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.

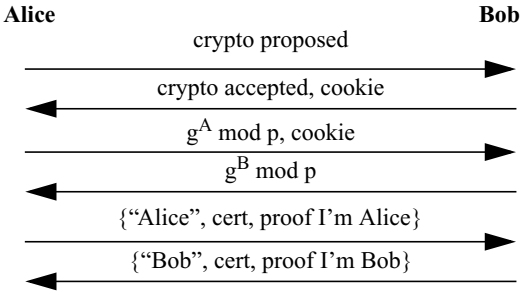

Figure 1. Basic structure of IKEv1 main mode

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.

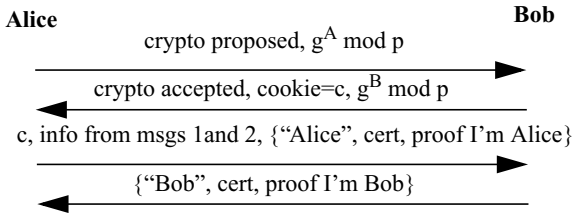

Figure 2. 4-msg handshake: msg 3 depends on fragmentation

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

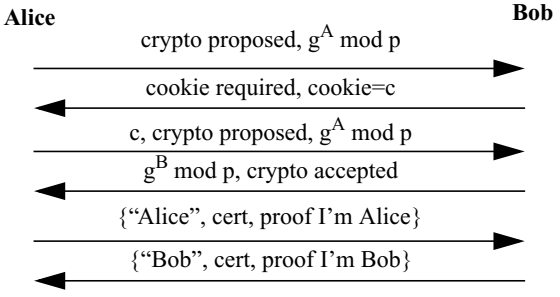

Figure 3. With optional additional round trip

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.

Therefore, this defense (together with additional implementation measures) allows the handshake to proceed successfully until Bob can verify the cookie.

3.1.1 The necessary implementation modification

In the IP stack in a typical operating system, IKE will be above UDP, which will be above IP. Ordinarily there is no channel by which IKE could give the IP reassembly code hints, but it is not hard to implement such, even though it is somewhat of a layer violation. If IKE is designed so that it can verify which IP source addresses have returned valid cookies, even in the presence of a fragmentation attack, IKE can pass a list of preferred IP addresses to the reassembly code, allowing resources to be used for completion of IKE handshakes from IP addresses that have returned valid cookies.

The scheme we are proposing has the IP reassembly code, when resources are getting scarce, devote resources to IP addresses recommended by IKE (based on having received a valid cookie).

3.1.2 The guessed IP address attack

There is an additional subtlety, which is that attackers that can guess IP addresses on the preferred list will still be able to mount a fragmentation attack, since the IP reassembly code would not be able to distinguish packet fragments from the legitimate source from packet fragments from an attacker.

To guard against this attack, an address should stay on the preferred list only long enough to complete the IKE exchange. Once an SA is established, there are other ways to defend against a fragmentation attack. The endpoints of the SA can, as TCP can, discover the MTU, and prefragment a large packet to a size that will not need to be fragmented along the path, with each fragment separately IPsec-protected. Reassembly at the SA endpoint can be done above the IP reassembly layer, i.e., after the fragment has been verified authentic by IPsec. Or if the SA is merely a VPN link along the path, say between two firewalls, then the fragments can be forwarded after being decapsulated.

Therefore, there is only a small time window in which IP reassembly must favor traffic from an IP address in order for the IKE exchange to complete and allow the SA to be established. In the 6-message protocol, the window is between receipt of message 3 (in which the cookie is received and verified) and the successful establishment of the SA. It would be tempting to assume that Bob's IKE can remove that IP address from the preferred list after receipt of the reassembled message 5. However, if Bob's message 6 were lost, then Alice will retransmit the (still large) message 5. Thus Bob will need to keep the IP address on the preferred list until he is assured that the SA has been established. This can be done after successful receipt of traffic on that SA, or (somewhat dangerously) after a timeout after receipt of message 5.

3.1.3 Making the extra messages optional

The extra round trip can be made optional (and actually is, in the IKEv2 protocol). Bob can choose, in message 2, whether to do the handshake in Figure 2 or the one in Figure 3, and Alice will know which one he has chosen based on his reply to her initial message. The idea of having the stateless cookie be an optional initial round-trip was proposed in [11]. The choice of doing the stateless cookie exchange with the optional initial round trip realizes the goal of keeping legitimate pre-cookie-verification messages sufficiently small so as not to require fragmentation.

How would an implementation know if it was under attack? The simplest strategy is to always use the 6-message protocol, since it is the safest. An alternative is to count the number of partially completed handshakes, and revert to the 6-message protocol if that number goes over some threshold. This is equivalent to saying that the implementation reserves a fixed amount of resources to IKE handshakes that have not returned a cookie.

3.1.4 Avoiding a CPU exhaustion attack

Note that in the stateless 4-message protocol in Figure 2 we are assuming that Bob will reuse the same Diffie-Hellman exponent ("b") for many exchanges, perhaps encoding into the cookie which "b" he'd chosen. Otherwise, if Bob had to compute a unique g^b mod p for each connection attempt, he would be vulnerable to a CPU resource exhaustion DoS attack. Reusing the same "b" does result in less than perfect PFS (perfect forward security) if knowledge of "b" persists beyond a connection whose key is computed from b and information exchanged over the wire.

If the stateless cookie is instead done as in Figure 3, as an optional additional round trip, then Bob will be keeping state starting with message 3, and there is no need to repeat the information from messages 1 and 2 into message 3.

3.2 First Fragment Defense

In this section we describe how to defend against the fragmentation attack with the 4-message design in Figure 2, despite the fact that message 3 is large, and might require fragmentation.

It is possible to protect against the fragmentation attack without adding an extra round trip for the stateless cookie. This is done by an additional unorthodox, but easily implementable, channel between the reassembly code and the IKE code.

Recall that message 3 in Figure 2, which includes Alice's return of Bob's cookie, is large enough to require fragmentation, because it contains information such as certificate chains.

The solution is to design message 3 so that Bob's cookie is the first item in the message, and to allow the reassembly code to pass the first fragment of an unassembled IP packet up to IKE. There will be enough information in the fragment for the reassembly code to detect that:

- it is the first fragment (offset=0)
- it is UDP (protocol type=UDP)
- port in UDP header=500.

This is additional layer violation beyond the defense in Section 3.1, since it requires the IP reassembly code to look at the UDP header, and it requires a more radical change to the API (passing up an unassembled IP fragment). But although unorthodox, it certainly is not difficult, and it is routine for routers (which claim to be layer 3 devices) to look beyond the IP header for firewalling and QOS-categorization reasons.

There are two variants of this approach. In one approach, the IP reassembly code always (whether or not it is at the resource limit) sends the first fragment of a fragmented IKE packet up to the IKE code. In the other variant, the IP reassembly code only passes up the first fragment if it was forced to throw away fragments due to resource exhaustion. In the first approach, Bob might be lucky and not need to throw away any fragments if the first fragment is validated by IKE, and IKE informs the reassembly process to add the address to the preferred list before Bob discards other fragments of that packet. In this case, Alice's IP address will only have to remain on the preferred list until message 3 is reassembled.

In the second approach, Bob's IKE only gets to inspect the first fragment of message 3 if the reassembly resources are exhausted and Bob is forced to discard fragments. In that case, Alice's IP address will have to remain on the preferred list until Alice's retransmission timer expires and Alice retransmits message 3, this time hopefully successfully reassembled because her IP address will be on the preferred list.

3.3 Avoiding-IP-Fragmentation Defense

Another approach to avoiding the fragmentation attack is to design IKE so that it does not depend on IP fragmentation. This requires first discovering the path MTU (PMTU) and then doing application-level fragmentation in order to make the packets smaller than the PMTU. So, instead of sending the entire message 3 from Figure 2 in a single chunk, it breaks the message into fragments of appropriate size and has the IKE peer at the other end reassemble the messages. In order to prevent fragmentation-flooding attacks on the IKE process, each application-fragment (remember, in this approach there are no IP-level fragments) should include the cookie, and any fragment containing a bogus cookie will be rejected by IKE.

There are two ways of ensuring that fragments are no larger than the PMTU. The first is to use essentially the technique of RFC 1191 and set the DF bit in each packet. Thus, an overlarge packet will be rejected with an ICMP error. This method has the drawback that it introduces latency (if Alice guesses too large a PMTU) even when no attack is in progress.

An alternate approach is to not set the DF bit. Then, if Bob is under attack and forced to flush his fragment reassembly queue, he can send the ICMP Time Exceeded (Reassembly) message. When Alice gets this message, she knows that she needs to back off and then can use the explicit PMTU discovery mechanism mentioned above. The difficulty here is that many implementations do not send the Time Exceeded message under these circumstances. However, those implementations concerned with defense against this attack could easily do so. Those not concerned with defense would do nothing.

Although this approach offers good protection, it makes the IKE protocol and implementation somewhat more complex since it has to do the work that would be done by IP fragmentation. Moreover, the efficiency is just as bad as ordinary IP fragmentation [9]. One might improve the efficiency of the fragmentation mechanism by introducing application-layer ACKs for individual chunks. This would remove the need to retransmit the entire message when a single chunk is lost. However, it would greatly complicate the protocol state machine. It might well be easier to simply use TCP instead of UDP.

3.4 Using-an-IP-option-for-the-cookie defense

Another potential defense has IKE inform IP of the strategy for cookie verification, and carry Bob's cookie value in a newly defined IPv4 option, or IPv6 extension header. For example, if the cookie is a function of the IP address and a secret S, then IKE will inform IP of the function and S. Alice's IKE would, in this strategy, insert the IP option with Bob's cookie into all the handshake messages following receipt of Bob's cookie. In this way, IP can discard all fragments with an invalid cookie, or nonexistent cookie. To ensure that all fragments contain the IP option, the "copy" flag on that option must be set.

The usual objection to anything involving IPv4 options is that current router implementations forward packets with options inefficiently, since they will not be able to be forwarded through the fast path. Another issue with options is that they are sufficiently rare these days that router code in some implementations has evolved that will mishandle packets with options. Header compression implementations are notorious for mangling headers with IP options.

Assuming an IP option would be handled correctly but slowly by the routers, this is not necessarily a problem for IKE. All this means is that the handshake messages will be slower at traversing the network. However, it might open up a new avenue for a DoS attack, which is to send a lot of nonstandard packets into the network, using up the CPU in a router's central processor. But this attack would not be introduced by this mechanism (an IP option to carry cookies), since an attacker could already use any unusual IP packets to swamp the slow path of existing routers.

The fragmentation defense in this section is much more radical than others proposed. It involves not only giving the IP reassembly code hints about preferred addresses, but actual code for verifying cookies. However, it is more resilient against the guessed-IP-address attack described in Section 3.1.2.

3.5 Using-the-IP-pktID-for-the-cookie defense

After the fragmentation attack and proposed defense was described in [4], the defense in this section was proposed and described in the 4th draft of JFK [2]. This defense is similar to the defense in Section 3.4 above, but has the cookie appear in the 2-byte packet identifier field in the IP header instead of as an IP option. As with the defense in Section 3.4, IKE will need to tell IP the cryptographic algorithm for verifying cookies. Since the packet identifier appears in all IP packets, Bob's IP reassembly code will need to compute the expected cookie on every fragment, even those on non-IKE packets, since fragments other than first fragments cannot be distinguished as IKE fragments. If the packet identifier field in the IP header matches the bottom 2 bytes of the cookie Bob's IKE would have computed for that IP address, then Bob's IP reassembly code queues that fragment on a preferred queue; otherwise, on a non-preferred queue.

This scheme has additional disadvantages over the defense in Section 3.4:

- It puts a lot of computation responsibility on the reassembly code (a cryptographic hash must be computed for every IP fragment; even those that are not fragments of IKE packets, since other than the first fragment, an IKE packet cannot be distinguished from packets from other protocols).

- The packet identifier field is already used by IP for other purposes, and if other processes at Alice are transmitting packets that must be fragmented, it is possible that the value in the bottom 2 bytes of the cookie she received from Bob has already been assigned to a packet identifier of a recently transmitted IP packet.

4. CONCLUSIONS

This paper describes a fragmentation DoS attack unique to protocols such as IKE that run on top of UDP and require sending large packets. Applications that run on top of TCP can defend against the attack in more straightforward ways. The paper presents several strategies for defense against a fragmentation DoS attack by UDP-based protocols such as IKE. The proposed defenses are:

- designing the protocol so that messages are small enough not to require fragmentation until a cookie is verified, and having IKE pass to the IP reassembly process a list of preferred IP addresses (those that have returned a valid cookie).
- designing the protocol so that the cookie is in the first fragment, and changing the API so that IP can pass the first fragment of an unreassembled IP packet to a process, if the process requests. Then, once IKE has verified the cookie in the first fragment, IKE will inform the reassembly process to add the IP address from which the fragment was received to its preferred list.
- designing IKE to do its own MTU discovery and fragmentation, so as not to depend on IP reassembly
- putting cookies into a new IP option, and having IKE pass the cookie-verification algorithm to IP, which will verify all fragments carrying that option
- having IKE pass the cookie-verification algorithm to IP, as well as the cookie to be carried in the packet identifier field in the IP header.

Additional subtleties, such as the desirability of keeping an IP address on the preferred list for as short a time as possible, are also explained.

We prefer either of the first two defense strategies. Although they require layer violation, it is simple to implement and does not put undue burden on IP. The first strategy is a simpler API change than the 2nd since the first only requires having IKE pass preferred IP addresses to IP, and the 2nd additionally requires having IP pass up first fragments of unfragmented packets.

The strategies that require IP to verify cookies (the last two defenses) require a much more difficult interface, and especially the final proposed defense presents an undo burden on IP because it requires cryptographic verification of the packet identifier on every fragment, even those that do not belong to IKE packets. The third defense (designing IKE to do its own MTU discovery), has the advantage of not changing the API, but does require a more complex IKE.

Ultimately, the IPsec WG chose the IKEv2 variant with the extra round trip, due to its being more amenable to incorporating legacy authentication, and because of its simplicity. That decision alone does not mean that IKEv2 implementations will be invulnerable to the fragmentation attack, since defense requires the additional implementation mechanism. However, it does enable implementations to implement the first defense suggested in this paper, if fragmentation attacks start being seen in the wild.

5. ACKNOWLEDGMENTS

Eric Rescorla was particularly helpful in reading several drafts of this paper, and offering suggestions.

6. REFERENCES

[1] Aiello, W., Bellovin, S., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A., Reingold, O., “Just Fast Keying (JFK)”, draft-ietf-ipsec-jfk-00.txt, Nov 2001.

[2] Aiello, W., Bellovin, S., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A., Reingold, O., draft-ietf-ipsec-jfk-03, April 2002.

[3] Harkins, D., and Carrel, D., “The Internet Key Exchange (IKE)”, RFC 2409, November 1998.

[4] Harkins, D., Kaufman, C., and Perlman, R., “The Internet Key Exchange (IKE) Protocol, draft-ietf-ipsec-ikev2-00.txt, Nov 2001.

[5] Harkins, D., Kaufman, C., Kent, S., Kivinen, T., and Perlman, R., “The Internet Key Exchange (IKE) Protocol, draft-ietf-ipsec-ikev2-01.txt, Feb 2002.

[6] Harkins, D., Kaufman, C., Kent, S., Kivinen, T., and Perlman, R., “Design Rationale for IKEv2, draft-ietf-ipsec-ikev2-rationale-00.txt, Feb 2002.

[7] Leiwo, J., Nikander, P., and Aura, T., “Towards network denial of service resistant protocol. In Proceedings of the 15th International Information Security Conference (IFIP/SEC), August 2000.

[8] Karn, P., “The Photuris Key Management Protocol”, internet draft draft-karn-photuris-00.txt, December 1994.

[9] Kent C., and Mogul, J, “Fragmentation Considered Harmful”, ACM SIGCOMM, 1987.

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

[11] Orman, H., “The OAKLEY Key Determination Protocol”, RFC 2412, November, 1998.

[12] Perlman, R., and Kaufman, C., “Key Exchange in IPsec: Analysis of IKE”, IEEE Internet Computing, Nov/Dec 2000.

[13] Perlman, R., and Kaufman, C., “Analysis of the IPsec key exchange Standard”, WET-ICE Security Conference, MIT, 2001, http://sec.femto.org/wetice-2001/papers/radia-paper.pdf.

[14] Simpson, W. A.: “Photuris: Design Criteria”, Selected Areas in Cryptography 1999: 226-242

[15] Simpson, W. A., “IKE/ISAKMP Considered Harmful”, Usenix ;login, December 1999, Volume 24, Number 6.