ICMP: an Attack Vector against IPsec Gateways

Ludovic Jacquin, Vincent Roca, Jean-Louis Roch

ICMP: an Attack Vector against IPsec Gateways

Ludovic Jacquin
Inria, France
ludovic.jacquin@inria.fr

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

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

***Abstract*—In this work we show that the Internet Control Message Protocol (ICMP) can be used as an attack vector against IPsec gateways. The main contribution of this work is to demonstrate that an attacker having eavesdropping and traffic injection capabilities in the black untrusted network (he only sees ciphered packets), can force a gateway to reduce the Path MTU of an IPsec tunnel to a minimum, which in turn creates serious issues for devices on the trusted network behind this gateway: depending on the Path MTU discovery algorithm, it either prevents any new TCP connection (Denial of Service), or it creates major performance penalties (more than 6 seconds of delay in TCP connection establishment and ridiculously small TCP segment sizes). After detailing the attack and the behavior of the various nodes, we discuss some counter measures, with the goal to find a balance between ICMP benefits and the associated risks.**

I. Introduction

IPsec/ESP [1][2] offer a convenient secure tunnelling capability that is largely used to interconnect remote sites, or a remote host to its home network, throughout an unsecured interconnection network (e.g. Internet). The various hosts can then exchange confidential information securely, even in presence of a powerful attacker on the Internet.

IPsec naturally has to interact with every protocol of the IP suite, and in particular the Internet Control Message Protocol (ICMP). The 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 therefore essential for performance aspects: if a packet is too large, its fragmentation and reassembly will negatively impact performance; at the other extreme, if a packet is too small, significantly lower than the maximum size permitted throughout the path, it will also negatively impact performance. Assessing the correct packet size on a network path is therefore a key aspect. But ICMP is also known to be a cause of attacks, and therefore there is an incentive for a network administrator to filter these packets. A balance is therefore required between these contradictory objectives and it is recognized that a subset of ICMP packets should be considered by IPsec gateways.

The problem this work addresses is the following: how can an attacker located in the unsecured interconnection network exploit the combination of IPsec and ICMP to mount Denial of Service (DoS) attacks? Note that we do not consider traditional trivial ICMP attacks (see section VI) in this work. Our contributions are the following:

- we demonstrate, through a real exploit on a testbed running a recent Debian distribution, that an external attacker having eavesdropping and traffic injection capabilities in the black untrusted network, without any access to clear-text (he only sees ciphered packets), can either stall TCP connections going through the IPsec tunnel, or create major performance penalties;
- we explain how IPsec and ICMP interact with one another, in presence of either the PMTUd or the PLPMTUd algorithms;
- we provide some ideas on how to thwart this attack, in particular a novel way of using PLPMTUd within IPsec gateways;

The remaining of the paper is organized as follows. In Section II we introduce the network and the attacker model. IPsec and ICMP are described in Section III. Section IV describes in detail our attack. Section V presents the counter-measures we propose. We position our paper and present related work in Section VI. Finally we conclude.

II. Network and attacker

A. The network model

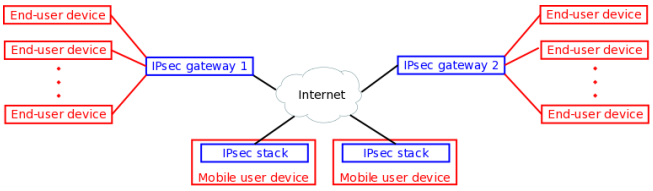

Fig. 1: Network model showing the site-to-site (top) and device-to-site (bottom) configurations.

Our network model identifies two zones: the trusted areas, consisting of networks and devices behind IPsec gateways, also called *"red networks"*, and the outer world considered as untrusted, and also called *"black network"*. Trusted areas can be interconnected by IPsec gateways through IPsec tunnels (site-to-site configuration of Fig. 1). Moreover, an isolated device can establish an IPsec tunnel if it includes the IPsec gateway functionalities (device-to-site configuration of Fig. 1).

The role of the IPsec gateway is to secure the traffic between remote trusted networks and/or devices by encrypting packets sent over the untrusted black network.

Being part of the trusted areas, the IPsec gateways are not considered malicious. Nevertheless, these gateways are directly connected to the untrusted network, making them the first line of defense against untrusted traffic.

B. The attacker model

Because of the network model, we assume that no attacker is located inside a trusted red network, all the attacks are conducted by adversaries from the the external black network. Therefore **an attacker only sees encrypted traffic and has no way to decrypt packets**. We also assume the attacker can **eavesdrop** the traffic and **inject** forged packets. However the attacker cannot decrypt a ciphered packet nor encrypt its own packet since the underlying cryptographic building blocks and the key exchange protocols are considered secured. Such an attacker can be a compromised router along the path followed by an IPsec tunnel, in the black network, or a station attached to a non-secured Wi-Fi network, when users of this Wi-Fi network connect to their home network through a VPN.

The goal of the attacker is to launch a DoS against the secure tunnel service provided by IPsec gateways. For instance he wants to significantly reduce the througput achieved by IPsec tunnels, and if possible, to prevent connections between devices located in different red networks.

III. IPsec and ICMP in a nutshell

In this section we give some background on IPsec [1], ICMP [7], 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 [8] and ESP [2], and two modes of operation, transport and tunnel. In our work we consider the traditional solution: IPsec/ESP in tunnel mode. In that case IPsec/ESP provide confidentiality, authentication, integrity and anti-replay services. More precisely, the initial IP header and data of an incoming packet (called inner IP packet) arriving at the IPsec tunnel endpoint are ciphered by ESP and then 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 (that are initialized by the IKEv2 [9] protocol), or 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.

B. ICMP overview

ICMP handles both error and informational messages [10]. Error messages can be any of the following: destination unreachable, source quench, redirect, time exceeded and parameter problem. Informational messages are other control signalizations: echo request/reply, router solicitation/advertisement, timestamp request/reply, information request/reply and address mask request/reply.

In particular, ICMP is heavily used in the Path MTU discovery (PMTUd) algorithm [3]. PMTUd is a key mechanism for the network performance since it enables a sender to determine the appropriate packet size, dynamically. Since many packet processing overheads remain the same regardless of the packet size, reducing their number is critical from a performance point of view, and this is what a well-chosen PMTU enables [11], [12], [13].

1) The legacy PMTUd mechanism: Let us illustrate the behavior of PMTUd in an IPv4 (resp. IPv6) network. A sender sets the *Don't Fragment* (DF) bit in a packet[1]. 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 messages ICMP PTB (Packet Too Big), independently of whether IPv4 or IPv6 is used.

Iteratively, upon receiving such ICMP PTB packet, the sender decreases the packet size until it reaches the lowest MTU on the path to the destination. Since this path can change dynamically (because of re-routing), this process needs to be performed periodically.

Although efficient, the PMTUd approach suffers from several limits, mainly because ICMP packets are often filtered by some routers/firewalls [14] along their route to the sender. In that case the sender has no way to discover and resolve the problem.

2) The Packetization Layer PMTUd mechanism: To overcome these issues, the IETF developed a new Path MTU discover mechanism that does not rely on ICMP, the Packetization Layer PMTUd (PLPMTUd) [15]. Instead of using ICMP, it relies on a packetization layer protocol with an acknowledgement mechanism, such as TCP. Using this protocol, PLPMTUd sends probing packets of an appropriate size to the destination in order to assess the MTU along the path. Upon receiving an acknowledgement for a probing packet, the sender validates that the PMTU is at least equal to the probing packet size, while a time-out is used to infer that the PMTU is smaller than the size probed. With TCP, any TCP data segment can be used as a probing packet if enough data is available to fill in the payload. Here also 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,

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

how to handle some of them. More precisely, a distinction is made between error and informational ICMP packets, and the network area they come from ([1] Sections 5 and 8). The recommended treatments are summarized in TABLE I. The key row of this table for our attack, is the last one that corresponds to untrusted ICMP error messages. If the policy is left open, there are strong incentives to consider the particular case of ICMP PTB error messages in order to enable PMTU discovery [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.

a) Minimum sanity check for untrusted ICMP error messages: when an administrator allows the processing of ICMP error messages coming from the untrusted network and triggered by a packet sent in an active IPsec tunnel (e.g. with the PMTUd mechanism), the following sanity check is performed ([16], section 2.3). ICMP specifies that a router generating such an error message must include in the ICMP packet payload the beginning of the (ciphered) packet that triggered the error. Upon receiving the error message, the IPsec protocol must verify that the outer header of the triggering packet (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 message is immediately dropped.

This is an easy solution to avoid blind attacks, coming from attackers that are not able to eavesdrop an active tunnel, but of course it offers no protection if the attacker is on the path followed by the IPsec tunnel.

[1] also recommends to "establish a minimum PMTU for 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. Using ICMP as attack vector against IPsec

A. Experimental conditions

Our attack is designed to take place in site-to-site or device-to-site configurations that involve at least one IPsec gateway (Fig 1). The attack is carried out from the untrusted network, and through the IPsec gateway, targets devices in the trusted network, behind the gateway. The attacker model is the one described in section II-B, i.e. the attacker can eavesdrop and inject traffic on the untrusted network.

We illustrate the attack by considering an on-the-shelf IPsec gateway with its default configuration[2]. The gateways as well as the end machines are all running the stable "Squeeze" Debian distribution [17], with Linux kernel 3.2.1 [18]. However this attack is not specific to this distribution. We exhibit the impact of the attack on a user, in a trusted red network, that tries to establish an ssh connection with a machine located on a remote trusted red network, through the IPsec tunnel, using IPv4[3].

In the following attack description, we first assume that devices rely on the classic PMTUd algorithm (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 devices rely on the PLPMTUd alternative and show that the attacker can **slow down** the ssh connection (6+ seconds of initial delay) as well as **limiting the TCP segment size** to a tiny value much lower than the minimum MTU size of IPv4.

Finally, we give a quick report on the attack on 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 devices using PMTUd

Let us assume that end-devices 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 flows of TCP (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).

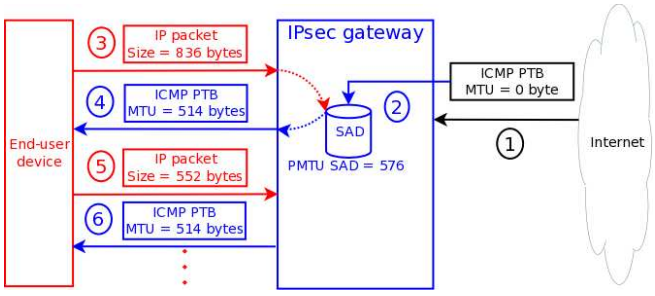

Fig. 2: Our attack on an IPsec gateway, PMTUd case

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

[3]In this configuration, the attacker targets the ssh server's gateway. We also tested with the symmetric configuration, where the attacker targets the ssh client's gateway. Since the results are exactly the same, they are not shown here.

```
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)

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

Note that this is the only packet an attacker needs to generate. 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 accept a proposed value smaller than the minimum guaranteed MTU.

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 for the IPsec tunnel and any packet exceeding the PMTU_{SAD} maximum size must now be fragmented (usually by the end-device 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 the associated TCP segment 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 message 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, containing a 784 byte TCP segment, is immediately followed by an ICMP error message with that MTU indication.

5) Deadlock on the red network: upon receiving this ICMP PTB packet, the device computes the PMTU to use:

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

The 552 value comes from the default Debian configuration (the Linux kernel itself uses 562 instead) and can be changed by the device administrator if needed.

Therefore the TCP segments are fragmented (remember that the device 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 IPsec gateway, the device generates $500 + 52 = 552$ byte packets (step 5). Since it is still too large, this packet is dropped by the gateway which sends another ICMP PTB packet.

At this point, no packet emitted by the device for this TCP connection is forwarded on the IPsec tunnel, and any data (even a few bytes) submitted by the application later gets stuck behind. To conclude, the TCP connection is totally blocked and the DoS attack is successful[4].

C. Attack on TCP connections with devices using PLPMTUd

As the DoS attack exploits a maximum segment size issue, we now experiment with the second path MTU

[4]After 2 minutes of failures, the ssh server initiates a half-close (FIN/ACK exchange). The other side of the TCP connection curiously remains open.

discovery algorithm, PLPMTUd. The attack is illustrated in Fig. 5 and the corresponding `tcpdump` traces, collected on the red network, are shown in Fig. 4.

We show below that the effect of the attack is different: first it significantly slows down the `ssh` connection opening, then it limits the `TCP` segment size to a value significantly lower than the minimum MTU size of `IPv4` which consumes more resources and reduces the maximum throughput.

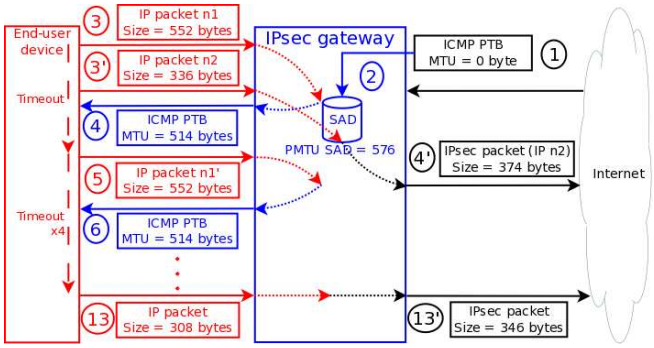

Fig. 5: Our attack on an IPsec gateway, PLPMTUd case

The first two steps that involve the attacker are identical to the PMTUd case and are not discussed here.

1) Fragments handling by the gateway: due to the PLPMTUd algorithm that progressively increases the segment size, the device now fragments the 784 byte `ssh` message into two `TCP` segments of size 500 and 284 bytes respectively (steps 3 and 3' of Fig. 5). The 500 byte size is typically a probing size, chosen by PLPMTUd, in order to test this small value. The 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 deadlock: the `ICMP` PTB packet is ignored by the PLPMTUd algorithm, this latter relying only on acknowledgments and delay expirations. In our test, after expiration of the timeout for the first packet (at time 0.21s), the end-user device sends an identical 552 byte packet to the gateway (step 5) because PLPMTUd already used the minimal size allowed by the host configuration.

The same problem happens 5 times (a total of 5 `ICMP` error messages are received).

3) Finally, the segment size is further reduced: 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 before the first data arrives to the destination. Additionally, any packet leaving this device after this initial delay contains at most 256 byte of data, which can drastically reduce the achieved throughput as well as consuming more resources in the forwarding nodes.

D. Attack on a bulk UDP flow

Let us now consider a UDP flow, where the application submits large data chunks (1100 byte) to the UDP socket. The beginning of the attack is the same. Then the device 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` message 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 with encapsulation, the first two IP packets are again too large (they exceed 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 or tiny IP packets (548, 60, 548, 60 and 112 bytes respectively), whereas three should have been sufficient with an appropriate IP fragmentation at the sender, or even a single one without any attack.

To conclude we see that the attack seriously impacts UDP flows, by reducing the achieved throughput and consuming more resources.

V. Counter-measures

A. Problems assessment

Our attack highlights several fundamental issues:

- an `IPsec` gateway cannot distinguish between legitimate and illegitimate unsecured (i.e. coming from the black network) `ICMP` packets. Although such packets are suspect, they are necessary in particular because many hosts still rely on the old PMTUd algorithm;
- when the Path MTU approaches the minimum packet size each link technology should support (576 bytes with `IPv4`), bad interactions can happen if tunnelling is used. In our case the end-user device does not accept the Path MTU advertised by the `IPsec` gateway because this latter is lower than the minimum packet size, even after removing `TCP/IP` headers. This mistake is also caused by the fact the end-user device is not aware of the presence of an `IPsec` tunnel, that requires some room in any packet to store the extra `IP/IPsec` headers;
- the compliance on the minimum packet size is too strong when using the traditional PMTUd approach. On the opposite, PLPMTUd is more flexible since it finally tries to use a `TCP` segment size lower than this minimum packet size. Communications are feasible, although in a sub-optimal way.

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

In the following sections we first discuss how to mitigate our attack, the we present an IPsec Path MTU evaluation mechanism.

B. Mitigation of the attack at the end-user device

In our testbed, the overhead of the new headers (outer IP, IPsec, ESP) is 38 bytes, so the end-user device must send at most $576 - 38 = 538$ byte long packets for them to go through the tunnel. Knowing that, the end-user can decrease the minimal value used by PMTUd or PLPMTUd algorithms to match the target IP packet size (which has little impact if no attack occurs). TABLE II summarizes the effect of the attack, observed in our experimental testbed, according to this configuration.

Algorithm	Minimum size	Attack effect
PMTUd	538 or less	throughput reduction
PMTUd	539 or more	DoS
PLPMTUd	486 or less	throughput reduction
PLPMTUd	487 or more	severe throughput reduction (256 byte packets)

TABLE II: ICMP PTB attack effects depending of the end–user device configuration.

We see that a correct configuration at the end-user device can mitigate the attack. However it has the main drawback of requiring the end-user to be perfectly aware of the presence of a tunnel and the encapsulation headers sizes that depend on the exact IPsec mode and cipher algorithm used.

C. IPsec gateway Path MTU discovery

Another approach is to let IPsec gateways assess the MTU along the path between them, without relying on ICMP PTB messages that are not reliable.

More precisely we propose to add in the gateway a probing mechanism like PLPMTUd to evaluate the PMTU on the black network. This probing mechanism must be coupled with an acknowledgment and timeout system to determine whether the probing packet found its way to the remote gateway. The PMTU is then stored in the SAD.

There are performance penalties with this mechanism, in particular because it generates additional packets on the black network (probes are ad-hoc packets, whereas PLPMTUd in TCP reuses application data in the probing packets). This mechanism is also less responsive to a PMTU change in the black network (e.g. caused by a route change) due to its use of timeouts. However we believe it's a good solution that prevents the attack we described[5].

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 team of Paterson [19], [20] has discovered several weaknesses related to an incorrect usage of encryption. [21], [22] focuses on DoS attacks against the "encryption

[5]This approach is still fragile in case of an attacker that can identify probes and selectively drop them. However this is an additional capability for the attacker that is not considered in our attack and that is not necessarily trivial to achieve.

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 [23]). DoS attacks such as Smurf attack [24] and Tribe Flood Network attack [25] 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 ICMP messages. 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 an host. Otherwise flooding the victim with "Echo Request/Reply" messages has long been considered. But this is also easily avoided with appropriate filtering rules on the IPsec gateway. So we see that our work totally departs from these "traditional" techniques.

[3], [26] that describe PMTUd acknowledge the existence of threats related to the use of ICMP PTB messages. (e.g. for blind throughput–reduction attack against TCP [16]). [10] details for the various ICMP (type,code) messages the associated threats and filtering advices to mitigate them. Our work goes more deeply in this direction and introduces a new threat with ICMP PTB messages.

VII. Conclusion

In this work we have shown that beyond the traditional DoS attacks, ICMP is also an attack vector against IPsec gateways for an attacker having eavesdropping and traffic injection capabilities in the black untrusted network, without any access to clear-text (he only sees ciphered packets). We demonstrated this attack in a Debian–based testbed and showed its efficiency to create DoS or major performance penalties, both with TCP and UDP flows.

We also discussed the fundamental problems and proposed countermeasures. In particular, within the IPsec gateways, we proposed to use PLPMTUd to probe the network and assess the various Path MTUs.

Future works will enlarge the scope of this study by considering other Operating Systems and IPv6. We will also experiment with our proposed gateway-level PLPMTUd to assess its efficiency.

References

[1] S. Kent and K. Seo, "RFC 4301: Security Architecture for the Internet Protocol," http://datatracker.ietf.org/doc/rfc4301/, December 2005.

[2] S. Kent, "RFC 4303: IP Encapsulating Security Payload (ESP)," http://datatracker.ietf.org/doc/rfc4303/, December 2005.

[3] J. Mogul and S. Deering, "RFC 1191: Path MTU Discovery," http://datatracker.ietf.org/doc/rfc1191/, November 1990.

[4] M. Luckie, K. Cho, and B. Owens, "Inferring and debugging path MTU discovery failures," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, ser. IMC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251086.1251103

[5] M. Luckie and B. Stasiewicz, "Measuring path MTU discovery behaviour," in *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 102–108. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879155

[6] Y. Desmedt, "Man-in-the-Middle Attack," in *Encyclopedia of Cryptography and Security (2nd Ed.)*. Springer, 2011, p. 759.

[7] J. Postel, "RFC 0792: Internet Control Message Protocol," http://datatracker.ietf.org/doc/rfc792/, September 1981.

[8] S. Kent, "RFC 4302: IP Authentication Header," http://datatracker.ietf.org/doc/rfc4302/, December 2005.

[9] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "RFC 5996: Internet Key Exchange Protocol Version 2 (IKEv2)," http://datatracker.ietf.org/doc/rfc5996/, September 2010.

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

[11] 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," 2009. [Online]. Available: http://infoscience.epfl.ch/record/134539

[12] 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 *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1629575.1629578

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

[14] L. Jacquin, V. Roca, M. A. Kaafar, F. Schuler, and J.-L. Roch, "IBTrack: an ICMP black holes tracker," *IEEE GLOBECOM*, pp. 2851–2857, 2012. [Online]. Available: http://hal.inria.fr/hal-00748153

[15] M. Mathis and J. W. Heffner, "RFC 4821: Packetization Layer Path MTU Discovery," http://datatracker.ietf.org/doc/rfc4821/, March 2007.

[16] F. Gont, "RFC 5927: ICMP Attacks against TCP," http://datatracker.ietf.org/doc/rfc5927/, July 2010.

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

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

[19] J. P. Degabriele and K. G. Paterson, "Attacking the IPsec Standards in Encryption-only Configurations," in *IEEE Symposium on Security and Privacy*. Oakland, California, USA: IEEE Computer Society, May 2007, pp. 335–349.

[20] J.-P. Degabriele and K. G. Paterson, "On the (in)security of IPsec in MAC-then-encrypt configurations," in *ACM Conference on Computer and Communications Security - CCS 2010*. Chicago, Illinois, USA: ACM, October 2010, pp. 493–504.

[21] C. B. McCubbin, A. A. Selçuk, and D. P. Sidhu, "Initialization Vector Attacks on the IPsec Protocol Suite," in *9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2000)*. IEEE Computer Society, 2000, pp. 171–175.

[22] V. Nikov, "A DoS Attack Against the Integrity-Less ESP (IPSec)," *IACR Cryptology ePrint Archive*, vol. 2006, p. 370, 2006.

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

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

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

[26] J. McCann, S. Deering, and J. Mogul, "RFC 1981: Path MTU Discovery for IP version 6," http://datatracker.ietf.org/doc/rfc1981/, August 1996.