism meant to detect errors in the IP header. While in principle there should not be security implications arising from this field, it should be noted that due to non-RFC-compliant implementations, the Header Checksum might be exploited to detect firewalls and/or evade NIDSs.

[Ed3f2002] describes the exploitation of the TCP checksum for performing such actions. As there are Internet routers known to not check the IP Header Checksum, and there might also be middle-boxes (NATs, firewalls, etc.) not checking the IP checksum allegedly due to performance reasons, similar malicious activity to the one described in [Ed3f2002] might be performed with the IP checksum.

3.11. Source Address

The Source Address of an IP datagram identifies the node from which the packet originated.

> Strictly speaking, the Source Address of an IP datagram identifies the interface of the sending system from which the packet was sent, (rather than the originating "system"), as in the Internet Architecture there's no concept of "node address".

Unfortunately, it is trivial to forge the Source Address of an Internet datagram because of the apparent lack of consistent "egress filtering" near the edge of the network. This has been exploited in the past for performing a variety of DoS attacks [NISCC2004] [RFC4987] [CERT1996a] [CERT1996b] [CERT1998a] and for impersonating other systems in scenarios in which authentication was based on the Source Address of the sending system [daemon91996].

The extent to which these attacks can be successfully performed in the Internet can be reduced through deployment of ingress/egress filtering in the Internet routers. [NISCC2006] is a detailed guide on ingress and egress filtering. [RFC2827] and [RFC3704] discuss ingress filtering. [GIAC2000] discusses egress filtering. [SpooferProject] measures the Internet's susceptibility to forged Source Address IP packets.

> Even when the obvious field on which to perform checks for ingress/egress filtering is the Source Address and Destination Address fields of the IP header, there are other occurrences of IP addresses on which the same type of checks should be performed. One example is the IP addresses contained in the payload of ICMP error messages, as discussed in [RFC5927] and [Gont2006].

There are a number of sanity checks that should be performed on the Source Address of an IP datagram. Details can be found in Section 4.3 ("Addressing").

Additionally, there exist freely available tools that allow administrators to monitor which IP addresses are used with which MAC addresses [LBNL2006]. This functionality is also included in many NIDSs.

It is also very important to understand that authentication should never rely solely on the Source Address used by the communicating systems.

3.12. Destination Address

The Destination Address of an IP datagram identifies the destination host to which the packet is meant to be delivered.

> Strictly speaking, the Destination Address of an IP datagram identifies the interface of the destination network interface, rather than the destination "system", as in the Internet Architecture there's no concept of "node address".

There are a number of sanity checks that should be performed on the Destination Address of an IP datagram. Details can be found in Section 4.3 ("Addressing").

3.13. Options

According to RFC 791, IP options must be implemented by all IP modules, both in hosts and gateways (i.e., end-systems and intermediate-systems). This means that the general rules for assembling, parsing, and processing of IP options must be implemented. RFC 791 defines a set of options that "must be understood", but this set has been updated by RFC 1122 [RFC1122], RFC 1812 [RFC1812], and other documents. Section 3.13.2 of this document describes for each option type the current understanding of the implementation requirements. IP systems are required to ignore options they do not implement.

> It should be noted that while a number of IP options have been specified, they are generally only used for troubleshooting purposes (except for the Router Alert option and the different Security options).

There are two cases for the format of an option:

- Case 1: A single byte of option-type.
- Case 2: An option-type byte, an option-length byte, and the actual option-data bytes.

In Case 2, the option-length byte counts the option-type byte and the option-length byte, as well as the actual option-data bytes.

All current and future options except End of Option List (Type = 0) and No Operation (Type = 1), are of Class 2.

The option-type has three fields:

- 1 bit: copied flag.

- 2 bits: option class.
- 5 bits: option number.

This format allows for the creation of new options for the extension of the Internet Protocol (IP) and their transparent treatment on intermediate-systems that do not "understand" them, under direction of the first three functional parts.

The copied flag indicates whether this option should be copied to all fragments in the event the packet carrying it needs to be fragmented:

- 0 = not copied.
- 1 = copied.

The values for the option class are:

- 0 = control.
- 1 = reserved for future use.
- 2 = debugging and measurement.
- 3 = reserved for future use.

Finally, the option number identifies the syntax of the rest of the option.

[IANA_IP_PARAM] contains the list of the currently assigned IP option numbers. It should be noted that IP systems are required to ignore those options they do not implement.

3.13.1. General Issues with IP Options

The following subsections discuss security issues that apply to all IP options. The proposed checks should be performed in addition to any option-specific checks proposed in the next sections.

3.13.1.1. Processing Requirements

Router manufacturers tend to do IP option processing on the main processor, rather than on line cards. Unless special care is taken, this represents DoS risk, as there is potential for overwhelming the router with option processing.

To reduce the impact of these packets on the system performance, a few countermeasures could be implemented:

- Rate-limit the number of packets with IP options that are processed by the system.

- Enforce a limit on the maximum number of options to be accepted on a given Internet datagram.

The first check avoids a flow of packets with IP options to overwhelm the system in question. The second check avoids packets with many IP options to affect the performance of the system.

3.13.1.2. Processing of the Options by the Upper-Layer Protocol

Section 3.2.1.8 of RFC 1122 [RFC1122] states that all the IP options received in IP datagrams must be passed to the transport layer (or to ICMP processing when the datagram is an ICMP message). Therefore, care in option processing must be taken not only at the Internet layer but also in every protocol module that may end up processing the options included in an IP datagram.

3.13.1.3. General Sanity Checks on IP Options

There are a number of sanity checks that should be performed on IP options before further option processing is done. They help prevent a number of potential security problems, including buffer overflows. When these checks fail, the packet carrying the option should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

RFC 1122 [RFC1122] recommends to send an ICMP "Parameter Problem" message to the originating system when a packet is dropped because of an invalid value in a field, such as the cases discussed in the following subsections. Sending such a message might help in debugging some network problems. However, it would also alert attackers about the system that is dropping packets because of the invalid values in the protocol fields.

We advice that systems default to sending an ICMP "Parameter Problem" error message when a packet is dropped because of an invalid value in a protocol field (e.g., as a result of dropping a packet due to the sanity checks described in this section). However, we recommend that systems provide a system-wide toggle that allows an administrator to override the default behavior so that packets can be silently dropped when an invalid value in a protocol field is encountered.

Option length

Section 3.2.1.8 of RFC 1122 explicitly states that the IP layer must not crash as the result of an option length that is outside the possible range, and mentions that erroneous option lengths have been observed to put some IP implementations into infinite loops.

For options that belong to the "Case 2" described in the previous section, the following check should be performed:

option-length >= 2

The value "2" accounts for the option-type byte and the option-length byte.

This check prevents, among other things, loops in option processing that may arise from incorrect option lengths.

Additionally, while the option-length byte of IP options of "Case 2" allows for an option length of up to 255 bytes, there is a limit on legitimate option length imposed by the space available for options in the IP header.

For all options of "Case 2", the following check should be enforced:

option-offset + option-length <= IHL * 4

Where option-offset is the offset of the first byte of the option within the IP header, with the first byte of the IP header being assigned an offset of 0.

This check assures that the option does not claim to extend beyond the IP header. If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

The aforementioned check is meant to detect forged option-length values that might make an option overlap with the IP payload. This would be particularly dangerous for those IP options that request the processing systems to write information into the option-data area (such as the Record Route option), as it would allow the generation of overflows.

Data types

Many IP options use pointer and length fields. Care must be taken as to the data type used for these fields in the implementation. For example, if an 8-bit signed data type were used to hold an 8-bit pointer, then, pointer values larger than 128 might mistakenly be interpreted as negative numbers, and thus might lead to unpredictable results.

3.13.2. Issues with Specific Options

3.13.2.1. End of Option List (Type=0)

This option is used to indicate the "end of options" in those cases in which the end of options would not coincide with the end of the Internet Protocol header. Octets in the IP header following the "End of Option List" are to be regarded as padding (they should set to 0 by the originator and must to be ignored by receiving nodes).

However, an originating node could alternatively fill the remaining space in the Internet header with No Operation options (see Section 3.13.2.2). The End of Option List option allows slightly more efficient parsing on receiving nodes and should be preferred by packet originators. All IP systems are required to understand both encodings.

3.13.2.2. No Operation (Type=1)

The No Operation option is basically meant to allow the sending system to align subsequent options in, for example, 32-bit boundaries, but it can also be used at the end of the options (see Section 3.13.2.1).

With a single exception (see Section 3.13.2.13), this option is the only IP option defined so far that can occur in multiple instances in a single IP packet.

This option does not have security implications.

3.13.2.3. Loose Source and Record Route (LSRR) (Type=131)

This option lets the originating system specify a number of intermediate-systems a packet must pass through to get to the destination host. Additionally, the route followed by the packet is recorded in the option. The receiving host (end-system) must use the reverse of the path contained in the received LSRR option.

The LSSR option can be of help in debugging some network problems. Some ISP (Internet Service Provider) peering agreements require support for this option in the routers within the peer of the ISP.

The LSRR option has well-known security implications. Among other things, the option can be used to:

- Bypass firewall rules
- Reach otherwise unreachable Internet systems
- Establish TCP connections in a stealthy way
- Learn about the topology of a network
- Perform bandwidth-exhaustion attacks

Of these attack vectors, the one that has probably received the least attention is the use of the LSRR option to perform bandwidth exhaustion attacks. The LSRR option can be used as an amplification method for performing bandwidth-exhaustion attacks, as an attacker could make a packet bounce multiple times between a number of systems by carefully crafting an LSRR option.

> This is the IPv4-version of the IPv6 amplification attack that was widely publicized in 2007 [Biondi2007]. The only difference is that the maximum length of the IPv4 header (and hence the LSRR option) limits the amplification factor when compared to the IPv6 counterpart.

While the LSSR option may be of help in debugging some network problems, its security implications outweigh any legitimate use.

All systems should, by default, drop IP packets that contain an LSRR option, and should log this event (e.g., a counter could be incremented to reflect the packet drop). However, they should provide a system-wide toggle to enable support for this option for those scenarios in which this option is required. Such system-wide toggle should default to "off" (or "disable").

> [OpenBSD1998] is a security advisory about an improper implementation of such a system-wide toggle in 4.4BSD kernels.

Section 3.3.5 of RFC 1122 [RFC1122] states that a host may be able to act as an intermediate hop in a source route, forwarding a source-routed datagram to the next specified hop. We strongly discourage host software from forwarding source-routed datagrams.

If processing of source-routed datagrams is explicitly enabled in a system, the following sanity checks should be performed.

RFC 791 states that this option should appear, at most, once in a given packet. Thus, if a packet contains more than one LSRR option, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop). Additionally, packets containing a combination of LSRR and SSRR options should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

As all other IP options of "Case 2", the LSSR contains a Length field that indicates the length of the option. Given the format of the option, the Length field should be checked to have a minimum value of three and be 3 (3 + n*4):

```
LSRR.Length % 4 == 3 && LSRR.Length != 0
```

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

The Pointer is relative to this option. Thus, the minimum legal value is 4. Therefore, the following check should be performed.

```
LSRR.Pointer >= 4
```

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop). Additionally, the Pointer field should be a multiple of 4. Consequently, the following check should be performed:

```
LSRR.Pointer % 4 == 0
```

If a packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

When a system receives an IP packet with the LSRR option passing the above checks, it should check whether or not the source route is empty. The option is empty if:

```
LSRR.Pointer > LSRR.Length
```

In that case, routing should be based on the Destination Address field, and no further processing should be done on the LSRR option.

[Microsoft1999] is a security advisory about a vulnerability arising from improper validation of the LSRR.Pointer field.

If the address in the Destination Address field has been reached, and the option is not empty, the next address in the source route replaces the address in the Destination Address field, and the IP address of the interface that will be used to forward this datagram is recorded in its place in the LSRR.Data field. Then, the LSRR.Pointer. is incremented by 4.

Note that the sanity checks for the LSRR.Length and the LSRR.Pointer fields described above ensure that if the option is not empty, there will be (4*n) octets in the option. That is, there will be at least one IP address to read and enough room to record the IP address of the interface that will be used to forward this datagram.

The LSRR must be copied on fragmentation. This means that if a packet that carries the LSRR is fragmented, each of the fragments will have to go through the list of systems specified in the LSRR option.

3.13.2.4. Strict Source and Record Route (SSRR) (Type=137)

This option allows the originating system to specify a number of intermediate-systems a packet must pass through to get to the destination host. Additionally, the route followed by the packet is recorded in the option, and the destination host (end-system) must use the reverse of the path contained in the received SSRR option.

This option is similar to the Loose Source and Record Route (LSRR) option, with the only difference that in the case of SSRR, the route specified in the option is the exact route the packet must take (i.e., no other intervening routers are allowed to be in the route).

The SSSR option can be of help in debugging some network problems. Some ISP (Internet Service Provider) peering agreements require support for this option in the routers within the peer of the ISP.

The SSRR option has the same security implications as the LSRR option. Please refer to Section 3.13.2.3 for a discussion of such security implications.

As with the LSRR, while the SSSR option may be of help in debugging some network problems, its security implications outweigh any legitimate use of it.

All systems should, by default, drop IP packets that contain an SSRR option, and should log this event (e.g., a counter could be incremented to reflect the packet drop). However, they should provide a system-wide toggle to enable support for this option for those scenarios in which this option is required. Such system-wide toggle should default to "off" (or "disable").

> [OpenBSD1998] is a security advisory about an improper implementation of such a system-wide toggle in 4.4BSD kernels.

In the event processing of the SSRR option were explicitly enabled, the same sanity checks described for the LSRR option in Section 3.13.2.3 should be performed on the SSRR option. Namely, sanity checks should be performed on the option length (SSRR.Length) and the pointer field (SSRR.Pointer).

If the packet passes the aforementioned sanity checks, the receiving system should determine whether the Destination Address of the packet corresponds to one of its IP addresses. If does not, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

> Contrary to the IP Loose Source and Record Route (LSRR) option, the SSRR option does not allow in the route other routers than those contained in the option. If the system implements the weak end-system model, it is allowed for the system to receive a packet destined to any of its IP addresses, on any of its interfaces. If the system implements the strong end-system model, a packet destined to it can be received only on the interface that corresponds to the IP address contained in the Destination Address field [RFC1122].

If the packet passes this check, the receiving system should determine whether the source route is empty or not. The option is empty if:

SSRR.Pointer > SSRR.Length

In that case, if the address in the destination field has not been reached, the packet should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

> [Microsoft1999] is a security advisory about a vulnerability arising from improper validation of the SSRR.Pointer field.

If the option is not empty, and the address in the Destination Address field has been reached, the next address in the source route replaces the address in the Destination Address field, and the IP address of the interface that will be used to forward this datagram is recorded in its place in the source route (SSRR.Data field). Then, the SSRR.Pointer is incremented by 4.

> Note that the sanity checks for the SSRR.Length and the SSRR.Pointer fields described above ensure that if the option is not empty, there will be (4*n) octets in the option. That is, there will be at least one IP address to read, and enough room to record the IP address of the interface that will be used to forward this datagram.

The SSRR option must be copied on fragmentation. This means that if a packet that carries the SSRR is fragmented, each of the fragments will have to go through the list of systems specified in the SSRR option.

3.13.2.5. Record Route (Type=7)

This option provides a means to record the route that a given packet follows.

The option begins with an 8-bit option code, which is equal to 7. The second byte is the option length, which includes the option-type byte, the option-length byte, the pointer byte, and the actual option-data. The third byte is a pointer into the route data, indicating the first byte of the area in which to store the next route data. The pointer is relative to the option start.

RFC 791 states that this option should appear, at most, once in a given packet. Therefore, if a packet has more than one instance of this option, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

The same sanity checks performed for the Length field and the Pointer field of the LSRR and the SSRR options should be performed on the Length field (RR.Length) and the Pointer field (RR.Pointer) of the RR option. As with the LSRR and SSRR options, if the packet does not pass these checks it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

When a system receives an IP packet with the Record Route option that passes the above checks, it should check whether there is space in the option to store route information. The option is full if:

RR.Pointer > RR.Length

If the option is full, the datagram should be forwarded without further processing of this option.

If the option is not full (i.e., RR.Pointer <= RR.Length), the IP address of the interface that will be used to forward this datagram should be recorded into the area pointed to by the RR.Pointer, and RR.Pointer should then incremented by 4.

This option is not copied on fragmentation, and thus appears in the first fragment only. If a fragment other than the one with offset 0 contains the Record Route option, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

The Record Route option can be exploited to learn about the topology of a network. However, the limited space in the IP header limits the usefulness of this option for that purpose if the target network is several hops away.

3.13.2.6. Stream Identifier (Type=136)

The Stream Identifier option originally provided a means for the 16-bit SATNET stream Identifier to be carried through networks that did not support the stream concept.

However, as stated by Section 4.2.2.1 of RFC 1812 [RFC1812], this option is obsolete. Therefore, it must be ignored by the processing systems.

In the case of legacy systems still using this option, the length field of the option should be checked to be 4. If the option does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

RFC 791 states that this option appears at most once in a given datagram. Therefore, if a packet contains more than one instance of this option, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

3.13.2.7. Internet Timestamp (Type=68)

This option provides a means for recording the time at which each system processed this datagram. The timestamp option has a number of security implications. Among them are the following:

- It allows an attacker to obtain the current time of the systems that process the packet, which the attacker may find useful in a number of scenarios.

- It may be used to map the network topology, in a similar way to the IP Record Route option.

- It may be used to fingerprint the operating system in use by a system processing the datagram.

- It may be used to fingerprint physical devices by analyzing the clock skew.

Therefore, by default, the timestamp option should be ignored.

For those systems that have been explicitly configured to honor this option, the rest of this subsection describes some sanity checks that should be enforced on the option before further processing.

The option begins with an option-type byte, which must be equal to 68. The second byte is the option-length, which includes the option-type byte, the option-length byte, the pointer, and the overflow/flag byte. The minimum legal value for the option-length byte is 4, which corresponds to an Internet Timestamp option that is empty (no space to store timestamps). Therefore, upon receipt of a packet that contains an Internet Timestamp option, the following check should be performed:

```
IT.Length >= 4
```

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

The Pointer is an index within this option, counting the option type octet as octet #1. It points to the first byte of the area in which the next timestamp data should be stored and thus, the minimum legal value is 5. Since the only change of the Pointer allowed by RFC 791 is incrementing it by 4 or 8, the following checks should be performed on the Internet Timestamp option, depending on the Flag value (see below).

If IT.Flag is equal to 0, the following check should be performed:

```
IT.Pointer %4 == 1 && IT.Pointer != 1
```

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

Otherwise, the following sanity check should be performed on the option:

IT.Pointer % 8 == 5

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

The flag field has three possible legal values:

- 0: Record time stamps only, stored in consecutive 32-bit words.
- 1: Record each timestamp preceded with the Internet address of the registering entity.
- 3: The internet address fields of the option are pre-specified. An IP module only registers its timestamp if it matches its own address with the next specified Internet address.

Therefore the following check should be performed:

IT.Flag == 0 || IT.Flag == 1 || IT.Flag == 3

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

The timestamp field is a right-justified 32-bit timestamp in milliseconds since UTC. If the time is not available in milliseconds, or cannot be provided with respect to UTC, then any time may be inserted as a timestamp, provided the high-order bit of the timestamp is set, to indicate this non-standard value.

According to RFC 791, the initial contents of the timestamp area must be initialized to zero, or Internet address/zero pairs. However, Internet systems should be able to handle non-zero values, possibly discarding the offending datagram.

When an Internet system receives a packet with an Internet Timestamp option, it decides whether it should record its timestamp in the option. If it determines that it should, it should then determine whether the timestamp data area is full, by means of the following check:

IT.Pointer > IT.Length

If this condition is true, the timestamp data area is full. If not, there is room in the timestamp data area.

If the timestamp data area is full, the overflow byte should be incremented, and the packet should be forwarded without inserting the timestamp. If the overflow byte itself overflows, the packet should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

If the timestamp data area is not full, then processing continues as follows (note that the above checks on IT.Pointer ensure that there is room for another entry in the option):

- If IT.Flag is 0, then the system's 32-bit timestamp is stored into the area pointed to by the pointer byte and the pointer byte is incremented by 4.

- If IT.Flag is 1, then the IP address of the system is stored into the area pointed to by the pointer byte, followed by the 32-bit system timestamp, and the pointer byte is incremented by 8.

- Otherwise (IT.Flag is 3), if the IP address in the first 4 bytes pointed to by IT.Pointer matches one of the IP addresses assigned to an interface of the system, then the system's timestamp is stored into the area pointed to by IT.Pointer + 4, and the pointer byte is incremented by 8.

[Kohno2005] describes a technique for fingerprinting devices by measuring the clock skew. It exploits, among other things, the timestamps that can be obtained by means of the ICMP timestamp request messages [RFC0791]. However, the same fingerprinting method could be implemented with the aid of the Internet Timestamp option.

3.13.2.8. Router Alert (Type=148)

The Router Alert option is defined in RFC 2113 [RFC2113] and later updates to it have been clarified by RFC 5350 [RFC5350]. It contains a 16-bit Value governed by an IANA registry (see [RFC5350]). The Router Alert option has the semantic "routers should examine this packet more closely, if they participate in the functionality denoted by the Value of the option".

According to the syntax of the option as defined in RFC 2113, the following check should be enforced, if the router supports this option:

RA.Length == 4

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

A packet that contains a Router Alert option with an option value corresponding to functionality supported by an active module in the router will not go through the router's fast-path but will be processed in the slow path of the router, handing it over for closer inspection to the modules that has registered the matching option value. Therefore, this option may impact the performance of the systems that handle the packet carrying it.

[ROUTER-ALERT] analyzes the security implications of the Router Alert option, and identifies controlled environments in which the Router Alert option can be used safely.

As explained in RFC 2113 [RFC2113], hosts should ignore this option.

3.13.2.9. Probe MTU (Type=11) (Obsolete)

This option was defined in RFC 1063 [RFC1063] and originally provided a mechanism to discover the Path-MTU.

This option is obsolete, and therefore any packet that is received containing this option should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

3.13.2.10. Reply MTU (Type=12) (Obsolete)

This option is defined in RFC 1063 [RFC1063], and originally provided a mechanism to discover the Path-MTU.

This option is obsolete, and therefore any packet that is received containing this option should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

3.13.2.11. Traceroute (Type=82)

This option is defined in RFC 1393 [RFC1393], and originally provided a mechanism to trace the path to a host.

The Traceroute option was specified as "experimental", and it was never deployed on the public Internet. Therefore, any packet that is received containing this option should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

3.13.2.12. Department of Defense (DoD) Basic Security Option (Type=130)

This option is used by Multi-Level-Secure (MLS) end-systems and intermediate-systems in specific environments to [RFC1108]:

- Transmit from source to destination in a network standard representation the common security labels required by computer security models,
- Validate the datagram as appropriate for transmission from the source and delivery to the destination, and
- Ensure that the route taken by the datagram is protected to the level required by all protection authorities indicated on the datagram.

It is specified by RFC 1108 [RFC1108] (which obsoletes RFC 1038 [RFC1038]).

> RFC 791 [RFC0791] defined the "Security Option" (Type=130), which used the same option type as the DoD Basic Security option discussed in this section. The "Security Option" specified in RFC 791 is considered obsolete by Section 3.2.1.8 of RFC 1122, and therefore the discussion in this section is focused on the DoD Basic Security option specified by RFC 1108 [RFC1108].

Section 4.2.2.1 of RFC 1812 states that routers "SHOULD implement this option".

The DoD Basic Security option is currently implemented in a number of operating systems (e.g., [IRIX2008], [SELinux2009], [Solaris2007], and [Cisco2008]), and deployed in a number of high-security networks.

Systems that belong to networks in which this option is in use should process the DoD Basic Security option contained in each packet as specified in [RFC1108].

RFC 1108 states that the option should appear at most once in a datagram. Therefore, if more than one DoD Basic Security option (BSO) appears in a given datagram, the corresponding datagram should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

RFC 1108 states that the option Length is variable, with a minimum option Length of 3 bytes. Therefore, the following check should be performed:

BSO.Length >= 3

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

Current deployments of the security options described in this section and the two subsequent sections have motivated the specification of a "Common Architecture Label IPv6 Security Option (CALIPSO)" for the IPv6 protocol [RFC5570].

3.13.2.13. DoD Extended Security Option (Type=133)

This option permits additional security labeling information, beyond that present in the Basic Security option (Section 3.13.2.13), to be supplied in an IP datagram to meet the needs of registered authorities. It is specified by RFC 1108 [RFC1108].

This option may be present only in conjunction with the DoD Basic Security option. Therefore, if a packet contains a DoD Extended Security option (ESO), but does not contain a DoD Basic Security option, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop). It should be noted that, unlike the DoD Basic Security option, this option may appear multiple times in a single IP header.

Systems that belong to networks in which this option is in use, should process the DoD Extended Security option contained in each packet as specified in RFC 1108 [RFC1108].

RFC 1108 states that the option Length is variable, with a minimum option Length of 3 bytes. Therefore, the following check should be performed:

ESO.Length >= 3

If the packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

3.13.2.14. Commercial IP Security Option (CIPSO) (Type=134)

This option was proposed by the Trusted Systems Interoperability Group (TSIG), with the intent of meeting trusted networking requirements for the commercial trusted systems market place. It is specified in [CIPSO1992] and [FIPS1994].

> The TSIG proposal was taken to the Commercial Internet Security Option (CIPSO) Working Group of the IETF [CIPSOWG1994], and an Internet-Draft was produced [CIPSO1992]. The Internet-Draft was never published as an RFC, but the proposal was later standardized by the U.S. National Institute of Standards and Technology (NIST) as "Federal Information Processing Standard Publication 188" [FIPS1994].

It is currently implemented in a number of operating systems (e.g., IRIX [IRIX2008], Security-Enhanced Linux [SELinux2009], and Solaris [Solaris2007]), and deployed in a number of high-security networks.

> [Zakrzewski2002] and [Haddad2004] provide an overview of a Linux implementation.

Systems that belong to networks in which this option is in use should process the CIPSO option contained in each packet as specified in [CIPSO1992].

According to the option syntax specified in [CIPSO1992], the following validation check should be performed:

CIPSO.Length >= 6

If a packet does not pass this check, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

3.13.2.15. Sender Directed Multi-Destination Delivery (Type=149)

This option is defined in RFC 1770 [RFC1770] and originally provided unreliable UDP delivery to a set of addresses included in the option.

This option is obsolete. If a received packet contains this option, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

4. Internet Protocol Mechanisms

4.1. Fragment Reassembly

To accommodate networks with different Maximum Transmission Units (MTUs), the Internet Protocol provides a mechanism for the fragmentation of IP packets by end-systems (hosts) and/or intermediate-systems (routers). Reassembly of fragments is performed only by the end-systems.

> [Cerf1974] provides the rationale for why packet reassembly is not performed by intermediate-systems.

During the last few decades, IP fragmentation and reassembly has been exploited in a number of ways, to perform actions such as evading NIDSs, bypassing firewall rules, and performing DoS attacks.

> [Bendi1998] and [Humble1998] are examples of the exploitation of these issues for performing DoS attacks. [CERT1997] and [CERT1998b] document these issues. [Anderson2001] is a survey of fragmentation attacks. [US-CERT2001] is an example of the exploitation of IP fragmentation to bypass firewall rules. [CERT1999] describes the implementation of fragmentation attacks in Distributed Denial-of-Service (DDoS) attack tools.

The problem with IP fragment reassembly basically has to do with the complexity of the function, in a number of aspects:

- Fragment reassembly is a stateful operation for a stateless protocol (IP). The IP module at the host performing the reassembly function must allocate memory buffers both for temporarily storing the received fragments and to perform the reassembly function. Attackers can exploit this fact to exhaust memory buffers at the system performing the fragment reassembly.

- The fragmentation and reassembly mechanisms were designed at a time in which the available bandwidths were very different from the bandwidths available nowadays. With the current available bandwidths, a number of interoperability problems may arise, and these issues may be intentionally exploited by attackers to perform DoS attacks.

- Fragment reassembly must usually be performed without any knowledge of the properties of the path the fragments follow. Without this information, hosts cannot make any educated guess on how long they should wait for missing fragments to arrive.

- The fragment reassembly algorithm, as described by the IETF specifications, is ambiguous, and allows for a number of interpretations, each of which has found place in different TCP/IP stack implementations.

- The reassembly process is somewhat complex. Fragments may arrive out of order, duplicated, overlapping each other, etc. This complexity has lead to numerous bugs in different implementations of the IP protocol.

4.1.1. Security Implications of Fragment Reassembly

4.1.1.1. Problems Related to Memory Allocation

When an IP datagram is received by an end-system, it will be temporarily stored in system memory, until the IP module processes it and hands it to the protocol machine that corresponds to the encapsulated protocol.

In the case of fragmented IP packets, while the IP module may perform preliminary processing of the IP header (such as checking the header for errors and processing IP options), fragments must be kept in system buffers until all fragments are received and are reassembled into a complete Internet datagram.

As mentioned above, because the Internet layer will not usually have information about the characteristics of the path between the system and the remote host, no educated guess can be made on the amount of time that should be waited for the other fragments to arrive. Therefore, the specifications recommend to wait for a period of time that is acceptable for virtually all the possible network scenarios in which the protocols might operate. After that time has elapsed, all the received fragments for the corresponding incomplete packet are discarded.

> The original IP Specification, RFC 791 [RFC0791], states that systems should wait for at least 15 seconds for the missing fragments to arrive. Systems that follow the "Example Reassembly Procedure" described in Section 3.2 of RFC 791 may end up using a reassembly timer of up to 4.25 minutes, with a minimum of 15 seconds. Section 3.3.2 ("Reassembly") of RFC 1122 corrected this advice, stating that the reassembly timeout should be a fixed value between 60 and 120 seconds.

However, the longer the system waits for the missing fragments to arrive, the longer the corresponding system resources must be tied to the corresponding packet. The amount of system memory is finite, and even with today's systems, it can still be considered a scarce resource.

An attacker could take advantage of the uncomfortable situation the system performing fragment reassembly is in, by sending forged fragments that will never reassemble into a complete datagram. That is, an attacker would send many different fragments, with different IP IDs, without ever sending all the necessary fragments that would be needed to reassemble them into a full IP datagram. For each of the fragments, the IP module would allocate resources and tie them to the corresponding fragment, until the reassembly timer for the corresponding packet expires.

There are some implementation strategies which could increase the impact of this attack. For example, upon receipt of a fragment, some systems allocate a memory buffer that will be large enough to reassemble the whole datagram. While this might be beneficial in legitimate cases, this just amplifies the impact of the possible attacks, as a single small fragment could tie up memory buffers for the size of an extremely large (and unlikely) datagram. The implementation strategy suggested in RFC 815 [RFC0815] leads to such an implementation.

The impact of the aforementioned attack may vary depending on some specific implementation details:

- If the system does not enforce limits on the amount of memory that can be allocated by the IP module, then an attacker could tie all system memory to fragments, at which point the system would become unusable, perhaps crashing.

- If the system enforces limits on the amount of memory that can be allocated by the IP module as a whole, then, when this limit is reached, all further IP packets that arrive would be discarded, until some fragments time out and free memory is available again.

- If the system enforces limits on the amount memory that can be allocated for the reassembly of fragments, then, when this limit is reached, all further fragments that arrive would be discarded, until some fragment(s) time out and free memory is available again.

4.1.1.2. Problems That Arise from the Length of the IP Identification Field

The Internet Protocols are currently being used in environments that are quite different from the ones in which they were conceived. For instance, the availability of bandwidth at the time the Internet Protocol was designed was completely different from the availability of bandwidth in today's networks.

The Identification field is a 16-bit field that is used for the fragmentation and reassembly function. In the event a datagram gets fragmented, all the corresponding fragments will share the same {Source Address, Destination Address, Protocol, Identification number} four-tuple. Thus, the system receiving the fragments will be able to uniquely identify them as fragments that correspond to the same IP datagram. At a given point in time, there must be at most only one packet in the network with a given four-tuple. If not, an Identification number "collision" might occur, and the receiving system might end up "mixing" fragments that correspond to different IP datagrams which simply reused the same Identification number.

> For example, sending over a 1 Gbit/s path a continuous stream of (UDP) packets of roughly 1 kb size that all get fragmented into two equally sized fragments of 576 octets each (minimum reassembly buffer size) would repeat the IP Identification values within less than 0.65 seconds (assuming roughly 10% link layer overhead); with shorter packets that still get fragmented, this figure could easily drop below 0.4 seconds. With a single IP packet dropped in this short time frame, packets would start to be reassembled wrongly and continuously once in such interval until an error detection and recovery algorithm at an upper layer lets the application back out.

For each group of fragments whose Identification numbers "collide", the fragment reassembly will lead to corrupted packets. The IP payload of the reassembled datagram will be handed to the corresponding upper-layer protocol, where the error will (hopefully) be detected by some error detecting code (such as the TCP checksum) and the packet will be discarded.

An attacker could exploit this fact to intentionally cause a system to discard all or part of the fragmented traffic it gets, thus performing a DoS attack. Such an attacker would simply establish a flow of fragments with different IP Identification numbers, to trash all or part of the IP Identification space. As a result, the receiving system would use the attacker's fragments for the reassembly of legitimate datagrams, leading to corrupted packets which would later (and hopefully) get dropped.

In most cases, use of a long fragment timeout will benefit the attacker, as forged fragments will keep the IP Identification space trashed for a longer period of time.

4.1.1.3. Problems That Arise from the Complexity of the Reassembly Algorithm

As IP packets can be duplicated by the network, and each packet may take a different path to get to the destination host, fragments may arrive not only out of order and/or duplicated but also overlapping. This means that the reassembly process can be somewhat complex, with the corresponding implementation being not specifically trivial.

[Shannon2001] analyzes the causes and attributes of fragment traffic measured in several types of WANs.

During the years, a number of attacks have exploited bugs in the reassembly function of several operating systems, producing buffer overflows that have led, in most cases, to a crash of the attacked system.

4.1.1.4. Problems That Arise from the Ambiguity of the Reassembly Process

Network Intrusion Detection Systems (NIDSs) typically monitor the traffic on a given network with the intent of identifying traffic patterns that might indicate network intrusions.

In the presence of IP fragments, in order to detect illegitimate activity at the transport or application layers (i.e., any protocol layer above the network layer), a NIDS must perform IP fragment reassembly.

In order to correctly assess the traffic, the result of the reassembly function performed by the NIDS should be the same as that of the reassembly function performed by the intended recipient of the packets.

However, a number of factors make the result of the reassembly process ambiguous:

- The IETF specifications are ambiguous as to what should be done in the event overlapping fragments were received. Thus, in the presence of overlapping data, the system performing the reassembly function is free to honor either the first set of data received, the latest copy received, or any other copy received in between.

- As the specifications do not enforce any specific fragment timeout value, different systems may choose different values for the fragment timeout. This means that given a set of fragments received at some specified time intervals, some systems will reassemble the fragments into a full datagram, while others may timeout the fragments and therefore drop them.

- As mentioned before, as the fragment buffers get full, a DoS condition will occur unless some action is taken. Many systems flush part of the fragment buffers when some threshold is reached. Thus, depending on fragment load, timing issues, and flushing policy, a NIDS may get incorrect assumptions about how (and if) fragments are being reassembled by their intended recipient.

As originally discussed by [Ptacek1998], these issues can be exploited by attackers to evade intrusion detection systems.

There exist freely available tools to forcefully fragment IP datagrams so as to help evade Intrusion Detection Systems. Frag router [Song1999] is an example of such a tool; it allows an attacker to perform all the evasion techniques described in [Ptacek1998]. Ftester [Barisani2006] is a tool that helps to audit systems regarding fragmentation issues.

4.1.1.5. Problems That Arise from the Size of the IP Fragments

One approach to fragment filtering involves keeping track of the results of applying filter rules to the first fragment (i.e., the fragment with a Fragment Offset of 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 Identification number. When the initial 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 Fragment Offset come in, look up the list element with a matching Source Address/Destination Address/Protocol/ Identification and apply the stored result (pass or block). When a fragment with a zero MF bit is seen, free the list element. Unfortunately, the rules of this type of packet filter can usually be bypassed. [RFC1858] describes the details of the involved technique.

4.1.2. Possible Security Improvements

4.1.2.1. Memory Allocation for Fragment Reassembly

A design choice usually has to be made as to how to allocate memory to reassemble the fragments of a given packet. There are basically two options:

- Upon receipt of the first fragment, allocate a buffer that will be large enough to concatenate the payload of each fragment.

- Upon receipt of the first fragment, create the first node of a linked list to which each of the following fragments will be linked. When all fragments have been received, copy the IP payload of each of the fragments (in the correct order) to a separate buffer that will be handed to the protocol being encapsulated in the IP payload.

While the first of the choices might seem to be the most straightforward, it implies that even when a single small fragment of a given packet is received, the amount of memory that will be allocated for that fragment will account for the size of the complete IP datagram, thus using more system resources than what is actually needed.

Furthermore, the only situation in which the actual size of the whole datagram will be known is when the last fragment of the packet is received first, as that is the only packet from which the total size of the IP datagram can be asserted. Otherwise, memory should be allocated for the largest possible packet size (65535 bytes).

The IP module should also enforce a limit on the amount of memory that can be allocated for IP fragments, as well as a limit on the number of fragments that at any time will be allowed in the system. This will basically limit the resources spent on the reassembly process, and prevent an attacker from trashing the whole system memory.

Furthermore, the IP module should keep a different buffer for IP fragments than for complete IP datagrams. This will basically separate the effects of fragment attacks on non-fragmented traffic. Most TCP/IP implementations, such as that in Linux and those in BSD-derived systems, already implement this.

[Jones2002] analyzes the amount of memory that may be needed for the fragment reassembly buffer depending on a number of network characteristics.

4.1.2.2. Flushing the Fragment Buffer

In the case of those attacks that aim to consume the memory buffers used for fragments, and those that aim to cause a collision of IP Identification numbers, there are a number of countermeasures that can be implemented.

Even with these countermeasures in place, there is still the issue of what to do when the buffer pool used for IP fragments gets full. Basically, if the fragment buffer is full, no instance of communication that relies on fragmentation will be able to progress.

Unfortunately, there are not many options for reacting to this situation. If nothing is done, all the instances of communication that rely on fragmentation will experience a denial of service. Thus, the only thing that can be done is flush all or part of the fragment buffer, on the premise that legitimate traffic will be able to make use of the freed buffer space to allow communication flows to progress.

There are a number of factors that should be taken into consideration when flushing the fragment buffers. First, if a fragment of a given packet (i.e., fragment with a given Identification number) is flushed, all the other fragments that correspond to the same datagram should be flushed. As in order for a packet to be reassembled all of its fragments must be received by the system performing the reassembly function, flushing only a subset of the fragments of a given packet would keep the corresponding buffers tied to fragments that would never reassemble into a complete datagram. Additionally, care must be taken so that, in the event that subsequent buffer flushes need to be performed, it is not always the same set of fragments that get dropped, as such a behavior would probably cause a selective DoS to the traffic flows to which that set of fragments belongs.

Many TCP/IP implementations define a threshold for the number of fragments that, when reached, triggers a fragment-buffer flush. Some systems flush 1/2 of the fragment buffer when the threshold is reached. As mentioned before, the idea of flushing the buffer is to create some free space in the fragment buffer, on the premise that this will allow for new and legitimate fragments to be processed by the IP module, thus letting communication survive the overwhelming situation. On the other hand, the idea of flushing a somewhat large portion of the buffer is to avoid flushing always the same set of packets.

4.1.2.3. A More Selective Fragment Buffer Flushing Strategy

One of the difficulties in implementing countermeasures for the fragmentation attacks described throughout Section 4.1 is that it is difficult to perform validation checks on the received fragments. For instance, the fragment on which validity checks could be performed, the first fragment, may be not the first fragment to arrive at the destination host.

Fragments cannot only arrive out of order because of packet reordering performed by the network, but also because the system (or systems) that fragmented the IP datagram may indeed transmit the fragments out of order. A notable example of this is the Linux TCP/IP stack, which transmits the fragments in reverse order.

This means that we cannot enforce checks on the fragments for which we allocate reassembly resources, as the first fragment we receive for a given packet may be some other fragment than the first one (the one with an Fragment Offset of 0).

However, at the point in which we decide to free some space in the fragment buffer, some refinements can be done to the flushing policy. The first thing we would like to do is to stop different types of traffic from interfering with each other. This means, in principle, that we do not want fragmented UDP traffic to interfere with fragmented TCP traffic. In order to implement this traffic separation for the different protocols, a different fragment buffer pool would be needed, in principle, for each of the 256 different protocols that can be encapsulated in an IP datagram.

We believe a trade-off is to implement two separate fragment buffers: one for IP datagrams that encapsulate IPsec packets and another for the rest of the traffic. This basically means that traffic not protected by IPsec will not interfere with those flows of communication that are being protected by IPsec.

The processing of each of these two different fragment buffer pools would be completely independent from each other. In the case of the IPsec fragment buffer pool, when the buffers needs to be flushed, the following refined policy could be applied:

- First, for each packet for which the IPsec header has been received, check that the Security Parameters Index (SPI) field of the IPsec header corresponds to an existing IPsec Security Association (SA), and probably also check that the IPsec sequence number is valid. If the check fails, drop all the fragments that correspond to this packet.

- Second, if still more fragment buffers need to be flushed, drop all the fragments that correspond to packets for which the full IPsec header has not yet been received. The number of packets for which this flushing is performed depends on the amount of free space that needs to be created.

- Third, if after flushing packets with invalid IPsec information (First step), and packets on which validation checks could not be performed (Second step), there is still not enough space in the

fragment buffer, drop all the fragments that correspond to packets that passed the checks of the first step, until the necessary free space is created.

The rationale behind this policy is that, at the point of flushing fragment buffers, we prefer to keep those packets on which we could successfully perform a number of validation checks, over those packets on which those checks failed, or the checks could not even be performed.

By checking both the IPsec SPI and the IPsec sequence number, it is virtually impossible for an attacker that is off-path to perform a DoS attack to communication flows being protected by IPsec.

Unfortunately, some IP implementations (such as that in Linux [Linux]), when performing fragmentation, send the corresponding fragments in reverse order. In such cases, at the point of flushing the fragment buffer, legitimate fragments will receive the same treatment as the possible forged fragments.

This refined flushing policy provides an increased level of protection against this type of resource exhaustion attack, while not making the situation of out-of-order IPsec-secured traffic worse than with the simplified flushing policy described in the previous section.

4.1.2.4. Reducing the Fragment Timeout

RFC 1122 [RFC1122] states that the reassembly timeout should be a fixed value between 60 and 120 seconds. The rationale behind these long timeout values is that they should accommodate any path characteristics, such as long-delay paths. However, it must be noted that this timer is really measuring inter-fragment delays, or, more specifically, fragment jitter.

If all fragments take paths of similar characteristics, the inter-fragment delay will usually be, at most, a few seconds. Nevertheless, even if fragments take different paths of different characteristics, the recommended 60 to 120 seconds are, in practice, excessive.

Some systems have already reduced the fragment timeout to 30 seconds [Linux]. The fragment timeout could probably be further reduced to approximately 15 seconds; although further research on this issue is necessary.

It should be noted that in network scenarios of long-delay and high-bandwidth (usually referred to as "Long-Fat Networks"), using a long fragment timeout would likely increase the probability of collision of IP ID numbers. Therefore, in such scenarios it is highly desirable to avoid the use of fragmentation with techniques such as PMTUD [RFC1191] or PLPMTUD [RFC4821].

4.1.2.5. Countermeasure for Some NIDS Evasion Techniques

[Shankar2003] introduces a technique named "Active Mapping" that prevents evasion of a NIDS by acquiring sufficient knowledge about the network being monitored, to assess which packets will arrive at the intended recipient, and how they will be interpreted by it. [Novak2005] describes some techniques that are applied by the Snort [Snort] NIDS to avoid evasion.

4.1.2.6. Countermeasure for Firewall-Rules Bypassing

One of the classical techniques to bypass firewall rules involves sending packets in which the header of the encapsulated protocol is fragmented. Even when it would be legal (as far as the IETF specifications are concerned) to receive such a packets, the MTUs of the network technologies used in practice are not that small to require the header of the encapsulated protocol to be fragmented (e.g., see [RFC2544]). Therefore, the system performing reassembly should drop all packets which fragment the upper-layer protocol header, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

Additionally, given that many middle-boxes such as firewalls create state according to the contents of the first fragment of a given packet, it is best that, in the event an end-system receives overlapping fragments, it honors the information contained in the fragment that was received first.

RFC 1858 [RFC1858] describes the abuse of IP fragmentation to bypass firewall rules. RFC 3128 [RFC3128] corrects some errors in RFC 1858.

4.2. Forwarding

4.2.1. Precedence-Ordered Queue Service

Section 5.3.3.1 of RFC 1812 [RFC1812] states that routers should implement precedence-ordered queue service. This means that when a packet is selected for output on a (logical) link, the packet of highest precedence that has been queued for that link is sent. Section 5.3.3.2 of RFC 1812 advises routers to default to maintaining strict precedence-ordered service.

Unfortunately, given that it is trivial to forge the IP precedence field of the IP header, an attacker could simply forge a high precedence number in the packets it sends to illegitimately get better network service. If precedence-ordered queued service is not required in a particular network infrastructure, it should be disabled, and thus all packets would receive the same type of service, despite the values in their Type of Service or Differentiated Services fields.

When precedence-ordered queue service is required in the network infrastructure, in order to mitigate the attack vector discussed in the previous paragraph, edge routers or switches should be configured to police and remark the Type of Service or Differentiated Services values, according to the type of service at which each end-system has been allowed to send packets.

Bullet 4 of Section 5.3.3.3 of RFC 1812 states that routers "MUST NOT change precedence settings on packets it did not originate". However, given the security implications of the Precedence field, it is fair for routers, switches, or other middle-boxes, particularly those in the network edge, to overwrite the Type of Service (or Differentiated Services) field of the packets they are forwarding, according to a configured network policy (this is the specified behavior for DS domains [RFC2475]).

Sections 5.3.3.1 and 5.3.6 of RFC 1812 state that if precedence-ordered queue service is implemented and enabled, the router "MUST NOT discard a packet whose precedence is higher than that of a packet that is not discarded". While this recommendation makes sense given the semantics of the Precedence field, it is important to note that it would be simple for an attacker to send packets with forged high Precedence value to congest some internet router(s), and cause all (or most) traffic with a lower Precedence value to be discarded.

4.2.2. Weak Type of Service

Section 5.2.4.3 of RFC 1812 describes the algorithm for determining the next-hop address (i.e., the forwarding algorithm). Bullet 3, "Weak TOS", addresses the case in which routes contain a "type of service" attribute. It states that in case a packet contains a non-default TOS (i.e., 0000), only routes with the same TOS or with the default TOS should be considered for forwarding that packet. However, this means that if among the longest match routes for a given packet are routes with some TOS other than the one contained in the received packet, and no routes with the default TOS, the corresponding packet would be dropped. This may or may not be a desired behavior.

An alternative for the case in which among the "longest match" routes there are only routes with non-default type of service that do not match the TOS contained in the received packet, would be to use a route with any other TOS. While this route would most likely not be able to address the type of service requested by packet, it would, at least, provide a "best effort" service.

It must be noted that Section 5.3.2 of RFC 1812 allows routers to not honor the TOS field. Therefore, the proposed alternative behavior is still compliant with the IETF specifications.

> While officially specified in the RFC series, TOS-based routing is not widely deployed in the Internet.

4.2.3. Impact of Address Resolution on Buffer Management

In the case of broadcast link-layer technologies, in order for a system to transfer an IP datagram it must usually first map an IP address to the corresponding link-layer address (for example, by means of the Address Resolution Protocol (ARP) [RFC0826]) . This means that while this operation is being performed, the packets that would require such a mapping would need to be kept in memory. This may happen both in the case of hosts and in the case of routers.

This situation might be exploited by an attacker, which could send a large amount of packets to a non-existent host that would supposedly be directly connected to the attacked router. While trying to map the corresponding IP address into a link-layer address, the attacked router would keep in memory all the packets that would need to make use of that link-layer address. At the point in which the mapping function times out, depending on the policy implemented by the attacked router, only the packet that triggered the call to the mapping function might be dropped. In that case, the same operation would be repeated for every packet destined to the non-existent host. Depending on the timeout value for the mapping function, this situation might lead the router to run out of free buffer space, with the consequence that incoming legitimate packets would have to be dropped, or that legitimate packets already stored in the router's buffers might get dropped. Both of these situations would lead either to a complete DoS or to a degradation of the network service.

One countermeasure to this problem would be to drop, at the point the mapping function times out, all the packets destined to the address that timed out. In addition, a "negative cache entry" might be kept in the module performing the matching function, so that for some amount of time, the mapping function would return an error when the IP module requests to perform a mapping for some address for which the mapping has recently timed out.

A common implementation strategy for routers is that when a packet is received that requires an ARP resolution to be performed before the packet can be forwarded, the packet is dropped and the router is then engaged in the ARP procedure.

4.2.4. Dropping Packets

In some scenarios, it may be necessary for a host or router to drop packets from the output queue. In the event that one of such packets happens to be an IP fragment, and there were other fragments of the same packet in the queue, those other fragments should also be dropped. The rationale for this policy is that it is nonsensical to spend system resources on those other fragments, because, as long as one fragment is missing, it will be impossible for the receiving system to reassemble them into a complete IP datagram.

Some systems have been known to drop just a subset of fragments of a given datagram, leading to a denial-of-service condition, as only a subset of all the fragments of the packets were actually transferred to the next hop.

4.3. Addressing

4.3.1. Unreachable Addresses

It is important to understand that while there are some addresses that are supposed to be unreachable from the public Internet (such as the private IP addresses described in RFC 1918 [RFC1918], or the "loopback" address), there are a number of tricks an attacker can perform to reach those IP addresses that would otherwise be unreachable (e.g., exploit the LSRR or SSRR IP options). Therefore, when applicable, packet filtering should be performed at the private network boundary to assure that those addresses will be unreachable.

Similarly, link-local unicast addresses [RFC3927] and multicast addresses with limited scope (link- and site-local addresses) should not be accessible from outside the proper network boundaries and not be passed across these boundaries.

[RFC5735] provides a summary of special use IPv4 addresses.

4.3.2. Private Address Space

The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets:

- 10.0.0.0 - 10.255.255.255 (10/8 prefix)

- 172.16.0.0 - 172.31.255.255 (172.16/12 prefix)

- 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

Use of these address blocks is described in RFC 1918 [RFC1918].

Where applicable, packet filtering should be performed at the organizational perimeter to assure that these addresses are not reachable from outside the private network where such addresses are employed.

4.3.3. Former Class D Addresses (224/4 Address Block)

The former Class D addresses correspond to the 224/4 address block and are used for Internet multicast. Therefore, if a packet is received with a "Class D" address as the Source Address, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop). Additionally, if an IP packet with a multicast Destination Address is received for a connection-oriented protocol (e.g., TCP), the packet should be dropped (see Section 4.3.5), and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

4.3.4. Former Class E Addresses (240/4 Address Block)

The former Class E addresses correspond to the 240/4 address block, and are currently reserved for experimental use. As a result, a most routers discard packets that contain a "Class" E address as the Source Address or Destination Address. If a packet is received with a 240/4 address as the Source Address and/or the Destination Address, the packet should be dropped and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

It should be noted that the broadcast address 255.255.255.255 still must be treated as indicated in Section 4.3.7 of this document.

4.3.5. Broadcast/Multicast Addresses and Connection-Oriented Protocols

For connection-oriented protocols, such as TCP, shared state is maintained between only two endpoints at a time. Therefore, if an IP packet with a multicast (or broadcast) Destination Address is received for a connection-oriented protocol (e.g., TCP), the packet should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

4.3.6. Broadcast and Network Addresses

Originally, the IETF specifications did not permit IP addresses to have the value 0 or -1 (shorthand for all bits set to 1) for any of the Host number, network number, or subnet number fields, except for the cases indicated in Section 4.3.7. However, this changed fundamentally with the deployment of Classless Inter-Domain Routing (CIDR) [RFC4632], as with CIDR a system cannot know a priori what the subnet mask is for a particular IP address.

Many systems now allow administrators to use the values 0 or -1 for those fields. Despite that according to the original IETF specifications these addresses are illegal, modern IP implementations should consider these addresses to be valid.

4.3.7. Special Internet Addresses

RFC 1812 [RFC1812] discusses the use of some special Internet addresses, which is of interest to perform some sanity checks on the Source Address and Destination Address fields of an IP packet. It uses the following notation for an IP address:

{ <Network-prefix>, <Host-number> }

where the length of the network prefix is generally implied by the network mask assigned to the IP interface under consideration.

> RFC 1122 [RFC1122] contained a similar discussion of special Internet addresses, including some of the form { <Network-prefix>, <Subnet-number>, <Host-number> }. However, as explained in Section 4.2.2.11 of RFC 1812, in a CIDR world, the subnet number is clearly an extension of the network prefix and cannot be distinguished from the remainder of the prefix.

{0, 0}

This address means "this host on this network". It is meant to be used only during the initialization procedure, by which the host learns its own IP address.

If a packet is received with 0.0.0.0 as the Source Address for any purpose other than bootstrapping, the corresponding packet should be silently dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop). If a packet is received with 0.0.0.0 as the Destination Address, it should be silently dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

{0, Host number}

This address means "the specified host, in this network". As in the previous case, it is meant to be used only during the initialization procedure by which the host learns its own IP address. If a packet is received with such an address as the Source Address for any purpose other than bootstrapping, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop). If a packet is received with such an address as the Destination Address, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

{-1, -1}

This address is the local broadcast address. It should not be used as a source IP address. If a packet is received with 255.255.255.255 as the Source Address, it should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

> Some systems, when receiving an ICMP echo request, for example, will use the Destination Address in the ICMP echo request packet as the Source Address of the response they send (in this case, an ICMP echo reply). Thus, when such systems receive a request sent to a broadcast address, the Source Address of the response will contain a broadcast address. This should be considered a bug, rather than a malicious use of the limited broadcast address.

{Network number, -1}

This is the directed broadcast to the specified network. As recommended by RFC 2644 [RFC2644], routers should not forward network-directed broadcasts. This avoids the corresponding network from being utilized as, for example, a "smurf amplifier" [CERT1998a].

As noted in Section 4.3.6 of this document, many systems now allow administrators to configure these addresses as unicast addresses for network interfaces. In such scenarios, routers should forward these addresses as if they were traditional unicast addresses.

In some scenarios, a host may have knowledge about a particular IP address being a network-directed broadcast address, rather than a unicast address (e.g., that IP address is configured on the local system as a "broadcast address"). In such scenarios, if a system can infer that the Source Address of a received packet is a network-

directed broadcast address, the packet should be dropped, and this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

As noted in Section 4.3.6 of this document, with the deployment of CIDR [RFC4632], it may be difficult for a system to infer whether a particular IP address that does not belong to a directly attached subnet is a broadcast address.

{127.0.0.0/8, any}

This is the internal host loopback address. Any packet that arrives on any physical interface containing this address as the Source Address, the Destination Address, or as part of a source route (either LSRR or SSRR), should be dropped.

For example, packets with a Destination Address in the 127.0.0.0/8 address block that are received on an interface other than loopback should be silently dropped. Packets received on any interface other than loopback with a Source Address corresponding to the system receiving the packet should also be dropped.

In all the above cases, when a packet is dropped, this event should be logged (e.g., a counter could be incremented to reflect the packet drop).

5. Security Considerations

This document discusses the security implications of the Internet Protocol (IP) and a number of implementation strategies that help to mitigate a number of vulnerabilities found in the protocol during the last 25 years or so.

6. Acknowledgements

The author wishes to thank Alfred Hoenes for providing very thorough reviews of earlier versions of this document, thus leading to numerous improvements.

The author would like to thank Jari Arkko, Ron Bonica, Stewart Bryant, Adrian Farrel, Joel Jaeggli, Warren Kumari, Bruno Rohee, and Andrew Yourtchenko for providing valuable comments on earlier versions of this document.

This document was written by Fernando Gont on behalf of the UK CPNI (United Kingdom's Centre for the Protection of National Infrastructure), and is heavily based on the "Security Assessment of the Internet Protocol" [CPNI2008] published by the UK CPNI in 2008.

The author would like to thank Randall Atkinson, John Day, Juan Fraschini, Roque Gagliano, Guillermo Gont, Martin Marino, Pekka Savola, and Christos Zoulas for providing valuable comments on earlier versions of [CPNI2008], on which this document is based.

The author would like to thank Randall Atkinson and Roque Gagliano, who generously answered a number of questions.

Finally, the author would like to thank UK CPNI (formerly NISCC) for their continued support.

7. References

7.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

[RFC1038] St. Johns, M., "Draft revised IP security option", RFC 1038, January 1988.

[RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, July 1988.

[RFC1108] Kent, S., "U.S", RFC 1108, November 1991.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1349] Almquist, P., "Type of Service in the Internet Protocol Suite", RFC 1349, July 1992.

[RFC1393] Malkin, G., "Traceroute Using an IP Option", RFC 1393, January 1993.

[RFC1770] Graff, C., "IPv4 Option for Sender Directed Multi-Destination Delivery", RFC 1770, March 1995.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

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

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2644] Senie, D., "Changing the Default for Directed Broadcasts in Routers", BCP 34, RFC 2644, August 1999.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, March 2004.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005.

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

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, August 2006.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, October 2007.

[RFC5350] Manner, J. and A. McDonald, "IANA Considerations for the IPv4 and IPv6 Router Alert Options", RFC 5350, September 2008.

[RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses", BCP 153, RFC 5735, January 2010.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

7.2. Informative References

[Anderson2001]
Anderson, J., "An Analysis of Fragmentation Attacks", 2001, <http://www.ouah.org/fragma.html>.

[Arkin2000]
Arkin, "IP TTL Field Value with ICMP (Oops - Identifying Windows 2000 again and more)", 2000, <http://ofirarkin.files.wordpress.com/2008/11/ofirarkin2000-06.pdf>.

[Barisani2006]
Barisani, A., "FTester - Firewall and IDS testing tool", 2001, <http://dev.inversepath.com/trac/ftester>.

[Bellovin1989]
Bellovin, S., "Security Problems in the TCP/IP Protocol Suite", Computer Communication Review Vol. 19, No. 2, pp. 32-48, 1989.

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

[Bendi1998]
Bendi, "Bonk exploit", 1998, <http://www.insecure.org/sploits/95.NT.fragmentation.bonk.html>.

[Biondi2007]
Biondi, P. and A. Ebalard, "IPv6 Routing Header Security", CanSecWest 2007 Security Conference, 2007, <http://www.secdev.org/conf/IPv6_RH_security-csw07.pdf>.

[CERT1996a]
CERT, "CERT Advisory CA-1996-01: UDP Port Denial-of-Service Attack", 1996, <http://www.cert.org/advisories/CA-1996-01.html>.

[CERT1996b]
CERT, "CERT Advisory CA-1996-21: TCP SYN Flooding and IP Spoofing Attacks", 1996, <http://www.cert.org/advisories/CA-1996-21.html>.

[CERT1996c]
CERT, "CERT Advisory CA-1996-26: Denial-of-Service Attack via ping", 1996, <http://www.cert.org/advisories/CA-1996-26.html>.

[CERT1997] CERT, "CERT Advisory CA-1997-28: IP Denial-of-Service Attacks", 1997, <http://www.cert.org/advisories/CA-1997-28.html>.

[CERT1998a]
CERT, "CERT Advisory CA-1998-01: Smurf IP Denial-of-Service Attacks", 1998, <http://www.cert.org/advisories/CA-1998-01.html>.

[CERT1998b]
CERT, "CERT Advisory CA-1998-13: Vulnerability in Certain TCP/IP Implementations", 1998, <http://www.cert.org/advisories/CA-1998-13.html>.

[CERT1999] CERT, "CERT Advisory CA-1999-17: Denial-of-Service Tools", 1999, <http://www.cert.org/advisories/CA-1999-17.html>.

[CERT2003] CERT, "CERT Advisory CA-2003-15: Cisco IOS Interface Blocked by IPv4 Packet", 2003, <http://www.cert.org/advisories/CA-2003-15.html>.

[CIPSO1992]
CIPSO, "COMMERCIAL IP SECURITY OPTION (CIPSO 2.2)", Work in Progress, 1992.

[CIPSOWG1994]
CIPSOWG, "Commercial Internet Protocol Security Option (CIPSO) Working Group", 1994, <http://www.ietf.org/proceedings/94jul/charters/cipso-charter.html>.

[CPNI2008] Gont, F., "Security Assessment of the Internet Protocol", 2008, <http://www.cpni.gov.uk/Docs/InternetProtocol.pdf>.

[Cerf1974] Cerf, V. and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications Vol. 22, No. 5, May 1974, pp. 637-648, 1974.

[Cisco2003]
Cisco, "Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 packet", 2003, <http://www.cisco.com/en/US/products/products_security_advisory09186a00801a34c2.shtml>.

[Cisco2008]
Cisco, "Cisco IOS Security Configuration Guide, Release 12.2", 2003, <http://www.cisco.com/en/US/docs/ios/12_2/security/configuration/guide/scfipso.html>.

[Clark1988]
Clark, D., "The Design Philosophy of the DARPA Internet Protocols", Computer Communication Review Vol. 18, No. 4, 1988.

[Ed3f2002] Ed3f, "Firewall spotting and networks analysis with a broken CRC", Phrack Magazine, Volume 0x0b, Issue 0x3c, Phile #0x0c of 0x10, 2002, <http://www.phrack.org/issues.html?issue=60&id=12&mode=txt>.

[FIPS1994] FIPS, "Standard Security Label for Information Transfer", Federal Information Processing Standards Publication. FIP PUBS 188, 1994, <http://csrc.nist.gov/publications/fips/fips188/fips188.pdf>.

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

[GIAC2000] GIAC, "Egress Filtering v 0.2", 2000, <http://www.sans.org/y2k/egress.htm>.

[Gont2006] Gont, F., "Advanced ICMP packet filtering", 2006, <http://www.gont.com.ar/papers/icmp-filtering.html>.

[Haddad2004]
Haddad, I. and M. Zakrzewski, "Security Distribution for Linux Clusters", Linux Journal, 2004, <http://www.linuxjournal.com/article/6943>.

[Humble1998]
Humble, "Nestea exploit", 1998, <http://www.insecure.org/sploits/linux.PalmOS.nestea.html>.

[IANA_ET] IANA, "Ether Types", <http://www.iana.org/assignments/ethernet-numbers>.

[IANA_IP_PARAM]
IANA, "IP Parameters", <http://www.iana.org/assignments/ip-parameters>.

[IANA_PROT_NUM]
IANA, "Protocol Numbers", <http://www.iana.org/assignments/protocol-numbers>.

[IRIX2008] IRIX, "IRIX 6.5 trusted_networking(7) manual page", 2008, <http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=0650&db=man&fname=/usr/share/catman/a_man/cat7/trusted_networking.z>.

[Jones2002]
Jones, R., "A Method Of Selecting Values For the Parameters Controlling IP Fragment Reassembly", 2002, <ftp://ftp.cup.hp.com/dist/networking/briefs/ip_reass_tuning.txt>.

[Kenney1996]
Kenney, M., "The Ping of Death Page", 1996, <http://www.insecure.org/sploits/ping-o-death.html>.

[Kent1987] Kent, C. and J. Mogul, "Fragmentation considered harmful", Proc. SIGCOMM '87 Vol. 17, No. 5, October 1987, 1987.

[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>.

[Kohno2005]
Kohno, T., Broido, A., and kc. Claffy, "Remote Physical Device Fingerprinting", IEEE Transactions on Dependable and Secure Computing Vol. 2, No. 2, 2005.

[LBNL2006] LBNL/NRG, "arpwatch tool", 2006, <http://ee.lbl.gov/>.

[Linux] Linux Kernel Organization, "The Linux Kernel Archives", <http://www.kernel.org>.

[Microsoft1999]
Microsoft, "Microsoft Security Program: Microsoft Security Bulletin (MS99-038). Patch Available for "Spoofed Route Pointer" Vulnerability", 1999, <http://www.microsoft.com/technet/security/bulletin/ms99-038.mspx>.

[NISCC2004]
NISCC, "NISCC Vulnerability Advisory 236929: Vulnerability Issues in TCP", 2004, <http://www.cpni.gov.uk>.

[NISCC2005]
NISCC, "NISCC Vulnerability Advisory 532967/NISCC/ICMP: Vulnerability Issues in ICMP packets with TCP payloads", 2005, <http://www.gont.com.ar/advisories/index.html>.

[NISCC2006]
NISCC, "NISCC Technical Note 01/2006: Egress and Ingress Filtering", 2006, <http://www.cpni.gov.uk>.

[Northcutt2000]
Northcut, S. and Novak, "Network Intrusion Detection - An Analyst's Handbook", Second Edition New Riders Publishing, 2000.

[Novak2005]
Novak, "Target-Based Fragmentation Reassembly", 2005, <http://www.snort.org/assets/165/target_based_frag.pdf>.

[OpenBSD-PF]
Sanfilippo, S., "PF: Scrub (Packet Normalization)", 2010, <ftp://ftp.openbsd.org/pub/OpenBSD/doc/pf-faq.pdf>.

[OpenBSD1998]
OpenBSD, "OpenBSD Security Advisory: IP Source Routing Problem", 1998, <http://www.openbsd.org/advisories/sourceroute.txt>.

[Paxson2001] Paxson, V., Handley, M., and C. Kreibich, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics", USENIX Conference, 2001.

[Ptacek1998] Ptacek, T. and T. Newsham, "Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection", 1998, <http://www.aciri.org/vern/Ptacek-Newsham-Evasion-98.ps>.

[RFC0815] Clark, D., "IP datagram reassembly algorithms", RFC 815, July 1982.

[RFC1858] Ziemba, G., Reed, D., and P. Traina, "Security Considerations for IP Fragment Filtering", RFC 1858, October 1995.

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.

[RFC3128] Miller, I., "Protection Against a Variant of the Tiny Fragment Attack (RFC 1858)", RFC 3128, June 2001.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

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

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.

[RFC5559] Eardley, P., "Pre-Congestion Notification (PCN) Architecture", RFC 5559, June 2009.

[RFC5570] StJohns, M., Atkinson, R., and G. Thomas, "Common Architecture Label IPv6 Security Option (CALIPSO)", RFC 5570, July 2009.

[RFC5670] Eardley, P., "Metering and Marking Behaviour of PCN-Nodes", RFC 5670, November 2009.

[RFC5696] Moncaster, T., Briscoe, B., and M. Menth, "Baseline Encoding and Transport of Pre-Congestion Information", RFC 5696, November 2009.

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

[ROUTER-ALERT]
Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", Work in Progress, June 2011.

[SELinux2009]
NSA, "Security-Enhanced Linux", <http://www.nsa.gov/research/selinux/>.

[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>.

[Shankar2003]
Shankar, U. and V. Paxson, "Active Mapping: Resisting NIDS Evasion Without Altering Traffic", 2003, <http://www.icir.org/vern/papers/activemap-oak03.pdf>.

[Shannon2001]
Shannon, C., Moore, D., and K. Claffy, "Characteristics of Fragmented IP Traffic on Internet Links", 2001.

[Silbersack2005]
Silbersack, M., "Improving TCP/IP security through randomization without sacrificing interoperability", EuroBSDCon 2005 Conference, 2005, <http://www.silby.com/eurobsdcon05/eurobsdcon_slides.pdf>.

[Snort] Sourcefire, Inc., "Snort", <http://www.snort.org>.

[Solaris2007]
Oracle, "ORACLE SOLARIS WITH TRUSTED EXTENSIONS", 2007, <http://www.oracle.com/us/products/servers-storage/solaris/solaris-trusted-ext-ds-075583.pdf>.

[Song1999] Song, D., "Frag router tool", <http://www.monkey.org/~dugsong/fragroute/>.

[SpooferProject]
MIT ANA, "Spoofer Project", 2010, <http://spoofer.csail.mit.edu/index.php>.

[US-CERT2001]
US-CERT, "US-CERT Vulnerability Note VU#446689: Check Point FireWall-1 allows fragmented packets through firewall if Fast Mode is enabled", 2001, <http://www.kb.cert.org/vuls/id/446689>.

[US-CERT2002]
US-CERT, "US-CERT Vulnerability Note VU#310387: Cisco IOS discloses fragments of previous packets when Express Forwarding is enabled", 2002.

[Watson2004]
Watson, P., "Slipping in the Window: TCP Reset Attacks", CanSecWest Conference, 2004.

[Zakrzewski2002]
Zakrzewski, M. and I. Haddad, "Linux Distributed Security Module", 2002, <http://www.linuxjournal.com/article/6215>.

[daemon91996]
daemon9, route, and infinity, "IP-spoofing Demystified (Trust-Relationship Exploitation)", Phrack Magazine, Volume Seven, Issue Forty-Eight, File 14 of 18, 1988, <http://www.phrack.org/issues.html?issue=48&id=14&mode=txt>.

Author's Address

Fernando Gont
UK Centre for the Protection of National Infrastructure

EMail: fernando@gont.com.ar
URI: http://www.cpni.gov.uk