Too Big or Too Small? The PTB-PTS ICMP-based Attack against IPsec Gateways

Ludovic Jacquin, Vincent Roca, Jean-Louis Roch

Too Big or Too Small? The PTB-PTS ICMP-based Attack against IPsec Gateways

Ludovic Jacquin
Inria, France
ludovic.jacquin@gmail.com

Vincent Roca
Inria, France
vincent.roca@inria.fr

Jean-Louis Roch
Inria, Grenoble Université, Grenoble INP, LIG, France
jean-louis.roch@imag.fr

***Abstract*—This work introduces the "Packet Too Big"-"Packet Too Small" ICMP based attack against IPsec gateways. We explain how an attacker having eavesdropping and packet injection capabilities, from the insecure network where he only sees encrypted packets, can force a gateway to reduce the Path MTU of an IPsec tunnel to the minimum, which triggers severe issues for the hosts behind this gateway: depending on the Path MTU discovery algorithm in use, the attack either creates a Denial of Service or major performance penalties. This attack highlights two fundamental problems that we discuss, along with potential counter-measures to mitigate the attack while keeping ICMP benefits.**

I. Introduction

IPsec and ESP [1][2] offer a convenient secure tunnelling capability that is largely used to interconnect remote networks, or a remote host to its home network, throughout an unsecured network, typically the Internet.

Naturally IPsec interacts with the IP protocol suite, and in particular with the Internet Control Message Protocol (ICMP). A first goal of ICMP is to exchange control and error messages, like packet processing error notifications. ICMP is also involved in several functionalities and in particular the Path Maximum Transmission Unit discovery (PMTUd) mechanism [3], [4], [5] whose goal is to find the maximum packet size on a path that avoids packet fragmentation. Such a mechanism is essential from a performance point of view: if a packet is too large, its fragmentation and reassembly will negatively impact performance. At the other extreme, if a packet is significantly smaller than the maximum size permitted throughout the path, it will also negatively impact performance. Assessing the correct packet size on a path is therefore essential. But ICMP is also known to be a cause of attacks and therefore there is an incentive for a network administrator to filter out these packets. A balance is therefore required between these contradictory objectives, and it is recognized that only a subset of ICMP packets should be considered by IPsec gateways (we detail this in section III-C).

The problem this work addresses is the following: how can an attacker exploit ICMP to mount Denial of Service (DoS) attacks to the users of IPsec gateways? We assume the attacker is located in the Internet (the insecure network) and has realistic capabilities. Our contributions are threefold:

- we identify a new ICMP based attack on IPsec gateways, called "Packet Too Big"-"Packet Too Small" (PTB-PTS) that departs from the already known ICMP attacks. Our attack uses realistic assumptions in terms of the attacker capabilities, making it a realistic threat: eavesdropping and traffic injection capabilities in the untrusted network (where the attacker only sees ciphered packets) are sufficient, which is fulfilled for instance if the attacker and the target are both attached to the same insecure WiFi network;
- we give an account of a real exploit, on a testbed running a recent Debian distribution, with default system and IPsec configurations, where an external attacker can either stall TCP connections going through the IPsec tunnel, or create major performance penalties. We show this attack is effective no matter whether the hosts, on the trusted networks, use the classic PMTUd or the new Packetization Layer PMTUd (PLPMTUd) algorithms;
- we analyze and discuss the two fundamental problems that made this attack possible, namely: (1) the impossibility to distinguish legitimate from illegitimate ICMP packets coming from the untrusted network, and (2) the contradictions in the way Path MTU is managed by end hosts when this Path MTU approaches the minimum packet size any link should support, whereas at the same time tunnelling is needed on the path;

The paper is organized as follows: section II introduces the network and attacker model; section III describes IPsec and ICMP; section IV details our attack, illustrated with a real exploit; section V discusses the fundamental problems and some counter-measures; section VI position our work in front of related works; and finally we conclude.

II. Network and attacker models

A. The red-black network model

Our network model identifies two zones: the trusted areas, consisting of networks and hosts behind IPsec gateways, also called *"red networks"*, and the outer world considered as untrusted, typically the Internet, also called *"black network"*. Trusted areas are interconnected by gateways through IPsec tunnels (site-to-site configuration of Fig. 1). An isolated host having IPsec capabilities can also establish an IPsec tunnel to its remote home network (host-to-site configuration).

The aim of the IPsec gateway is to secure the traffic between remote trusted networks and/or hosts by encrypting packets sent over the untrusted black network. Since the gateways are directly connected to the black network, they are naturally the first line of defense against attackers.

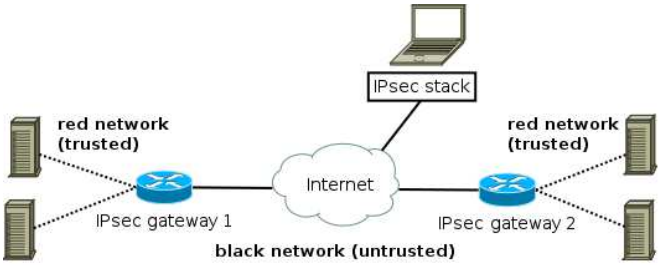

Fig. 1: Network model showing the site-to-site (bottom) and host-to-site (top) configurations.

B. The attacker model: a realistic scenario

Because of the network model, all the attacks are conducted by adversaries located on the external black network. We assume **an attacker can both eavesdrop the traffic in the** `IPsec` **tunnel and inject forged packets** (we justify these choices below). We also assume **an attacker has no way to decrypt packets nor encrypt its own packets** because the underlying `IPsec` cryptographic building blocks and key exchange protocols are considered secure. The goal of the attacker is to launch a DoS against the secure tunnel service provided by `IPsec` gateways, for both kinds of `IPsec` configurations: host-to-site and site-to-site.

This is a realistic attacker model. For instance, the attacker can be located on a compromised router along the path followed by an `IPsec` tunnel, in the black network[1]. But more simply it can also be an attacker attached to the same insecure WiFi network (e.g. that does not use WPA/WPA2 security) as the target user that connects to his home network through an `IPsec` VPN, which is a rather common situation.

III. IPsec and ICMP in a nutshell

In this section we give some background on `IPsec` [1], `ICMP` [6], and the two standardized algorithms to discover the maximum packet size along a path. Then we discuss standard recommendations for `ICMP` processing policies within `IPsec`.

A. IPsec overview

`IPsec` has two core protocols, `AH` [7] and `ESP` [2], and two modes of operation, transport and tunnel. In our work we consider the usual solution, `IPsec`/`ESP` in tunnel mode, where `IPsec`/`ESP` provide confidentiality, authentication, integrity and anti-replay services. More precisely, the incoming `IP` datagram (called inner `IP` packet) arriving at the tunnel endpoint is ciphered by `ESP` and tunneled in a new `IP` header (called outer `IP` packet). Upon leaving the other tunnel endpoint, the opposite operations take place.

`IPsec` requires three major databases: the *Security Policy Database (SPD)*, the *Security Association Database (SAD)* and the *Peer Authorization Database (PAD)*. Throughout the paper, we only focus on the SAD since it stores important information about active tunnels, like ciphering keys (initialized by the `IKEv2` [8] protocol) and the PMTU (see section III-B1) for this tunnel, which plays a key role in our attack. Each entry in this SAD is identified by a Security Parameters Index (SPI), that is copied in clear in the outer `IP` packet header.

[1]The 2013 revelations about large scale surveillance demonstrate this assumption is realistic.

B. Path MTU discovery overview

Path MTU discovery (or PMTUd) is a key mechanism for optimum network performance since it enables a sender to determine the appropriate packet size along a path dynamically (the path may change over time) [9], [10], [11]. Two complementary PMTU discovery algorithms have been standardized and are in use nowadays.

1) The legacy PMTUd mechanism: PMTUd [3] is the legacy approach. Let us illustrate its behavior in an `IPv4` (resp. `IPv6`) network. A sender sets the *Don't Fragment* (DF) bit in a packet[2]. If a router cannot transmit this packet because of its size, it must send back to the sender an `ICMP` "Destination unreachable"/"Fragmentation needed" packet (resp. an `ICMPv6` "Packet Too Big"/"Fragmentation needed"), along with the next hop MTU information. In the following we will call these error packets `ICMP PTB` (Packet Too Big), regardless of whether `IPv4` or `IPv6` is used. Iteratively, upon receiving such an `ICMP PTB` packet, the sender decreases the packet size until it reaches the lowest MTU on the path to the destination. The PMTU is then found and will be used by the sender for outgoing packets sent to this destination. Since the path can change dynamically (e.g., due to re-routing), this process needs to be performed periodically. So we see that `ICMP` is heavily used in the this PMTUd algorithm.

Although efficient, the PMTUd approach suffers from several limits, mainly because `ICMP` packets are often filtered out by some routers/firewalls [12] along their route to the sender. In that case the sender needs another technique to discover the Path MTU.

2) The Packetization Layer PMTUd mechanism: To overcome these issues, the IETF developed a new Path MTU discovery mechanism that does not rely on `ICMP`, the Packetization Layer PMTUd (PLPMTUd) [13]. Instead of using `ICMP`, it relies on a packetization layer protocol with an acknowledgement mechanism, such as `TCP`. Using this protocol (e.g. `TCP`), the sender sends probing packets of a specific size to the destination. If the probing packet is acknowledged, the sender validates that the PMTU is at least equal to the probing packet size, while a time-out indicates that the PMTU is smaller. With `TCP`, any data segment can be used as a probing packet if enough data is available to fill in the payload. Here also, because the path may change, the PLPMTUd process needs to be performed periodically.

C. ICMP processing in IPsec

`IPsec` specifies dedicated rules to process `ICMP` packets and administrators need to decide, through configuration, how to handle them. More precisely, a distinction is made between error and informational `ICMP` packets, as well where they are coming from ([1] Sections 5 and 8). The recommended treatments are summarized in Table I. For our attack, the most important row is the last one that corresponds to untrusted `ICMP` error packets. If the policy is unspecified, there are

[2]This is useless in IPv6 since fragmentation is not supported any more.

strong incentives to consider separately the case of ICMP PTB error packets in order to enable PMTUd [1].

ICMP type	origin	recommended treatment
info. message	trusted	administrator's policy
info. message	untrusted	administrator's policy
error message	trusted	check packet, process if okay
error message	untrusted	administrator's policy

TABLE I: Recommended ICMP processing rules in IPsec.

We now review two measures that help improving the security of ICMP PTB packets processing.

a) Minimum sanity check for untrusted ICMP error packets: the processing of ICMP error packets coming from the untrusted network must satisfy the following sanity check ([14], section 2.3). ICMP requires that such an ICMP error packet include in its payload the beginning of the packet that triggered the error. Upon receiving the error packet, the IPsec protocol must verify that the outer header of the packet that triggered the error (i.e., contained in the ICMP payload) maps to a valid entry in the SAD, by checking the source/destination IP addresses and SPI. If not, the ICMP packet must be immediately discarded.

b) Additional sanity checks: in addition to the minimum sanity check, some IPsec implementations (including the one we considered, see section IV) decrypt the ICMP packet payload, recover the inner IP packet header and verify that the source/destination IP addresses of the inner packet match the SAD entry associated to the SPI. If the check fails, the packet is immediately dropped. This is an easy solution to avoid blind attacks, coming from attackers that are not able to eavesdrop an active tunnel. However it offers no protection if the attacker is on the path followed by the IPsec tunnel (a feature our PTB-PTS attack relies on).

[1] also recommends to "establish a minimum PMTU for the traffic (on a per destination basis), to prevent receipt of an unauthenticated ICMP from setting the PMTU to a trivial size". We will see in our attack that this is not necessarily sufficient.

IV. Attack description based on a real exploit

Our attack is designed to take place both in site-to-site or host-to-site configurations (Fig 1). It is carried out from the untrusted network, and through the IPsec gateway, the attack targets hosts in the trusted network, behind the gateway (i.e., in host-to-site configuration, the site is the target, not the isolated host). We assume the attacker can eavesdrop and inject traffic on the untrusted network, as described in section II-B. However a single ICMP packet is sufficient for the attack, which means it can easily remain unnoticed.

A. Experimental conditions

We illustrate the attack through an exploit, using two on-the-shelf IPsec gateways with their default configuration[3]. The gateways as well as the end machines are all running the stable "Squeeze" Debian distribution [15], with Linux kernel 3.2.1 [16]. However this attack is not specific to this distribution. We exhibit the impacts of the attack on a user, in a trusted red network, that tries to establish an ssh connection with a machine located on the remote trusted red network, through the IPsec tunnel, using IPv4[4].

In the next section, we assume that hosts rely on the classic PMTUd algorithm (the default) and show that it **leads to a DoS** since the attacker can easily prevent any new ssh connection from being established.

Then, in section IV-C, we consider the case where hosts rely on the PLPMTUd alternative and show that the attacker can **slow down** the ssh connection (6+ seconds of connection delay) as well as **limiting the TCP segment size** to a tiny value much lower than the minimum MTU size of IPv4 which negatively impacts the throughput[5].

Finally we also tested with a bulk UDP flow. Here also, the attack leads to a major slow down of the connection since the gateway needs to further segment IP datagrams.

B. DoS on TCP connections with hosts using PMTUd

Let us assume that end-hosts use PMTUd. The attack is illustrated in Fig. 2 and the corresponding tcpdump traces, collected on the red network, are shown in Fig. 3. Note that the traces show the two TCP flows (connections are bidirectional), whereas Fig. 2 is simplified and only shows the flow being attacked. In particular the ssh connection establishment involves the exchange of 784 bytes in one direction (which hit the gateway PMTU entry) and 848 bytes in the other direction (this segment is not subject to PMTU restrictions).

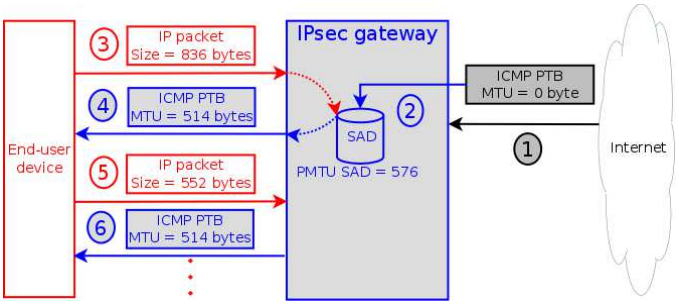

Fig. 2: Our attack on an IPsec gateway, PMTUd case

1) Forging an ICMP PTB packet from the untrusted network: the attacker first has to forge an appropriate ICMP PTB packet (a single packet is sufficient). This is done by eavesdropping a valid packet from the IPsec tunnel on the untrusted network. Then the attacker forges an ICMP PTB packet (step 1 in Fig. 2), specifying a very small MTU value equal or smaller than 576 with IPv4 (resp. 1280 with IPv6). The attacker uses 0 in this case. This packet spoofs the IP address of a router of the untrusted network (in case the

[3]Since we assume that most administrators do not change the default IPsec policies with regard to ICMP processing, we did not change them.

[4]In this configuration, the attacker targets the gateway of the ssh server. We also tested the symmetric configuration (gateway of the ssh client). Since the results are exactly the same, they are not shown.

[5]We show an exploit with an interactive ssh connection for which throughput is not an issue. But the attack consequences will be more serious with an application doing bulk data transfer on top of TCP or HTTP.

```
0.000000 a.b.10.7.48058 > a.b.11.5.ssh:   S *:*(0) win 17920 <mss 8960,sackOK,timestamp 1245892 0,nop,wscale 7> (DF)
0.000146 a.b.11.5.ssh   > a.b.10.7.48058: S *:*(0) ack * win 17896 <mss 8960,sackOK,timestamp 1319280 1245892,nop,wscale 7> (DF)
0.000304 a.b.10.7.48058 > a.b.11.5.ssh: . ack 1 win 140 <nop,nop,timestamp 1245892 1319280> (DF)
0.004561 a.b.11.5.ssh   > a.b.10.7.48058: P 1:33(32) ack 1 win 140 <nop,nop,timestamp 1319281 1245892> (DF)
0.004698 a.b.10.7.48058 > a.b.11.5.ssh:   . ack 33 win 140 <nop,nop,timestamp 1245893 1319281> (DF)
0.004773 a.b.10.7.48058 > a.b.11.5.ssh:   P 1:33(32) ack 33 win 140 <nop,nop,timestamp 1245893 1319281> (DF)
0.004858 a.b.11.5.ssh   > a.b.10.7.48058: . ack 33 win 140 <nop,nop,timestamp 1319281 1245893> (DF)
0.004933 a.b.11.5.ssh   > a.b.10.7.48058: P 33:817(784) ack 33 win 140 <nop,nop,timestamp 1319281 1245893> (DF)
0.004953 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.004998 a.b.10.7.48058 > a.b.11.5.ssh:   P 33:881(848) ack 33 win 140 <nop,nop,timestamp 1245893 1319281> (DF)
0.005084 a.b.11.5.ssh   > a.b.10.7.48058: . 33:533(500) ack 33 win 140 <nop,nop,timestamp 1319281 1245893> (DF)
0.005092 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.005095 a.b.11.5.ssh   > a.b.10.7.48058: P 533:817(284) ack 33 win 140 <nop,nop,timestamp 1319281 1245893> (DF)
0.005228 a.b.10.7.48058 > a.b.11.5.ssh:   . ack 33 win 140 <nop,nop,timestamp 1245893 1319281,nop,nop,sack 1 {533:817} > (DF)
0.043580 a.b.11.5.ssh   > a.b.10.7.48058: . ack 881 win 154 <nop,nop,timestamp 1319291 1245893> (DF)
0.215586 a.b.11.5.ssh   > a.b.10.7.48058: . 33:533(500) ack 881 win 154 <nop,nop,timestamp 1319334 1245893> (DF)
0.215594 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.639580 a.b.11.5.ssh   > a.b.10.7.48058: . 33:533(500) ack 881 win 154 <nop,nop,timestamp 1319440 1245893> (DF)
0.639586 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
```

Fig. 3: tcpdump trace on the red network during the attack, PMTUd case. Here the remote client machine with IP address a.b.10.7 tries to ssh to the local machine with IP address a.b.11.5, located behind the IPsec gateway with IP address a.b.11.4. (NB: non required information has been removed from these traces)

source IP address is checked), and in order to bypass the IPsec protection mechanism against blind attacks, it includes as a payload a part of the outer IP packet that has just been eavesdropped.

This is the only packet an attacker needs to send. The following steps do not involve any action from the attacker.

2) Reset of the PMTU on the gateway: this ICMP packet is processed by the IPsec gateway. As the packet appears to belong to an active tunnel, the gateway stores the following PMTU value in its SAD (step 2):

$$\text{PMTU}_{\text{SAD}} = max(\text{MTU}_{\text{ICMP PTB}}, 576) = 576$$

It is important to note that the gateway does not store a proposed value smaller than the minimum guaranteed MTU (the attacker proposed 0 but 576 has been stored).

At this point, the traffic is not blocked in any way between the targeted gateway and the remote end of the tunnel. Nevertheless the throughput is reduced on the IPsec tunnel as any packet exceeding the PMTU_{SAD} size must be fragmented (usually by the end-host as the DF bit is set).

3) Drop of the first large TCP segment: let us consider an ssh connection from outside, to a server located in the red network. The TCP three-way handshake happens normally because these TCP segments are tiny. However any further bulk data transfer on this connection is impacted. This is the case of the $784 + 52 = 836$ byte packet (52 bytes for the TCP/IP headers, including TCP options) of step 3, which exceeds the PMTU_{SAD} value stored in the SAD.

4) ICMP PTB error packet on the trusted network: therefore the IPsec gateway emits an ICMP PTB packet (step 4) with the following MTU indication:

$$\text{MTU} = \text{PMTU}_{\text{SAD}} - size_{encapsulation\ IP/IPsec/ESP}$$

Due to the encapsulation header (whose size depends on the chosen ciphering algorithm), the gateway restricts the MTU value to 514 bytes. Looking at Fig. 3, we see that the 8th packet (784 byte TCP segment) is immediately followed by an ICMP error packet with that MTU information.

5) Deadlock on the red network: upon receiving this ICMP PTB packet, the host computes the PMTU to use[6]:

$$\text{PMTU} = max(\text{MTU}_{\text{ICMP PTB}}, \text{MTU}_{\text{config.}}) = 552$$

Therefore the TCP segments are fragmented (remember that the host sets the DF bit to be sure that no fragmentation appears later on in the network for performance reasons). Nevertheless, instead of creating 514 byte packets, as requested by the gateway, the host generates $500 + 52 = 552$ byte packets (step 5). Since it remains too large, they are dropped by the gateway and this latter replies with ICMP PTB packets. After 2 minutes of failures, the ssh server initiates a half-close (FIN/ACK exchange) (but the other side of the TCP connection curiously remains open).

To conclude, the TCP connection is completely blocked and the DoS is successful.

C. Attack on TCP connections with hosts using PLPMTUd

As the DoS attack relies on a maximum packet size feature, we also experiment with the second path MTU discovery algorithm, PLPMTUd. The attack is illustrated in Fig. 4 and the corresponding tcpdump traces, collected on the red network, are shown in Fig. 5. We show below that the attack does have severe consequences even if it does not result on a DoS: (1) it considerably slows down the ssh connection opening and (2) it limits the TCP segment size to a tiny value which generates more overhead and reduces the maximum throughput.

The first two steps involving the attacker, identical to the PMTUd case, are omitted.

1) Fragments handling by the gateway: due to the PLPMTUd algorithm that progressively increases the segment size, the host now fragments the 784 byte ssh message into two TCP segments of size 500 and 284 bytes respectively (steps 3 and 3' of Fig. 4). The 500 byte size is typically a probing size, chosen by PLPMTUd, in order to test this small value. The

[6]The 552 value comes from the default Debian configuration, but a recent Linux (here FedoraCore 20) uses the same value: cat /proc/sys/net/ipv4/route/min_pmtu returns 552. This value can be changed by the administrator.

```
0.000000 a.b.10.7.48063 > a.b.11.5.ssh:   S *:*(0) win 17920 <mss 8960,sackOK,timestamp 1572549 0,nop,wscale 7> (DF)
0.000142 a.b.11.5.ssh   > a.b.10.7.48063: S *:*(0) ack * win 17896 <mss 8960,sackOK,timestamp 1645937 1572549,nop,wscale 7> (DF)
0.000417 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 1 win 140 <nop,nop,timestamp 1572550 1645937> (DF)
0.004208 a.b.11.5.ssh   > a.b.10.7.48063: P 1:33(32) ack 1 win 140 <nop,nop,timestamp 1645938 1572550> (DF)
0.004535 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 33 win 140 <nop,nop,timestamp 1572551 1645938> (DF)
0.004538 a.b.10.7.48063 > a.b.11.5.ssh:   P 1:33(32) ack 33 win 140 <nop,nop,timestamp 1572551 1645938> (DF)
0.004676 a.b.11.5.ssh   > a.b.10.7.48063: . ack 33 win 140 <nop,nop,timestamp 1645938 1572551> (DF)
0.004688 a.b.10.7.48063 > a.b.11.5.ssh:   . 33:545(512) ack 33 win 140 <nop,nop,timestamp 1572551 1645938> (DF)
0.004711 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 33 win 140 <nop,nop,timestamp 1645938 1572551> (DF)
0.004719 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.004721 a.b.11.5.ssh   > a.b.10.7.48063: P 533:817(284) ack 33 win 140 <nop,nop,timestamp 1645938 1572551> (DF)
0.004960 a.b.10.7.48063 > a.b.11.5.ssh:   P 545:881(336) ack 33 win 140 <nop,nop,timestamp 1572551 1645938> (DF)
0.005006 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 33 win 140 <nop,nop,timestamp 1572551 1645938,nop,nop,sack 1 {533:817} > (DF)
0.005046 a.b.11.5.ssh   > a.b.10.7.48063: . ack 881 win 156 <nop,nop,timestamp 1645938 1572551> (DF)
0.214634 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 881 win 156 <nop,nop,timestamp 1645991 1572551> (DF)
0.214643 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.638636 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 881 win 156 <nop,nop,timestamp 1646097 1572551> (DF)
0.638646 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
1.486639 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 881 win 156 <nop,nop,timestamp 1646309 1572551> (DF)
1.486645 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
3.186646 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 881 win 156 <nop,nop,timestamp 1646734 1572551> (DF)
3.186655 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
6.586634 a.b.11.5.ssh   > a.b.10.7.48063: . 33:289(256) ack 881 win 156 <nop,nop,timestamp 1647584 1572551> (DF)
6.586831 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 289 win 148 <nop,nop,timestamp 1574196 1647584,nop,nop,sack 1 {533:817} > (DF)
6.586941 a.b.11.5.ssh   > a.b.10.7.48063: . 289:533(244) ack 881 win 156 <nop,nop,timestamp 1647584 1574196> (DF)
6.587143 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 817 win 156 <nop,nop,timestamp 1574196 1647584> (DF)
6.587147 a.b.10.7.48063 > a.b.11.5.ssh:   P 881:905(24) ack 817 win 156 <nop,nop,timestamp 1574196 1647584> (DF)
6.588458 a.b.11.5.ssh   > a.b.10.7.48063: P 817:969(152) ack 905 win 156 <nop,nop,timestamp 1647584 1574196> (DF)
6.589189 a.b.10.7.48063 > a.b.11.5.ssh:   P 905:1049(144) ack 969 win 164 <nop,nop,timestamp 1574197 1647584> (DF)
6.593662 a.b.11.5.ssh   > a.b.10.7.48063: . 969:1225(256) ack 1049 win 164 <nop,nop,timestamp 1647585 1574197> (DF)
6.593739 a.b.11.5.ssh   > a.b.10.7.48063: . 1225:1481(256) ack 1049 win 164 <nop,nop,timestamp 1647585 1574197> (DF)
6.593750 a.b.11.5.ssh   > a.b.10.7.48063: P 1481:1689(208) ack 1049 win 164 <nop,nop,timestamp 1647585 1574197> (DF)
6.593946 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 1481 win 176 <nop,nop,timestamp 1574198 1647585> (DF)
```

Fig. 5: tcpdump trace on the red network during the attack, PLPMTUd case. Notations are consistent with Fig. 3

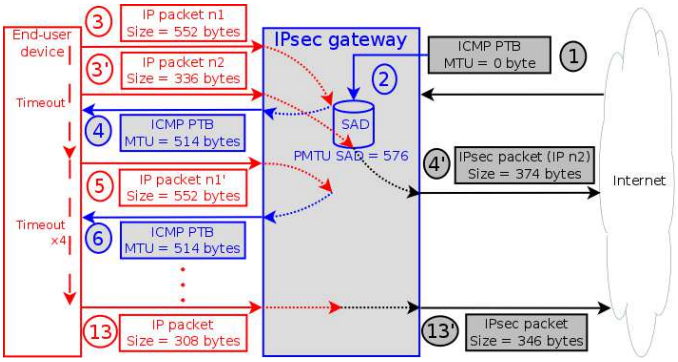

Fig. 4: Our attack on an IPsec gateway, PLPMTUd case

gateway processes each packet, returning an ICMP PTB packet for the first one (step 4) as it is too large, and forwarding the second one (step 4').

2) Large segment and ICMP PTB: the ICMP PTB packet is ignored as the PLPMTUd component only relies on acknowledgments and delay expirations. In our test, after expiration of the timeout for the first packet (at time 0.21s), the host sends an identical 552 byte packet to the gateway (step 5) because PLPMTUd already used the minimal size allowed by the host configuration. This pattern happens 5 times, generating a total of 5 ICMP PTB packets.

3) Further reduction of the segment size: 6.59s after the TCP connection establishment, the PLPMTUd component decides to drastically reduce the segment size: instead of a single 500 byte segment, it now sends a 256 byte TCP segment (step 13) followed by a 244 byte TCP segment. Being small enough, both of them are forwarded by the IPsec gateway. The ssh connection finishes after a few additional segments and a prompt appears in the terminal.

To conclude a huge delay of 6.59s was required for data to arrive to the destination. Additionally, any packet leaving the host after this initial delay contains at most 256 byte of data, which drastically reduces the TCP throughput and consumes more resources in the forwarding nodes[7].

D. Attack on a bulk UDP flow

Let us now consider a UDP flow, where the application submits 1,100 byte data chunks to the UDP socket. The beginning of the attack is the same. Then the host sends a $1100 + 28 = 1128$ byte IP packet with the DF bit set to 1 (no fragmentation). The IPsec gateway discards this packet and returns an ICMP PTB packet with the same 514 byte MTU indication as before. The following UDP datagram is fragmented into three IP packets of size 548, 548 and 72 bytes respectively. This time the DF bit is set to 0 in all three packets (fragmentation is authorized), probably to reduce the risks that these packets be dropped. At the gateway, it turns out that after encapsulation, the first two IP packets are again too large compared to the PMTU value of the SAD. Since fragmentation is authorized, they are once again fragmented into two packets each, of size 528 and 60 bytes respectively. At the end, the initial large UDP datagram is transmitted in the IPsec tunnel in five medium size or tiny IP packets (548, 60, 548, 60 and 112 bytes respectively), instead of a single packet (without the attack).

[7]Router performance (number of packets per second) is relatively independent of the packet size, but the lower the size, the lower the throughput.

To conclude, our attack has different severe effects, depending on the transport protocol and PMTUd algorithm used, ranging from throughput reduction to DoS.

V. Back to the fundamental problem highlighted by the attack

The comprehensive analysis of our attack in section IV highlights two fundamental issues that we successively discuss, before considering potential counter-measures.

A. Issue 1: determining the legitimacy of untrusted ICMP packets

The first problem is the impossibility, for an IPsec gateway, to determine whether an ICMP PTB packet, coming from the black untrusted network, is legitimate (i.e., has been triggered by a valid transmission error) or not (i.e., has been forged by an attacker on the path to the destination). The two security measures discussed in section III-C (i.e., outer header verification and payload verification) are essential to avoid blind attacks, but not sufficient if the attacker is on the path followed by the IPsec tunnel as we have shown (in particular in presence of an unsecured WiFi network, see Section II-B). This is a fundamental limit given the current IPsec specifications. At the same time, filtering out all ICMP packets is not a solution as many hosts still rely on the traditional PMTUd algorithm. Another approach is required.

B. Issue 2: dealing with minimum Path MTU in presence of a tunnel

When the Path MTU advertised to the IPsec gateway approaches the minimum MTU each link technology should support (i.e., 576 bytes with IPv4, 1280 with IPv6), problems can arise as IPsec tunnelling adds the IP/IPsec/ESP headers.

There are two sides to the problem. First of all, we observe that the end-host does not accept the Path MTU advertised by the IPsec gateway if it is smaller than the minimum MTU configured locally. Indeed, the local component that takes this decision is not aware that the gateway operates an IPsec tunnel and needs some additional room. With the PMTUd approach, the compliance on the minimum MTU is strict and a DoS results. With the PLPMTUd approach, the end-host pragmatically uses (after some time) a TCP segment size significantly lower than this minimum MTU, no matter the local PMTU information. Communications are therefore feasible, although in a sub-optimal way.

The second side of the problem is that the IPsec gateway should not accept from the black network an ICMP PTB asking to reduce the MTU to 576 bytes, since the gateway will not be able to offer a minimum MTU at least equal to 576 bytes to the hosts on the red network. However there is a fundamental contradiction here since 576 bytes is a valid MTU value for a link. This is typically a situation where an alarm should be sent to the IPsec gateway administrator for this later to analyze the situation and decide what to do. The current situation where this contradiction is silently ignored is definitively not appropriate.

C. A (trivial) counter-measure: ignoring the DF bit

Several trivial counter-measures exist. One of them consists in configuring the IPsec gateway in such a way packets are fragmented regardless of the original DF bit setting. In our three scenarios, all the data packets sent by the end-host would have been fragmented silently as soon as they exceed the local 514 byte threshold. This is feasible (and recommended) with Cisco IOS 12.2(11)T and above ([17], "DF Bit Override Functionality with IPsec Tunnels" section). The attack is mitigated, but as mentioned in this reference, "a significant performance impact occurs at a high data rate". In some sense, the attack has succeeded since it may remain unnoticed for a long period while seriously impacting the VPN performance.

D. Another counter-measure: confirming the ICMP PTB information with a gateway probing mechanism

An approach to the problem is the following: since the legitimacy of an untrusted ICMP packet cannot be determined, the idea is to confirm an ICMP PTB information with a side mechanism. Therefore we propose that the IPsec gateway uses a PLPMTUd–like probing mechanism. It works as follows. The gateway generates a probing packet of a certain size and sends it on a given path. In order to determine if this size is compatible with this path, the remote IPsec gateway is supposed to acknowledge it upon reception, otherwise a timeout is supposed to be triggered at the gateway. If the PLPMTUd probing does not confirm the ICMP PTB information, then this ICMP packet is declared erroneous and should be ignored. For this mechanism to be effective, the attacker should not be able to identify in real time the probing packets in the tunnel in order to discard them selectively and let the gateway think the probe is actually too large[8].

In some situations, this gateway level probing, done periodically, could totally replace the ICMP mechanism. However ICMP (if ICMP packets are not already filtered out by Internet routers) may be more reactive than a periodic probing, and we believe that both mechanisms can safely work in parallel.

VI. Related work

IPsec has attracted the attention of the security and cryptography communities. Some works describe attacks related to the misuse of cryptographic primitives in IPsec. The authors of [18], [19] have discovered several weaknesses related to an incorrect usage of encryption. [20], [21] focus on DoS attacks against the "encryption only" configuration of IPsec. These works are based on the possibility of a forgery attack on the Initialization Value to change an inner header field. Our work follows a purely network approach to uncover weaknesses.

ICMP has been used for DoS and distributed DoS attacks for a long time (ping flood is detailed in every network security textbooks [22]). DoS attacks such as Smurf attack [23] and Tribe Flood Network attack [24] are respectively based on ICMP "Echo Request" using a spoofed IP source (the victim) plus a broadcast IP destination, and ICMP "Echo Reply" plus a botnet. Both attacks aim to flood the target with

[8]Note that these capabilities go much farther than what is permitted in our attacker model (section II-B).

ICMP packets. Many implementations of ICMP do not handle correctly packets larger than the maximal value recommended by the RFCs. These oversized ICMP packets, well–known as "Ping of Death", can disable a host. Otherwise flooding the victim with "Echo Request/Reply" packets has long been considered. But this is also easily avoided with appropriate filtering rules in the IPsec gateway. A more advanced attack detection mechanism [25] relies on the response of monitored systems to probes but does not propose any algorithm to handle the ICMP PTB packets attacks. We see that our work totally departs from these "traditional" attacks and uses techniques that bypass the counter measures.

The authors of [3], [26] acknowledge the existence of threats related to the use of ICMP PTB packets (e.g. for blind throughput–reduction attack against TCP [14]). The authors of [27] detail for the various ICMP packets the associated threats and provide advice to mitigate them. Our work goes more deeply in this direction and introduces a new threat associated to the ICMP PTB packets.

VII. Conclusion

In this work we show that the ICMP PTB (Packet Too Big) messages are an efficient attack vector against IPsec gateways, as soon as an attacker has eavesdropping and traffic injection capabilities in the "black" untrusted network, which we believe are realistic assumptions (e.g. in an insecure WiFi network). We demonstrate the PTB-PTS attack through an exploit in a Debian–based testbed. We show the attack results in either a DoS or major performance penalties, for both TCP and UDP flows.

The PTB-PTS attack highlights two fundamental problems: the impossibility to determine the legitimacy of untrusted ICMP packets, and issues in dealing with minimum Path MTU in presence of a tunnel. We therefore suggest a counter measure at the IPsec gateways that consists in confirming the information carried by ICMP PTB packets with an active probing approach.

Future works will enlarge the scope of this study, by considering other Operating Systems as well as tests with IPv6. We will also experiment with the gateway-level counter measure that we propose.

References

[1] S. Kent and K. Seo, "Security architecture for the internet protocol," IETF Request For Comments RFC 4301, http://datatracker.ietf.org/doc/rfc4301/, December 2005.

[2] S. Kent, "IP encapsulating security payload (ESP)," IETF Request For Comments RFC 4303, http://datatracker.ietf.org/doc/rfc4303/, December 2005.

[3] J. Mogul and S. Deering, "Path mtu discovery," IETF Request For Comments 1191, http://datatracker.ietf.org/doc/rfc1191/, November 1990.

[4] M. Luckie, K. Cho, and B. Owens, "Inferring and debugging path MTU discovery failures," in *5th ACM Internet Measurement Conference*, ser. IMC'05, October 2005.

[5] M. Luckie and B. Stasiewicz, "Measuring path MTU discovery behaviour," in *10th ACM Internet Measurement Conference*, ser. IMC'10, November 2010.

[6] J. Postel, "Internet control message protocol," IETF Request For Comments 792, http://datatracker.ietf.org/doc/rfc792/, September 1981.

[7] S. Kent, "IP authentication header," IETF Request For Comments RFC 4302, http://datatracker.ietf.org/doc/rfc4302/, December 2005.

[8] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "Internet key exchange protocol version 2 (IKEv2)," IETF Request For Comments RFC 5996, http://datatracker.ietf.org/doc/rfc5996/, September 2010.

[9] N. Egi, M. Dobrescu, J. Du, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, L. Mathy, and S. Ratnasamy, "Understanding the packet Processing Capabilities of Multi-core Servers," Technical Report LABOS-REPORT-2009-001, EPFL, Switzerland, February 2009.

[10] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: exploiting parallelism to scale software routers," in *22nd ACM Symposium on Operating Systems principles (SOSP'09)*, October 2009.

[11] "Myricom 10-Gigabit Ethernet Performance Measurements," http://www.myricom.com/scs/performance/Myri10GE/.

[12] L. Jacquin, V. Roca, M. A. Kaafar, F. Schuler, and J.-L. Roch, "IBTrack: An ICMP Black holes Tracker," Dec. 2012. [Online]. Available: http://hal.inria.fr/hal-00695746

[13] M. Mathis and J. W. Heffner, "Packetization layer path mtu discovery," IETF Request For Comments RFC 4821, http://datatracker.ietf.org/doc/rfc4821/, March 2007.

[14] F. Gont, "ICMP attacks against TCP," IETF Request For Comments RFC 5927, http://datatracker.ietf.org/doc/rfc5927/, July 2010.

[15] "Debian – The Universal Operating System," http://www.debian.org.

[16] "The Linux kernel," http://www.kernel.org.

[17] ., *IPsec Data Plane Configuration Guide, Cisco IOS Release 15M&T*. Cisco Systems, Inc., http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_conn_dplane/configuration/15-mt/sec-ipsec-data-plane-15-mt-book.pdf, 2012.

[18] J. P. Degabriele and K. G. Paterson, "Attacking the IPsec Standards in Encryption-only Configurations," in *IEEE Symposium on Security and Privacy*, May 2007.

[19] J.-P. Degabriele and K. G. Paterson, "On the (in)security of IPsec in MAC-then-encrypt configurations," in *17th ACM Conference on Computer and Communications Security (CCS'10)*, October 2010.

[20] C. B. McCubbin, A. A. Selçuk, and D. P. Sidhu, "Initialization Vector Attacks on the IPsec Protocol Suite," in *9th IEEE Int. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'00)*, 2000.

[21] V. Nikov, "A DoS Attack Against the Integrity-Less ESP (IPSec)," *IACR Cryptology ePrint Archive, http://eprint.iacr.org/2006/370.pdf*, 2006.

[22] S. Northcutt and J. Novak, *Network Intrusion Detection, Third Edition*. New Riders Publishing, September 2002.

[23] "Smurf IP Denial–of–Service Attacks," http://www.cert.org/advisories/CA-1998-01.html, January 1998.

[24] "Similar Attacks Using Various RPC Services," http://www.cert.org/incident_notes/IN-99-04.html, July 1999.

[25] F. A. Barbhuiya, S. Roopa, R. Ratti, S. Biswas, and S. Nandi, "An Active Detection Mechanism for Detecting ICMP Based Attacks," in *11th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom'12)*, June 2012.

[26] J. McCann, S. Deering, and J. Mogul, "Path mtu discovery for ip version 6," IETF Request For Comments 1981, http://datatracker.ietf.org/doc/rfc1981/, August 1996.

[27] F. Gont, G. Gont, and C. Pignataro, "Recommendations for filtering ICMP messages," IETF Working Document, http://datatracker.ietf.org/doc/draft-ietf-opsec-icmp-filtering/, July 2013.