DoS Protection for UDP-Based Protocols

Charlie Kaufman
IBM
ckaufman@us.ibm.com

Radia Perlman
Sun Microsystems Laboratories
radia.perlman@sun.com

Bill Sommerfeld
Sun Microsystems
sommerfeld@east.sun.com

ABSTRACT

Since IP packet reassembly requires resources, a denial of service attack can be mounted by swamping a receiver with IP fragments. In this paper we argue how this attack need not affect protocols that do not rely on IP fragmentation, and argue how most protocols, e.g., those that run on top of TCP, can avoid the need for fragmentation. However, protocols such as IPsec's IKE protocol, which both runs on top of UDP and requires sending large packets, depend on IP packet reassembly. Photuris, an early proposal for IKE, introduced the concept of a stateless cookie, intended for DoS protection. However, the stateless cookie mechanism cannot protect against a DoS attack unless the receiver can successfully receive the cookie, which it will not be able to do if reassembly resources are exhausted. Thus, without additional design and/or implementation defenses, an attacker can successfully, through a fragmentation attack, prevent legitimate IKE handshakes from completing. Defense against this attack requires both protocol design and implementation defenses. The IKEv2 protocol was designed to make it easy to design a defensive implementation. This paper explains the defense strategy designed into the IKEv2 protocol, along with the additional needed implementation mechanisms. It also describes and contrasts several other potential strategies that could work for similar UDP-based protocols.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General---Security and Protection.

General Terms

Algorithms, Performance, Design, Security, Reliability.

Keywords

DoS, IPsec, IKE, fragmentation, protocol design, network security, denial of service, buffer exhaustion.

CCS'03, October 27-31, 2003, Washington, DC, USA.

1-58113-738-9/03/0010...$5.00.

1. INTRODUCTION

One of the major concerns in the design of IPsec key management protocols has been to make them resistant to denial of service (DoS) attacks. Since IPsec implementations are deployed in environments that are assumed to be hostile, they must be able to establish security associations even while under attack.

The concept of stateless cookies as a protection against certain classes of DoS (denial of service) attacks originated with Photuris [8], an early key management protocol for IPsec. The purpose of stateless cookies is to defend against attackers that send traffic from fake source addresses, exhausting state and/or computation resources at the victim node. The reason for the attacker sending from forged IP addresses is twofold: to avoid prosecution for mounting a denial of service attack, and to make it difficult for a firewall to screen out traffic from the attacker. In the stateless cookie design, when a node "Bob" receives a connection initiation request from a node "Alice", Bob creates a number (called the "cookie"), unpredictable to Alice, returns that cookie to the IP address in the source address field in the IP header of the received packet, and keeps no state and does no additional computation. If the cookie is stateless, it must be recomputable by Bob, and is typically a function of the source IP address from the received packet and a secret known only to Bob. If Bob asks Alice to return a cookie before he is willing to consume significant resources, Alice must try again, this time returning the cookie. When Bob receives a connect initiate request with a cookie, Bob computes whether that is the cookie he would have sent to that IP address. If so, he is willing to devote state and computation to the connection from that IP address.

2. FRAGMENTATION ATTACKS IN UDP-BASED PROTOCOLS

Although in theory stateless cookies allow Bob not to devote state or significant computation until he is assured that Alice can receive at the address she claims to be coming from, in practice there is a DoS threat that a straightforward implementation of a UDP-based protocol will be vulnerable to, if (like IKE) it sends large packets and depends on IP fragmentation. An attacker can take advantage of the fact that IP fragment reassembly requires storing packet fragments of partially reassembled IP packets, which consumes memory resources on the victim. Since the kernel reassembly queue is limited in size this sort of flooding will prevent legitimate packets from being reassembled.

It was a decision in the design of IKE [3] to run on top of UDP, to avoid the DoS attacks on TCP. Another decision was to keep IKE simple and rely on IP fragmentation in order to send a large message. IKE messages can be large because they

contain structures such as certificates. The proposed successors to IKE, including JFK [1], and IKEv2 [4], also have made the decision to run on UDP and rely on IP fragmentation for delivery of large messages.

Protocols that run on top of TCP are not as vulnerable to the fragmentation attack, because TCP can avoid IP-level fragmentation. TCP can do this because it is connection-oriented, and because it is designed so that it can send data in chunk sizes independent of the size of application messages. However TCP itself is prone to various DoS attacks, and the decision to run on top of UDP was made with the intention to make the protocol more robust against DoS attacks.

But IKE's decision to send large messages, and use UDP, make it particularly vulnerable to the fragmentation attack described in this paper. This paper explains various strategies that a redesigned IKE, together with a DoS-resistant implementation of the IP stack, can employ to defend against such attacks. These strategies would be applicable to protocols similar to IKE which send large messages on UDP.

The attack has not been addressed in the literature (except for the IKEv2-related internet drafts). In [7] many types of DoS attacks are discussed, but the attack in this paper is not included. Without a defense against this attack it is easy for an attacker to send IP fragments, overwhelm the IP reassembly resources, and prevent an IKE exchange from ever completing. It can appear on paper that a protocol has been defensive against DoS, for instance by using stateless cookies, but still be vulnerable to this fragmentation attack, and therefore, in practice remain vulnerable to DoS. IKEv2, from the beginning [4], was designed to enable a defense against this attack, but it was not explicitly explained in the document. Description of the attack and proposed defense was described in [5] and [6]. After the attack and a particular defense strategy was described in [5] and [6], an alternative defense against the attack was proposed in [2]. In this paper, we will describe and contrast the defenses described in the various internet drafts, along with several alternate potential defenses.

2.1 IP Fragmentation

IP is designed to work over a variety of link types. Unfortunately, different link types have different maximum packet sizes. For instance, Ethernets have a maximum packet size of 1500 bytes. Also even if the link itself did not have a maximum packet size, routers on the link might have limited buffer sizes.

A few years ago, it was considered safe to assume that all links (and routers) could handle packet sizes of 576 bytes. If links (such as ATM, with cell sizes of 48 bytes) could not handle 576 bytes, hop-by-hop fragmentation would be employed. This means that the router neighbors on the link with the tiny packets would chop the packet up for transit across that link, but reassemble the packet on the other side of the link. This hid the packet size of that link from the rest of the network.

However, for links with "reasonable" packet size, the traditional approach has been to do fragmentation at the IP layer, with reassembly at the destination. These days it is generally assumed safe to assume that all links can handle about 1500 byte packets (though with extra headers due to tunnels, etc., "1500" sometimes means a little less than 1500), so packets smaller than 1500 bytes should not require fragmentation.

2.2 Stateless Cookies in IKE version 1

Version 1 of IKE's first phase consisted of two types of handshakes; a 6-message "main mode" handshake that did identity hiding, and a 3-message "aggressive mode" that also did mutual authentication and SA (security association) establishment, but did not do identity hiding. The basic structure of the full-featured (i.e., main mode) handshake is as follows, where we are leaving out fields that are not relevant to this paper. Curly brackets (i.e., "{" and "}") around a quantity indicate it is encrypted.

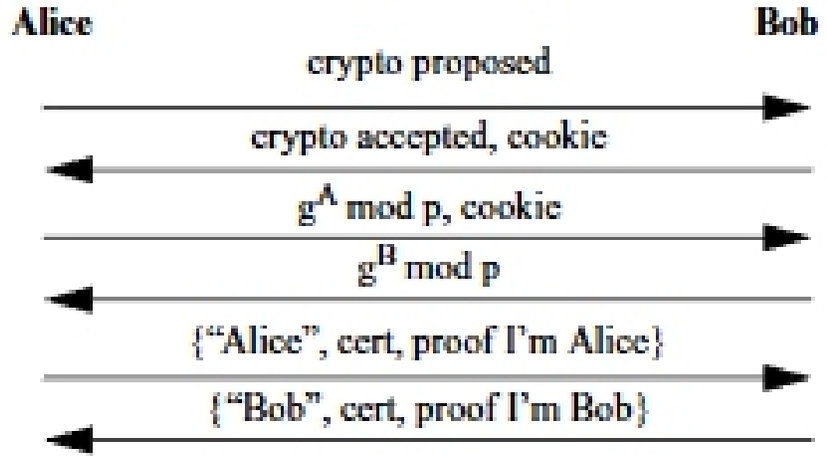

Figure 1. Basic structure of IKEv1 main mode

In [15], it was pointed out that the ISAKMP/IKE design precluded stateless cookies. In [12], it was asserted that the IKE exchange could be altered, without adding messages, by having the initiator (Alice) repeat in message 3, what she had sent in message 1. Although in theory this approach allows Bob to be stateless until he receives a valid cookie, in practice it would fail to prevent a fragmentation denial of service attack. Since message 3 is large and typically needs to be fragmented, an attacker that is using up reassembly resources can block reassembly of a legitimate initiator's IKE messages, preventing successful creation of an IPsec SA.

2.3 How likely is fragmentation?

How large are IKE messages? The largest field in IKE messages is likely to be the certificate. Certificates can be upwards of 500 bytes (many Verisign certificates are > 1000 bytes). And the "certificate" field can actually be a certificate chain containing multiple certificates. There are other potentially large fields such as:

- the Diffie-Hellman value (which for a 1024-bit group will be about 128 bytes),
- a certificate request field (not shown in the figure), which is a list of distinguished names of acceptable CAs (typically around 100 bytes each, and there might be many acceptable CAs), and
- the "proof" which is a public key signature, and in the case of 2048-bit RSA keys will be about 256 bytes.

In practice, in current IKE implementations, fragmentation is very common.

2.4 Defense, post-handshake

Once an IPsec SA is successfully created, there is connection state, and the endpoints of the SA can protect themselves against the fragmentation DoS attack by doing MTU

(maximum transmission unit) discovery to find out what size packets can be sent over that SA without needing to be fragmented. Once the MTU is known, IPsec itself (AH/ESP), in tunnel mode, can fragment packets that are too large to be forwarded over the SA. Then either the SA tunnel endpoint can reassemble the packets (which is not prone to DoS since each fragment is separately cryptographically protected, and therefore attackers' fragments will be thrown away without consuming reassembly resources), or the fragments can be decapsulated and forwarded on for the remainder of the path. In that case, reassembly is left as a problem for the true source and destination of those packets--the SA is just another link in the network with a limited MTU size, so they can do standard MTU discovery [10].

Therefore the only issue with protecting IPsec against this fragmentation attack is ensuring that SA establishment from legitimate IP addresses not get locked out during the initial IKE handshake.

Although the solutions proposed in this paper could in theory protect legitimate sources throughout an IPsec SA, an attacker that guesses the source address of a legitimate node could send IP fragments from that forged source address. Therefore, the defense is most robust if it depends on keeping an IP address on the list of preferred addresses for as short a time as possible. Therefore, the most robust defense is to only rely on fragmentation during the handshake, and use one of the defenses in this paper to defend the handshake from this attack.

2.5 How feasible is this attack?

There are many implementations of IP, and it's safe to say that each one does reassembly slightly differently. Differences in the algorithms used to limit the amount of storage used for packets being reassembled will cause some variation in how easy or difficult it is to mount this attack.

Two and a half implementations were examined -- the ones found in the Solaris(TM)[1] Operating System (Solaris OS) and NetBSD; NetBSD also includes a firewall package, ipfilter, which does its own fragment state tracking, hence the "half".

Different metrics are used on the Solaris OS and NetBSD. NetBSD counts the number of partially assembled packets, while the Solaris OS counts bytes in the fragments to be reassembled. Limits are tunable at run time. By default, the Solaris OS allows a megabyte per interface for reassembly, while NetBSD keeps at most 200 packets.

Both have similar lifetimes for partially assembled packets -- 60 seconds for the Solaris OS, 30 seconds for NetBSD, and 60 seconds for ipfilter. When the limits were reached, both use a "tail drop" scheme -- if there were too many bytes or packets pending, fragments from previously unseen packets would be dropped.

While this has not been experimentally verified, it appears that the NetBSD reassembly subsystem could be clogged by an attack involving as few as 7 small fragments per second, while the Solaris OS might be clogged by a few dozen large fragments per second.

[1] Solaris is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.

3. DEFENSES

In this section we explore and contrast various defenses against the fragmentation attack. As explained in Section 2.4, we only need to ensure that the IKE handshake completes in spite of this attack, since the remainder of the security association can be defended through straightforward means.

3.1 Small Initial Messages Defense

The defense in this section involves designing the protocol so that all messages are small until a cookie can be verified, and then having IKE (or a similar UDP-based protocol) pass a hint to the IP reassembly code as to which IP addresses should be preferred when reassembly resources are limited. If messages are small enough so that legitimate pre-cookie-verification packets will not require fragmentation, then the fragmentation attack will not interfere with cookie verification, and after cookie verification, fragments from verified IP addresses will get priority for reassembly resources.

To accomplish this, we needed to create an extra optional round trip in the IKEv2 handshake. To see why this is required, we first show, in Figure 2, a 4-message handshake that has all the properties the IPsec WG wants for IKE, including mutual authentication and identity hiding (hiding the names of the communicating parties from eavesdroppers), as well as a stateless cookie for DoS protection. However, message 3 in this handshake is likely to be sufficiently large that it requires fragmentation, since it contains certificates and must repeat information from the first two messages. We are only showing the fields necessary to illustrate the point of this paper.

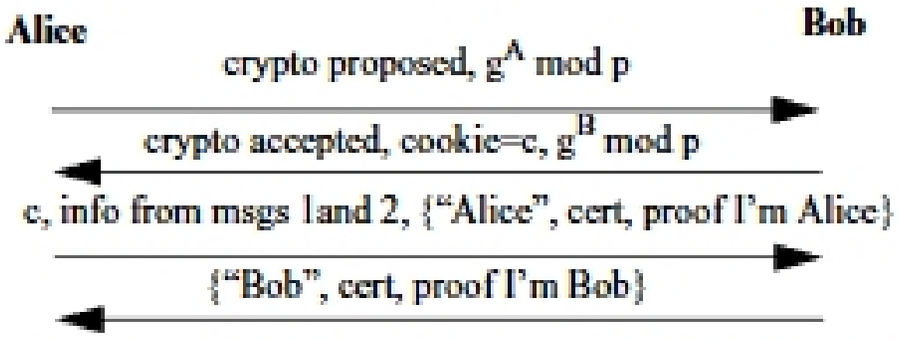

Figure 2. 4-msg handshake: msg 3 depends on fragmentation

In Figure 3, we show a handshake which is a 6-message protocol.

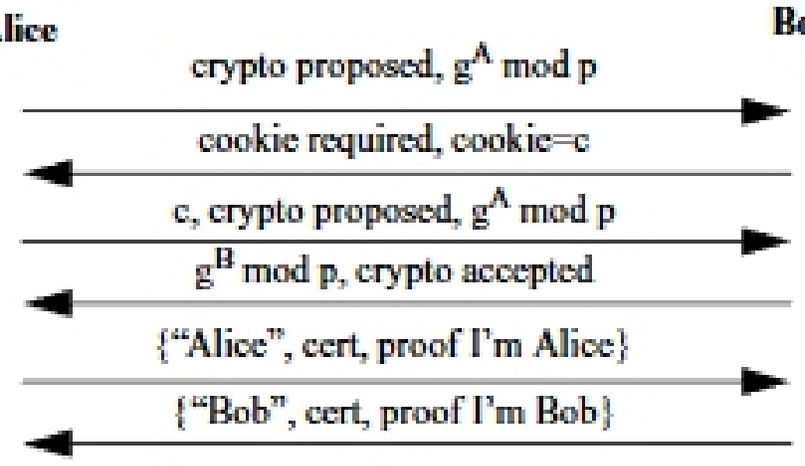

Figure 3. With optional additional round trip

The protocol in Figure 3 has the disadvantage over the one in Figure 2 of requiring an extra two messages (i.e., an extra round trip). However, the purpose of adding the extra messages is that the result is that messages 1, 2, and 3 are all small enough that they would not require fragmentation.