Attacking Predictable IPsec ESP Initialization Vectors

Antti Nuopponen, Netseal
Sami Vaarala, Netseal
antti.nuopponen@netseal.com
sami.vaarala@iki.fi

Helsinki University of Technology
Tik-110.452 Special course in practical security of data system
Spring 2002

Abstract

Predictable initialization vectors in IPsec ESP encryption, allowed by the IPsec specifications and used by most implementations, compromise IPsec confidentiality. By using an adaptive chosen plaintext attack, an attacker can break low entropy plaintext blocks using brute force, and confirm guesses of the contents of arbitrary plaintext blocks. We analyze the preconditions and the seriousness of such attacks, and provide results of practical attack experiments.

1 Introduction

The IP Security Architecture (IPsec) [4] is widely used for end-to-end connection encryption, for remote access to a protected intranet, and for interconnecting sites using encrypted VPN tunnels. The currently specified IPsec ESP encryption algorithms use cipher block chaining (CBC) mode [7, 8]. The initialization vector (IV) is included in the ciphertext of every packet to allow the receiver to decrypt individual packets regardless of packet loss or reordering of packets.

The specifications for ESP DES [7] and other ciphers [8] do not specify an explicit IV selection algorithm, but require that the algorithm satisfy certain properties. RFC 2451 [8] states that:

> The IV field MUST be same size as the block size of the cipher algorithm being used. The IV MUST be chosen at random. Common practice is to use random data for the first IV and the last block of encrypted data from an encryption process as the IV for the next encryption process.

and further that:

> To avoid ECB encryption of very similar plaintext blocks in different packets, implementations MUST NOT use a counter or other low-Hamming distance source for IVs.

Note that the specification allows predictable – but random – initialization vectors, and explicitly allows the common practice of using the last ciphertext block as the next initialization vector.

The use of predictable initialization vectors leads to an adaptive chosen plaintext attack, which was pointed out by Scott Fluhrer on the IPsec working group mailing list. The attack allows an attacker to break low entropy plaintext blocks using brute force, and confirm guesses of the contents of arbitrary plaintext blocks. In this paper, we analyze the preconditions and seriousness of such attacks, and provide results of practical attack experiments that confirm the vulnerability in practice.

1.1 Some Known CBC Weaknesses

A known CBC weakness is "ciphertext collision": two identical ciphertext blocks in a CBC stream leaks information to the attacker. Let c_i be the ith ciphertext block and p_i the corresponding ith plaintext block. If $c_i = c_j$, then

$$c_{i-1} \oplus c_{j-1} = p_i \oplus p_j$$

where i and j are arbitrary indices. If p_i and p_j have low entropy, the attacker has a high probability of uncovering both plaintext blocks. [9]

Vaudenay [9] presents an attack on CBC when padding of the last plaintext block uses a simple form, such as the one used in IPsec. The attack requires modification of the ciphertext and an oracle that verifies the correctness of the decrypted padding after modification. However, the attack does not seem feasible if ESP or AH authentication is used.

1.2 Terms

The term *victim host* refers to a host that performs IPsec ESP encryption and possibly ESP or AH authentication. A *victim packet* is an IPsec-protected plaintext packet whose (arbitrary) plaintext block, the *victim block*, the attacker wants to guess.

The term *attack packet* refers to the IPsec-protected plaintext packet that the attacker forces the victim host to encrypt and send. The first plaintext block of the attack packet is called the *attack block*.

The term *chained IV* refers to the common practice of using the last ciphertext block of the previously encrypted packet as the initialization vector for the next encrypted packet.

2 Description of the Attack

2.1 The Attack

Figure 1 describes the ESP CBC processing of the victim and attack blocks. The attacker observes the victim packet (and block), and then causes the victim host to encrypt and send the attack packet. The victim block may be any block of the victim packet, including the first and last plaintext blocks, while the attack block is always first in the attack packet. The victim and attack packets do not

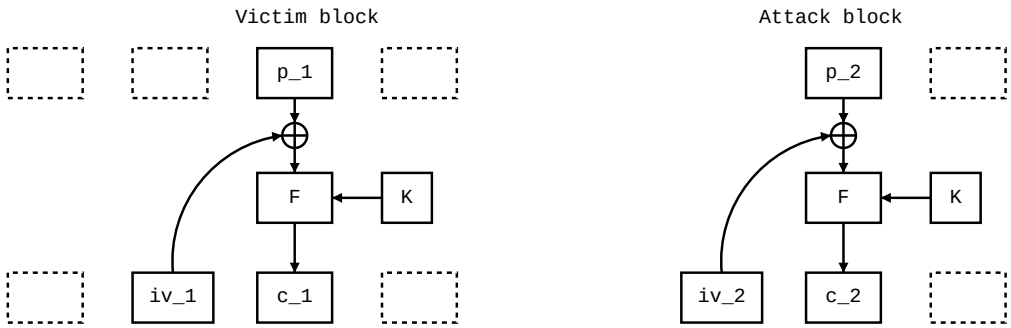

Figure 1: The victim and attack packets

have to be adjacent in the packet stream; the victim may encrypt and send one or more packets between the victim and attack packets.

By definition of CBC, the encrypted victim block is

$$c_1 = F(K, p_1 \oplus iv_1),$$

where F is the block encryption function (e.g. 3DES), K is the key, p_1 is the victim block, and iv_1 is either the IV of the packet, if the victim block is the first block in the packet, or the previous ciphertext block, otherwise.

Similarly, the encrypted attack block is

$$c_2 = F(K, p_2 \oplus iv_2).$$

The attacker chooses the attack block, p_2, as

$$p_2 = iv_1 \oplus iv_2 \oplus G,$$

where G is the attacker's guess of the victim block, p_1, and iv_2 is the attacker's prediction of the ESP initialization vector that the victim host will use to encrypt the attack packet. The encrypted attack block is then

$$c_2 = F(K, iv_1 \oplus iv_2 \oplus G \oplus iv_2) = F(K, G \oplus iv_1).$$

If the guess G is correct, c_2 will equal c_1, confirming the attacker's guess of the contents of the victim block.

The preconditions for the attack are discussed in Section 3.2.

2.2 Previous Work

The attack studied in this paper was clearly outlined by Scott Fluhrer in an e-mail to the IPsec mailing list[1]. Although the e-mail was related to AES, the attack is independent of the underlying cipher.

[1]See [11], message titled "Suggested modification to AES privacy draft", January 2002.

The security requirements for ESP initialization vectors have been debated on the IPsec mailing list since the beginning of the IPsec working group.

Phillip Rogaway pointed out that the initialization vector generation (and security related algorithms in general) should be concretely specified to avoid cryptographically unsound implementations. He also suggested that a correlation between the initialization vector and the first plaintext block is harmful [10]. The conclusion reached by the working group was that the IV generation has to be random to avoid correlations; unpredictability was not explicitly required.

Later, Hugo Krawczyk pointed out a chosen plaintext attack against predictable IVs (especially chained IVs). The attack reveals the cipher key being used for encryption, but requires $O(2^n)$ memory, where n is the cipher key size in bits[2].

Philip Gladstone suggested that, although unlikely, a predictable initialization vector might open IPsec up to a chosen plaintext attack[3]. Others commented that such an attack is not practical; the attacker cannot choose the plaintext directly because there are protocol headers before the actual plaintext, and the attacker does not have full control of the headers.

The overall consensus seems to have been that it is sufficient that the IV does not correlate with plaintext. Thus, a random IV, predictable or not, is acceptable – in particular, the common practice of IV chaining is acceptable. The attack described by Hugo Krawczyk is difficult to exploit in practice, especially against a cipher with a large keyspace, and the potential vulnerability against chosen plaintext attacks (described by Philip Gladstone and others) was also considered impractical.

In his e-mail, Scott Fluhrer described how predictable IVs could be exploited. His exploit indicates that the proper requirement is indeed that the IV should be unpredictable – not merely random.

3 Analysis of the Attack

3.1 Assumptions

Throughout the discussion below we assume that a cipher with a 64-bit block size is used. The attack applies to arbitrary block sizes, but the analysis details vary depending on how the block boundaries align with the protocol headers and data. Similarly, we only cover IPv4 although the attack applies to IPv6 as well.

3.2 Preconditions for the Attack

3.2.1 Observing the Victim Packet

The attacker must be able to observe a victim packet in ciphertext form, and to extract the ciphertext block, c_1, corresponding to the victim block, p_1, and the initialization vector used in encrypting the victim block, iv_1.

[2] See [11], message titled “pf_key comments (predictable IVs)”, January 1997.

[3] See [11], message titled “Re: I-D ACTION:draft-ietf-ipsec-skipjack-cbc-00.txt”, May 1999.

3.2.2 Making a Guess of the Victim Block Contents

The attacker has to make a guess about the entire contents of the victim block in order to generate one attack packet. If there are n possible contents for the victim block, the attacker has to try each of the n alternatives in turn.

3.2.3 Predicting the Initialization Vector

The attacker must be able to predict the initialization vector, iv_2, which the victim host will use to encrypt the attack packet. The prediction can be verified from the encrypted attack packet.

If the victim host uses IV chaining, this precondition means that the attacker has to capture the last ciphertext block of the packet encrypted immediately prior to the attack packet.

3.2.4 Forcing the Attack Packet to Be Sent

Determining the contents of the attack block is simple, involving simply XOR. However, when the victim host processes the attack packet, the attack block corresponds to a protocol header in transport mode, and an IP header in tunnel mode. Thus, the attack block must meet any validity constrains of the header in question; otherwise the victim host will refuse to encrypt the packet.

If the attacker determines that the attack block does not meet such validity constraints, the attacker simply forces the victim to encrypt and send a dummy packet (which meets the constraints). This resets the predicted IV to a new value, and changes the corresponding attack block to a new value. The attacker then simply tries again with the new attack block. If necessary, the attacker can iterate this process indefinitely until the attack block meets the validity constraints (or the security association expires).

Having obtained an attack block (dummy or not), the attacker must force the victim host to encrypt and send the attack packet. This is not an easy task; transport and tunnel mode each require a different approach. We will analyze this precondition in more detail in Section 6.

3.2.5 Verifying the Result

The attacker must be able to observe the encrypted version of the attack block. If the encrypted attack block (c_2) equals the encrypted victim block (c_1), the guess in the attack packet was correct.

The attacker should also be able to verify that the encrypted IPsec packet received is actually the encrypted attack packet and not some unrelated encrypted packet. If this condition is not met, the attack may yield a false negative.

Verifying this condition reliably seems impossible because of encryption. However, the attacker may use e.g. timing and length of the encrypted packet as sanity checks and redo the attack if such checks fail. The attacker may also simply rely on her luck and compensate by attempting every guess several times.

The attacker must also verify that the attack packet was encrypted using the same cipher key as the

victim packet. This can be done by simply verifying that the SPI fields in the two packets agree because the SPI maps statically to the cipher parameters, including the key.

3.3 Analysis of the Preconditions

Observing the victim block requires that the attacker be able to passively monitor the packets routed between the victim host and the other IPsec endpoint; the attacker may also use a routing attack to get access to the packets. If the victim uses IV chaining, predicting the IV is trivial; however, any predictable method of choosing the IV opens up the same vulnerability. Verifying the resulting encrypted attack packet is trivial.

The difficult preconditions are covered in separate sections. Section 4 covers guessing of the victim packet contents, Section 5 discusses how the attacker forms an attack packet that meets validity constraints imposed by the victim host, and Section 6 discusses how the victim can be forced to encrypt and send the attack packet.

3.4 Confirming that an Implementation is Vulnerable

The vulnerability can be confirmed by the attacker before she decides to mount an attack. The attacker can simply monitor the encrypted traffic flow and ensure her IV prediction algorithm works correctly. If the attacker cannot correctly predict the IVs, she can simply give up the attack as futile without wasting resources or getting caught while attempting an active attack.

If IKE [6] is used to set up the IPsec security associations, vendor identification payloads in the phase 1 IKE messages may provide a clue about the IV generation algorithm used by the implementation.

3.5 Kinds of Attacks

3.5.1 Brute Force Guessing

The attacker may simply iterate through a set of possible plaintexts in the victim block. Such attacks are difficult to mount if there are more than one or two octets to guess, because the amount of attack traffic easily becomes excessive.

Some potential realizations of this kind of attack:

1. The victim sends a TCP segment with a single character of e.g. password data. The attacker iterates through all the possibilities and discovers the character. (We tried this attack; see Section 7 and [14].)

2. The victim downloads a file from an IPsec-protected FTP site. The attacker iterates through all potential files to determine which one was downloaded. In more detail, the attacker identifies one or more plaintext blocks that are different in every potential file, and then iterates through these plaintext block possibilities.

3.5.2 Confirming Suspected Plaintext

The attacker may have a strong suspicion about the plaintext, and simply wants to use the vulnerability to verify her guess. In such cases, the attack is extremely efficient, and can verify a large amount of plaintext very easily.

Some potential realizations of this kind of attack:

1. The victim is sending an e-mail to a correspondent. The attacker verifies the receiver (and the sender) of the mail by predicting what the headers should (probably) look like, and by verifying these plaintext guesses. If the position of the address in the packet is uncertain, the attacker simply shifts the guess through all possible positions in the packet.
2. The victim accesses a web page. The attacker has a suspicion that the victim is accessing a certain URI, and verifies her suspicion. Note that even if the web server address is known, the attacker may be interested in knowing which URI the victim is accessing. (We verified that this attack is feasible; see Section 7 and [13].)
3. The attacker may confirm which service the victim is accessing by verifying her suspicion about e.g. TCP ports.
4. The attacker may routinely scan all e-mail correspondence of the victim for a few chosen words, by trying each word at every possible place in each IPsec packet that might be related to e-mail.

3.6 Effort Estimate

To uncover the contents of a single victim block requires an average of

$$N = \frac{W}{2p}$$

attempts, where p is the probability that the computed attack block meets the validity constraints imposed by the victim host (given a predicted IV), and W is the maximum number of possible plaintexts p_1.

Suppose that, $p = 2^{-16}$ (which is a realistic figure for IPsec tunnel mode attack; see Section 7) and $W = 256$, In this case, $N = 2^{23}$. At 100 packets per second, the attack requires an average of about 23.3 hours. At 1000 packets per second, the average is 2.3 hours.

When using chained IVs, the time between sending an attack packet and observing the encrypted version (containing the next IV prediction) dictates the maximum rate of attack packets. Thus, network latency plays a crucial role in the feasibility of the attack (e.g. 100 packets per second corresponds to 10 ms cycle time).

If the attacker is able to predict IVs for multiple packets in advance, latency becomes less important for feasibility. A pseudo random IV generator independent of the ciphertext (or plaintext) in previous packet(s) would allow this, but such generators are not used in practice to our knowledge.

Octet number 1	2	3	4
0x7a	0x01 (pad)	0x02 (pad)	0x03 (pad)

5	6	7	8
0x04 (pad)	0x05 (pad)	0x05 (pad len)	0x04 (next)

Figure 2: A padded plaintext block example

4 Guessing the Victim Block

It is not feasible to attack a completely unknown plaintext block. The attacker thus needs to somehow limit the number of alternative plaintext blocks. Doing so is dependent on how the plaintext aligns with the cipher blocks.

Attacking the last plaintext block of the victim packet is usually easier than attacking the other plaintext blocks, because the last plaintext is padded with a (usually) deterministic padding.

The ESP padding consists of 0...255 padding octets, followed by a “pad length” field indicating the number of such padding octets, followed by a “next header” field indicating which protocol ESP protects. The padding octets are used to bring the total amount of plaintext to a multiple of the cipher block size. However, the implementation is allowed to add extra padding octets in order to conceal the true length of the encrypted data. The padding octets are specified to have the form 0x01, 0x02, 0x03, etc, unless the cipher in question has a different requirement. [5]

In practice, all cipher algorithms for ESP use the default padding octet sequence. Since most implementations also use a minimum size padding, the entire sequence of octets following the plaintext data (padding, “pad length”, and “next header”) is completely deterministic. The “next header” field contains the transport protocol being protected in IPsec transport mode, and the value 0x04 (IP-IP tunnelling) for IPsec tunnel mode.

An example: suppose that the last plaintext block contains a single data octet 0x7a, and tunnel mode is used. The whole last plaintext block is shown in Figure 2. All octets except the data octet are known (assuming that the implementation uses the shortest padding sequence).

Attacking the first user data octets is, in many cases, complicated by interference from a preceding protocol header; if the first user data octets share a cipher block with the protocol header, the attacker must guess the protocol header in addition to the user data.

We next cover how the user data in UDP and TCP may be guessed in both transport and tunnel mode. In both cases, IPv4 and a cipher with 64-bit block size is assumed.

4.1 UDP

In transport mode, the UDP payload begins at an eight octet boundary, and thus there are never any unpredictable octets (other than the unknown data) in the plaintext blocks.

In tunnel mode, the IPv4 header preceding the UDP header changes the alignment (assuming

there are no IPv4 options): the first plaintext block containing UDP payload data also contains the "length" and "checksum" UDP header fields. The attacker may either try to guess both fields (which can be done with a good probability), or try to force IPv4 options that fix the alignment[4] to be used (which is difficult, because the attacker does not construct the packet).

It the victim block is farther in the UDP data, such changes in alignment do not require guessing any header fields.

4.2 TCP

TCP options [3] change the alignment of TCP user data with the cipher blocks. Similarly to UDP in tunnel mode, the attacker may compensate by guessing the TCP header fields in addition to the user data. The attacker may also try to force the victim to use suitable TCP or IPv4 (in tunnel mode) options that fix the alignment, but this is difficult because the attacker cannot directly affect the construction of the victim packet.

In transport mode without TCP options, the "checksum" and "urgent pointer" fields of the TCP header interfere with the first four octets of user data. While the urgent pointer is almost always zero, the checksum field is considerably harder to guess, because it is affected by e.g. the sequence and acknowledgement numbers, the window size, etc. A TCP header with $4 + 8n$ octets of options (and padding) does not interfere with the first plaintext containing data.

In tunnel mode, a TCP header without options does not interfere with the first plaintext block containing data. Thus, to attack TCP in tunnel mode, the attacker would prefer to either have no TCP options, or have an integral multiple of eight octets of TCP options.

5 Controlling the Attack Block

One of the preconditions of the attack described is that the attacker must be able to control the first plaintext block of the attack packet. Such control is heavily dependent on the IPsec encapsulation mode (tunnel or transport). We assume IPv4 and a cipher with 64-bit block size, although the attack applies to IPv6 and other cipher block sizes as well.

5.1 Transport Mode

In transport mode, the first plaintext block of the attack packet begins with the protocol header of the protocol carried inside the IP packet. Using the UDP transport protocol is the easiest method to force the victim host to encrypt and send a chosen plaintext block, because the UDP header [2] is exactly 64 bits long (which was assumed to be the cipher block size).

The attacker has almost full control of the UDP header (Figure 3), except for minor limitations: the length field has a minimum value (8), zero ports should not be used, and the attacker may not be able to force an arbitrary source port to be used (due to lack of privileges, for instance).

[4] Note, however, that some IPsec implementations do not deal correctly with IPv4 options.

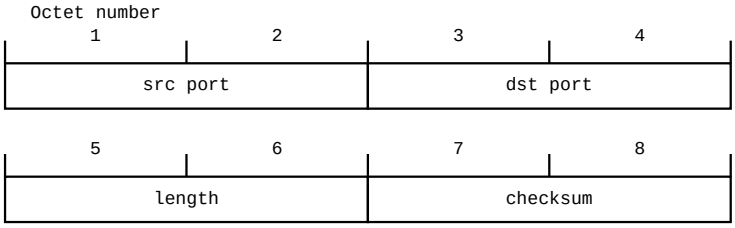

Figure 3: UDP header

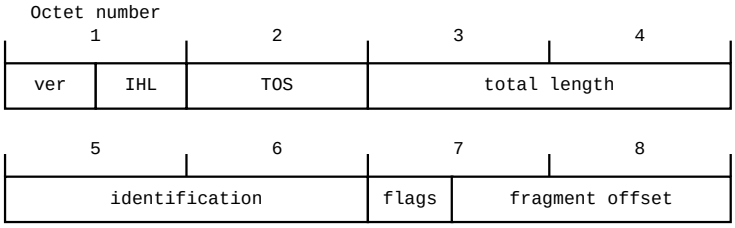

Figure 4: IPv4 header

5.2 Tunnel Mode

In tunnel mode, the first plaintext block in the attack packet consists of the eight first octets in the IPv4 header [1] (Figure 4).

The "Version" field contains four fixed bits. The "IHL" field has a value in the range 5—15; if we assume that the attacker does not use IP options, this field contains the value 5, and thus four fixed bits. The "Type of Service" field can be entirely controlled by the attacker, but may be modified by some routers; we assume that the attacker controls this field.

The "Total length" field is limited by the medium used; we assume Ethernet and thus the total length must be 20 at minimum and 1470 at maximum[5] The attacker can control $\log_2 1451 \approx 10.5$ bits, while approximately 5.5 bits cannot be controlled.

The "Identification" field can be controlled fully. The "Flags" field consists of three flags: the reserved bit (set to zero), the "Don't Fragment"-bit (assumed set to zero to avoid problems), and the "More Fragments"-bit, which can be controlled. The "Fragment Offset" field can be chosen freely, as long as it is compatible with the "Total length" field[6].

In summary, with the given assumptions, there are roughly 16 bits beyond the control of the attacker.

A note on encrypting fragments. Above, we assume that the IPsec implementation being attacked encrypts fragments in tunnel mode; this assumptions holds for FreeS/WAN 1.91 which was used in

[5]Because we did not want the IPsec packets to be fragmented, the maximum total length is 1500 minus IPsec overhead; the overhead consists of SPI (4 octets), sequence number (4 octets), the initialization vector (8 octets), padding, padding length and "next header" field inside ESP (2 octets, at minimum), and the ESP authenticator (we are assuming 12 octets). The resulting maximum total length is 1470.

[6]The combination of "Total Length" and "Fragment Offset" (converted to octets) must not exceed 65536, the size of the maximum IPv4 packet.

our tests. Some IPsec implementations first reassemble all the fragments, then encrypt, and finally fragment the resulting packet again. Attacking such implementations is more difficult because the fragment related fields cannot be freely controlled; the attack is harder (roughly) by a constant of 2^{13} (the size of the "Fragment Offset" field).

6 Forcing the Victim Host to Encrypt the Attack Packet

Forcing the victim to encrypt and send the attack packet is the most difficult part of the attack. Satisfying this precondition is entirely different in tunnel and transport mode, and is also sensitive to the network topology.

6.1 Tunnel mode

The attacker may route attack packets through the tunnel if the attacker has access to the network behind the tunnel endpoint. The common objection to this approach is that the attacker must have access to the trusted side of the tunnel, and thus there is no point in attempting this attack anyway. This argument is, however, not always valid.

The trusted network might be a large routed network. The victim and the attacker may reside in entirely different parts of the network, and the attacker might not be otherwise able to observe the victim's traffic.

Also, if the network setup allows access to an external network (through a NAT or a firewall), the attacker may be able to carry out the attack without access to the internal network. For instance, if NATted Internet access is allowed, any NAT mapping (created by the victim host by accessing the Internet) can be exploited by the attacker. The attacker can then simply forge IPv4 packets and send them using the NATted address information. If there is no NAT or a stateful firewall, the attacker can send attack packets directly without waiting for the victim to initiate an external access.

Note, however, that in such attacks, the attacker is only able to attack IPsec traffic directed towards the victim host – not traffic coming from the victim host. This is fortunate, because the traffic sent by the victim (e.g. passwords) is more interesting than the traffic flowing in the reverse direction.

Ciphers with larger than 64-bit block size force the attacker to control more bits of the IPv4 header. For instance, a cipher with a 128-bit block size (such as AES) force the attacker to control the source IPv4 address, the TTL, and the checksum fields, among other fields.

6.2 Transport mode

Applications running on the victim host may provide a method for sending a chosen plaintext block. Such applications could include e.g. streaming protocols, FTP, and SMTP. The victim may, for instance, have separate IPsec transport connections to both the attacker and a third host. The application on the victim host may "route" data from one connection to another at the application layer (consider, for instance, e-mail).

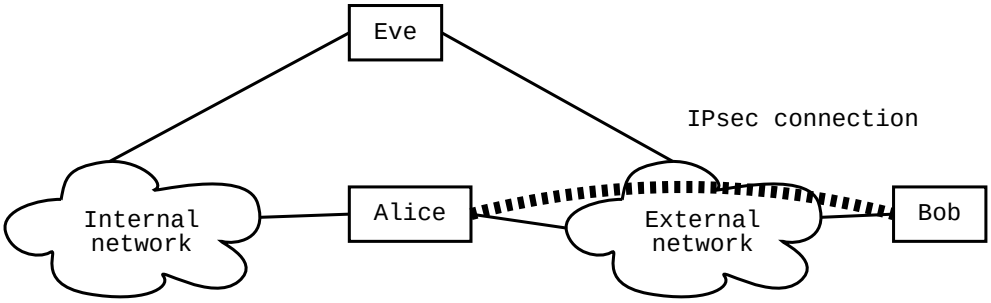

Figure 5: Attack setup

If the victim host is a multi-user machine, one user may be sending data using one application while the attacker is causing attack packets to be sent by using another. This approach may work for tunnel mode as well.

7 Practical Attacks

7.1 Overview

We used a setup of three hosts, as shown in Figure 5.

Alice and Bob use Linux FreeS/WAN, while Eve uses Linux with custom software able to sniff (encrypted) packets exchanged by Alice and Bob, and route forged packets through Alice in a tunnel mode attack. We used ESP with 3DES and HMAC-SHA1 for encryption and authentication, respectively.

7.2 Attack 1: Confirming the Vulnerability

We first verified the vulnerability manually. We set up an IPsec transport connection between Alice and Bob, and sent an ordinary UDP packet with known plaintext from Alice to Bob. We then captured the IPsec-processed ESP packet, and extracted the encrypted victim block and the initialization vector used to encrypt it. Based on these two blocks, the known plaintext, and an IV prediction, we computed the attack block that Alice would need to be forced to encrypt.

We used a Java program to force Alice to encrypt the attack packet. The program used standard Java socket API calls to create a UDP data payload which forced the UDP header “checksum” and “length” fields to desired values. The source port was controlled by binding a datagram socket to the desired port (this failed in a small minority of cases because of insufficient privileges). The destination port was controlled by simply sending the datagram to the desired port.

7.3 Attack 2: Web Page Access in Transport Mode

The second attack was against suspected plaintext in transport mode. Alice accesses a web page using HTTP protected by IPsec transport mode. Eve suspects that Alice is accessing a certain URI on the web server. By using the attack described in this paper, Eve verifies her guess.

The attack was a success [13]. The URI of the web page being accessed was uncovered with three attack packets, confirming a guess of 24 plaintext characters.

Attacking a multi-user host where several users share an IPsec security association is not new. Bellovin describes a similar attack against ESP without authentication in [12]. However, the attack described here works regardless of ESP or AH authentication.

Note that this attack is easy to carry out in tunnel mode as well.

7.4 Attack 3: A Simulated Telnet Login in Tunnel Mode

In the third attack, Alice logs in to Bob using a telnet-like protocol, protected by IPsec tunnel mode. The login consists of single-character TCP segments[7]. Bob captures the encrypted packets and determines each character in turn.

We first sent a single test character through the IPsec tunnel connection, and tried the attack first using a few false guesses and then using the correct guess. This attack worked, and we were able to correctly verify desired characters of the login traffic.

Our intention was then to crack a single unknown character to obtain a practical effort estimate. This attack failed because the victim rekeyed spontaneously during the attack; we ran out of time and could not continue the attack further. Note that rekeying does not really prevent the attack – we could have continued after waiting for the victim to login again.

Even though the second part of the attack failed, the first part indicates that the vulnerability can be exploited, as long as the attacker deals with rekeying events. We were able to obtain an effort estimate in our network from the second part of the attack: the average rate of attack packets was about 1012 packets/second (0.987 milliseconds between packets). At this rate, trying a single guess requires approximately 25.6 seconds, on average. To uncover an entire eight character password would require an average of 5.4 hours. The probability of the attack block meeting the validity constraints for IPv4 headers was slightly better than 2^{-16}. [14]

Note that our network was very simple, and thus had extremely low latency. The time required for the attack increases linearly with the latency (unless the attack is able to predict more than one IV at a time).

8 Preventing the Attack

In his mail to the IPsec mailing list, Fluhrer suggested the following change to the AES CBC draft:

[7] In real environments, the username and/or password characters might be combined into larger TCP segments.

> The IV field MUST be the same size as the block size of the cipher algorithm being used. The IV MUST be chosen at random, and MUST be unpredictable.

Of course, the same requirement suffices to prevent the attack regardless of which cipher is used. Any algorithm for choosing IVs that can be predicted by the attacker opens up the same vulnerability; IV chaining is simply one vulnerable algorithm.

Changing the ESP cipher has no effect on the attack, unless the cipher block size changes. In particular, the adaptive chosen plaintext attack described by Fluhrer is feasible even if the cipher itself resists such attacks. A larger block size makes the attack harder because there is more plaintext data to guess, and more bits to control in the attack packet.

Rekeying slows down the attacker, because an iteration of guesses against a given plaintext block cannot be continued if the key changes. The attacker can, however, wait for the victim to resend the interesting plaintext and continue the attack (of course, the attacker must somehow guess which encrypted block is a resend of the previous plaintext). Note that the attacker does not need to restart the attack from scratch; previously eliminated guesses do not have to be reconfirmed. Thus, rekeying does not protect against the attack fully.

If IV chaining is used, sending high speed data through the IPsec connection makes the attack very difficult: once the attacker has predicted the initialization vector, it may have already been used before the attacker has time to exploit the prediction. However, should the high speed traffic stop, the attacker could mount the attack immediately against any previous plaintext block, even blocks that were sent when the high speed traffic was still being sent.

Authentication (ESP or AH) does not prevent the attack, since packets are not directly modified by the attacker but the attacker is causing the victim host to encrypt the attack blocks.

Note that the attacker does not get information that helps in breaking the encryption key, and consequently a successful attack will simply reveal the contents of one plaintext block. The attacker gains no advantage for later attacks. Knowledge of a verified plaintext-ciphertext block pair may be useful information, although such pairs are easy to guess (with a high degree of certainty) anyway.

9 Conclusions

If initialization vectors are chosen in a predictable manner in ESP, an adaptive chosen plaintext vulnerability opens up. The preconditions of the attack are restrictive, and the vulnerability is thus difficult, but probably not impossible, to exploit in practice.

We demonstrated that the vulnerability can be exploited to guess single characters of TCP connections, and to verify suspected plaintext blocks, such as URIs being accessed. ESP or AH authentication does not prevent the attack.

If the victim chooses initialization vectors using an unpredictable algorithm, the attack is prevented.

References

[1] Jon Postel. *Internet Protocol*, Request For Comments 760, January 1980.

[2] Jon Postel. *User Datagram Protocol*, Request For Comments 768, August 1980.

[3] Jon Postel. *Transmission Control Protocol*, Request For Comments 793, September 1981.

[4] Randall Atkinson and Stephen Kent. *Security Architecture for IP*, Request For Comments 2401, November 1998.

[5] Randall Atkinson and Stephen Kent. *IP Encapsulating Security Payload (ESP)*, Request For Comments 2406, November 1998.

[6] Dave Carrel and Dan Harkins. *The Internet Key Exchange (IKE)*, Request For Comments 2409, November 1998.

[7] Naganand Doraswamy and Cheryl Madson. *The ESP DES-CBC Cipher Algorithm*, Request For Comments 2405, November 1998.

[8] Rob Adams and Roy Pereira. *ESP CBC-Mode Cipher Algorithms*, Request For Comments 2451, November 1998.

[9] Serge Vaudenay. *CBC Padding: Security Flaws in SSL*, IPSEC, WTLS, . . .

[10] Phillip Rogaway. *Problems with Proposed IP Cryptography*, April 1995, Internet-Draft (expired), see http://wwwcsif.cs.ucdavis.edu/∼rogaway/papers/comments.html.

[11] The IPsec mailing list, see http://www.ietf.org/ for more information. Archive currently at ftp://ftp.tis.com/pub/lists/ipsec.

[12] Steven M. Bellovin *Problem Areas for the IP Security Protocols*, Proceedings of the Sixth Usenix UNIX Security Symposium, July 1996.

[13] Antti Nuopponen and Sami Vaarala. *An Attack against IPsec Transport Mode HTTP Access*, March 2002.

[14] Antti Nuopponen and Sami Vaarala. *An Attack against Single Character TCP Segments Protected by IPsec Tunnel Mode*, March 2002.