Global Information Assurance Certification Paper

SANS Ottawa

Man-in-the-middle attack against security protocols
Hacker Exploits

Presented by

Frédéric Massicotte

Exploit details

Name:	Man-in-the-middle attack against the initiator of Otway-Rees Key Exchange Protocol.
Variants:	Man-in-the-middle attack against the two parties of Otway-Rees Key Exchange Protocol.
Operating System:	All operating systems with which the Otway-Rees Key Exchange Protocol specification may be implemented, because it concerns a specification flaw in the key exchange protocol.
Protocols/Services:	Otway-Rees Key Exchange Protocol.
Brief Description:	This vulnerability allows a hacker to find the session key distributed by a key exchange protocol. This is a man-in-the-middle type of attack. He can exploit this vulnerability without launching a brute-force attack on encrypted messages or breaking into any computer. The hacker simply manipulates protocol messages and uses an impersonation tool such as *Hunt*.

Introduction

Since the coming of the Internet in the 1980s and 90s, information has often been transmitted unscrambled and unprotected over the Internet or various computer networks during remote access, electronic transactions, etc. The problem resides in the fact that the Internet is not a proprietary network but, instead, consists of thousands of independent networks. No one has complete control over the route that information takes on the way to destination. With the Internet's current design, information security has become a priority. Given the billions of dollars exchanged over the Internet through e-commerce and e-business transactions and the real threat posed by the interception of secret information, we need tools to make electronic transactions over the Internet secure.

To meet this need, the Internet community has designed new types of communication protocols that function above normal communication and transmission protocols, such as TCP/IP, etc. These are known as security or cryptography protocols. SSL (Secure Socket Layer) and SET (Secure Electronic Protocol) are good examples of security protocols currently used in e-commerce to make vender-purchaser transactions secure. People want assurance that their Internet transactions are secure and that they will not lose important information, such as credit card numbers, passwords, etc. to hackers. There are several types of cryptography protocols, such as authentication protocols, e-commerce protocols, key distribution protocols, etc. In this report, we are specifically interested in vulnerability in key exchange protocols. These protocols are used to

Secret info

distribute a session key to two or more principals, to allow them to communicate in a secure fashion by encrypting future information exchanges. Virtual Private Networks are good examples of the use of key exchange protocols. VPNs generally use IKE, a key exchange protocol, for two entities to exchange a secret session key. With this key, they are able to communicate in a secure fashion by encrypting information transmitted in a hostile environment such as the Internet. Because a hacker does not have the session key to encrypt information, it becomes very difficult for him to obtain secret information by launching a man-in-the-middle attack and monitoring network traffic with a sniffer.

Secret info Secret info Secret info

Protocol description

No one has complete control over the Internet, and it is almost impossible to prevent a hacker from launching a man-in-the-middle attack or retrieving information packets from the network, as **figure 1** indicates. One of the solutions to this problem is cryptography and cryptographic algorithms. If each of the principals has the right key, they can make their transaction secure by using cryptographic algorithms, as indicated in **figure 2.** This key is generally called a session key, since it is only used once for a specific session. This key is only good for one session, after which, if the two entities wish to communicate again, they must obtain a new key for a new session. However, one problem remains—how to distribute a session key to the two entities in a secure fashion. This is a difficult problem, which may be solved by using a key exchange protocol.

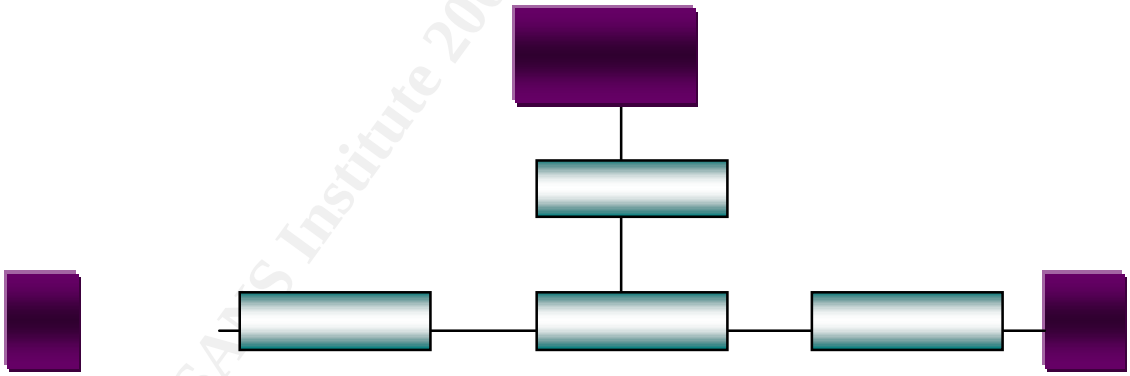

Figure 1 Message transmitted in clear text format

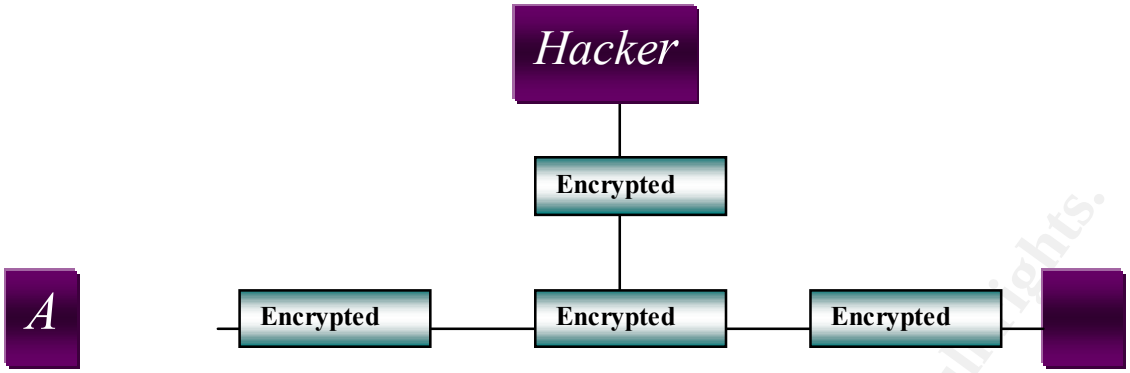

Figure 2 Message transmitted in encrypted format

Basic notions

Before explaining the vulnerabilities of key exchange protocols, we must mention the syntax and symbols used to define security protocol specifications in general. The following symbols are used to specify security protocols:

Table 1: Symbols used in security protocols

A	***A's name***
B	***B's name***
S	***Key Sever***
k_{as}	***symetric key shared by A and S***
k_{bs}	***symetric key shared by B and S***
k_{ab}	***symetric session key shared by A and B***
I	***session number***
N_a	***random number generated by A***
N_b	***random number generated by B***

The aforementioned symbols are very simple. Upper case letters generally represent computer entities, called principals. We will discuss these in greater depth in the discussion on protocol specifications.

Otway-Rees Key Exchange Protocol Specification

When deploying a key exchange protocol in a wide area network, security people want to ensure that there are no vulnerabilities in protocol implementation and especially specification design, to make it difficult for a hacker to compromise the security of information transmitted over the network. When selecting a secure key exchange protocol, we must expect that the protocol key will never be sent unscrambled in a hostile environment outside our control, and that, at the end of the protocol, the entities to receive the session key do actually receive it in a trouble-free manner, without hackers intercepting it. However, as we will see, without decrypting any message or attacking or breaking into any computer, hackers may manipulate information to obtain the session key without either of the entities detecting the ruse. To illustrate this problem, we are

going to study the Otway-Rees Key Exchange Protocol. The complete protocol specification can be found in Bruce Scheier's book *Applied Cryptography* [1].

The Otway-Rees Protocol makes it possible to distribute a session key k_{ab} created by the trusted server ***S*** to two principals ***A*** and ***B.*** This key will encrypt the information transmitted between these two principals. Sharing this key and the cryptographic algorithms creates a VPN-type communication tunnel between the two principals.

As well, this protocol authenticates the principals to ensure the integrity of messages and that the key has been correctly distributed to the correct principals. This prevents the key from falling into the wrong hands, such as those of a hacker who is hijacking a session or conducting a man-in-the-middle attack. The Otway-Rees Key Exchange Protocol is specified as follows:

Table 2: Otway-Rees Key Exchange Protocol

Message 1	$A \rightarrow B: I,A,B \{ N_a , I, A, B \}_{kas}$
Message 2	$B \rightarrow S: I,A,B \{ N_a , I, A, B \}_{kas} , \{ N_b , I , A, B \}_{kbs}$
Message 3	$S \rightarrow B: I, \{ N_a , k_{ab} \}_{kas} , \{ N_b, k_{ab} \}_{kbs}$
Message 4	$B \rightarrow A: I, \{ N_a , k_{ab} \}_{kas}$

Thus, as we can see, at the end of the protocol, the key k_{ab} is received by ***A*** and ***B***, who are now ready to exchange confidential or secret information.

The message in the form $\{ m \}_k$ symbolizes that message ***m*** has been encrypted with ***k*** using a symmetrical cryptographic algorithm. Before starting the protocol, each of the principals has certain initial knowledge. Keys k_{as} and k_{bs} are permanent keys given to ***A*** and ***B*** respectively, as personal keys. They must share these with the server to communicate with it. With these permanent keys, the principals are able to obtain a session key from the server.

The cryptography protocol may be described in more detailed fashion as follows:

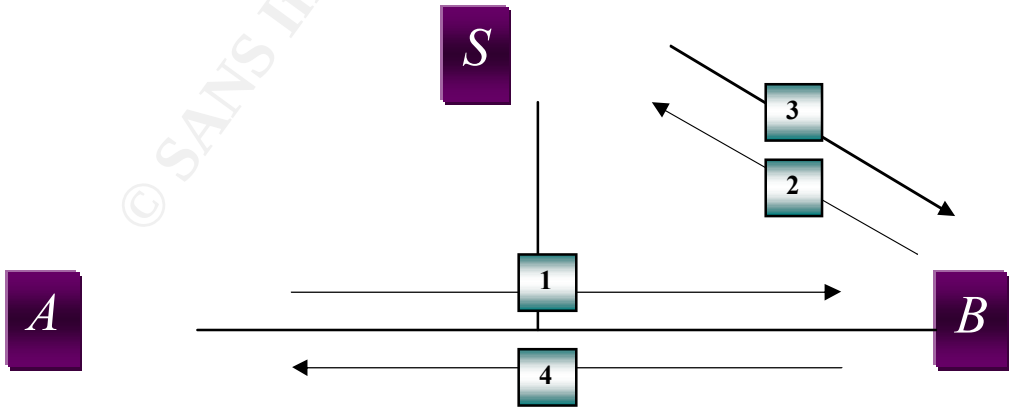

Figure 3 Otway-Rees Key Exchange Protocol

1. ***A*** sends ***B*** the protocol session number, his identity, the identity of the principal with whom he wishes to communicate, and a message encrypted with the key k_{as}.
2. ***B*** receives ***A's*** message and adds his own message encrypted with the key k_{bs} before sending it to the trusted server ***S***.
3. ***S*** receives the message and is able to retrieve the session number, the random number from ***A,*** N_a **,** using his shared key k_{as}, the random number from ***B***, N_b with the other shared key k_{bs}, and generates the session key k_{ab}. With this information, he is able to generate message 3 and sends it to ***B***.
4. The principal in question receives message 3, removes the last encrypted part with his shared key, decrypts this sub-message with his key k_{bs}, retrieves the session key k_{ab}, and sends the remaining part of the message to ***A***. In this way, ***A*** is also able to retrieve the session key k_{ab}, based on the last part of message 4, by using his shared key k_{as}, and the two principals are able to start communicating.

In addition to using session numbers to ensure message authentication and integrity, this key exchange protocol uses random numbers like stamps to identify sessions. The random number N_a can only be known by ***A*** and the trusted server ***S***, since it is always encrypted by the key k_{as} when it is transmitted over the network. Therefore, when ***A*** receives the key k_{ab} at the end of the protocol, he can be sure that the session key is genuine and that it was, in fact, generated by the server, by verifying if the random number received in the last message is the same one generated in the first message. In this protocol, we can use the same logic with respect to the random number N_b for principal ***B***.

In addition, we can see that a session key is always encrypted when it is transmitted between principals over a computer network. Indeed, the session key is always encrypted by the keys k_{as} and k_{bs} and the only parties with these keys are the principals ***A, B***, and ***S***. Therefore, a hacker cannot retrieve the session key in any way if the cryptographic algorithm is perfect and there are no security vulnerabilities in the protocol implementation.

If a hacker impersonates principal ***A*** to principal ***B*** and impersonates ***B*** without ***A*** realizing it, by launching a classic man-in-the-middle attack with *Hunt* type software [2] or controlling a router through which the information is transmitted, he will able to deceive the principals about his identify. However, he will not be able to retrieve the key, as shown in the following figure:

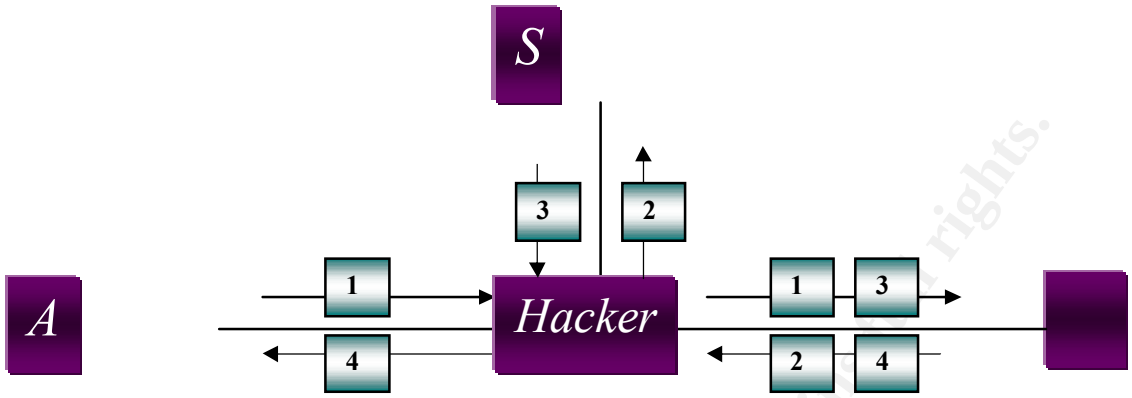

Figure 4 Simple man-in-the-middle attack against Otway-Rees Key Exchange Protocol

How the exploit works

Most of the time, hackers exploit security vulnerabilities in software implementation or find insecure systems to attack. Most of the time, even the most sophisticated systems maintained by the best security technicians may have security vulnerabilities. Given their current level of complexity, it is almost impossible for programs to be completely free of vulnerabilities or bugs. This is what we must expect with security software or protocols. However, most security vulnerabilities discovered in recent years and posted on Internet sites such as www.securityfocus.com and www.ntbugtraq.com or on hacker sites are almost exclusively implementation vulnerabilities such as overflow buffers, program errors, etc.

We often forget that some vulnerabilities may only be discernable at the specification and design level. Even today, it may be difficult to develop cryptography protocols without vulnerabilities. Otway-Rees Key Exchange Protocol does, in fact, have a specification vulnerability that allows a hacker to steal the session key.

In particular, if a hacker wants to steal the session key, he must be able to determine the content of messages 3 and 4. He may also attack the server or one of the principals to retrieve the permanent keys or attempt a brute-force attack on the encrypted messages that he retrieved from the computer network. However, this may be more complicated than exploiting the protocol vulnerability. Indeed, the hacker may retrieve the key by simply manipulating information. He has to impersonate principal ***B*** with *Hunt* type software. When ***A*** is ready to start a protocol session, the hacker launches the attack as follows:

Table 3: Attack against the initiator of Otway-Rees Key Exchange Protocol	
Message 1	$A \rightarrow I(B)$: $I,A,B \{ N_a , I, A, B \}_{kas}$
Message 4	$I(B) \rightarrow A$: $I, \{ N_b , I, A, B \}_{kas}$

When ***A*** wants to start a session with ***B,*** hacker ***I(B)*** impersonates ***B*** with *Hunt* type software. By placing a sniffer at the right place on the computer network, he retrieves the first message and the session number and the encrypted part of the message, concatenates all the components, and sends the result to principal ***A***. This party retrieves the message, verifies the session number, decrypts the encrypted message with his permanent key, verifies if he has correctly received the random number that he sent in message 1, and concludes that the second part of the decrypted message is the session key. Therefore, the session key for this session is the message ***I,A,B***. Since this message is sent unscrambled over the computer network, the hacker may intercept it and thereby steal the secret key. Indeed, principal ***A*** does not know the session key before receiving message 4. Therefore, ***A*** will accept any bit string which is the same length as the session key and encrypted with the right random number and the key k_{as}.

To carry out this attack, a hacker only needs to know the protocol and how it behaves. He does not need to carry out steps 2 and 3 of the protocol. In fact, there is no way for ***A*** to know that these two steps in the protocol have not been carried out.

Thus, the hacker does not need to know the permanent keys and does not have to encrypt or decrypt any information at all. By simply manipulating information, he can find the protocol session key and start exchanging secret information with principal ***A***, without anyone suspecting that ***A*** is in the process of exchanging secret information with a hacker.

Diagram

Here is how one of the attacks that may be carried out on a network:

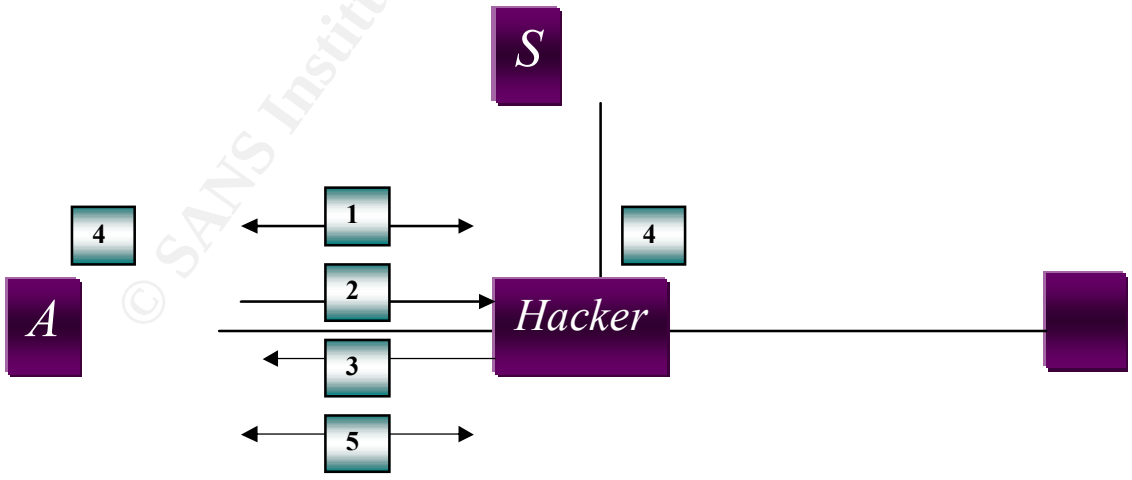

Figure 5 Attack against the initiator of the Otway-Rees Key Exchange Protocol

1. When ***A*** wants to establish a connection with ***B*** for this protocol, the hacker manages to gain control over an entity through which information is transmitted and uses *Hunt* type software to impersonate ***B***.
2. When the connection is established, ***A*** starts a protocol session and sends the hacker impersonating ***B*** the message: $I,A,B\ \{ N_a , I, A, B \}_{kas}$.
3. The hacker receives the message, removes ***A,B***, generates the message: $I,\ \{ N_a, I, A, B \}_{kas}$, and sends it to ***A***.
4. ***A*** retrieves the session key ***I,A,B*** and the hacker does the same with the information retrieved from the network.
5. Principal ***A*** sends encrypted messages with the session key. The hacker now has complete control over the connection, as if it had never been encrypted with a session key. To him, the information on the network appears to be unscrambled.

For more information on session hijacking and man-in-the-middle attacks, refer to [8, 9,10,11,4].

Variants description

Otway-Rees has several variants of this vulnerability. There are, in fact, several ways of actually carrying out this attack on a real computer network. In addition, the hacker has several options concerning which principal's identity he may assume in order to steal the session key and compromise the security of information transfer. In fact, even if the hacker does not control the connection between ***A*** and ***B,*** he will nevertheless be able to carry out an attack if he is able to monitor traffic between ***A*** and ***B*** and control traffic between ***B*** and the server. This variant of the attack is shown in **table 4**:

Table 4: Attack against the two parties of Otway-Rees Key Exchange Protocol

Message 1	$A \rightarrow B: I,A,B\ \{ N_a , I, A, B \}_{kas}$
Message 2	$B \rightarrow I(S): I,A,B\ \{ N_a , I, A, B \}_{kas} , \{ N_b , I , A, B \}_{kbs}$
Message 3	$I(S) \rightarrow B: I, \{ N_a , I, A, B \}_{kas} , \{ N_b , I , A, B \}_{kbs}$
Message 4	$B \rightarrow A: I, \{ N_a , I,A,B \}_{kas}$

In this attack, the hacker lets principals ***A*** and ***B*** establish a connection and exchange the first message in the protocol. Then ***B*** must establish a connection with the server. At this point, the hacker, who has managed to gain control over the information moving between ***B*** and ***S,*** uses *Hunt* type impersonation software and, by impersonating the server, establishes a connection with principal ***B***, who sends him message 2. After receiving it, the hacker returns the same message after removing the message ***A,B***. Therefore, based on the protocol, principal ***B*** takes the second encrypted message and finds the session key ***I,A,B***, as ***A*** did in the first attack. Then, according to the protocol specification, ***B*** sends message 4 to principal ***A,*** which is, in fact, message 3 without the part encrypted with ***B's*** permanent key (which was removed before being sent). At this point, ***B*** just follows the protocol specification and has no way of determining whether the session key that he is going to transmit to principal ***A*** is in fact the right key generated

by the server. ***A*** then retrieves the message and also finds the key ***I,A,B***, like the key generated by the server. ***A*** and ***B*** can then send each other information encrypted with this new session key. However, if the hacker is able to sniff the information moving between ***A*** and ***B,*** he will be able to decrypt the information in its entirety without either principal realizing it.

Diagram

This is one of the methods that a hacker may use on an actual computer network:

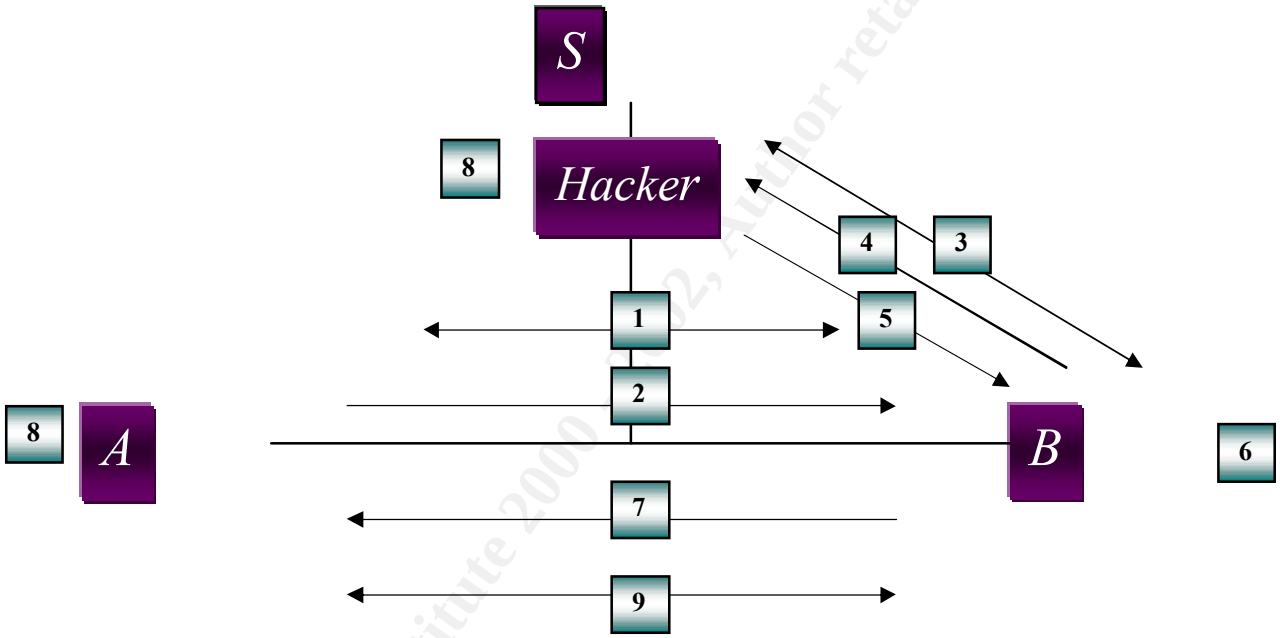

Figure 6 Attack against the two parties of the Otway-Rees Key Exchange Protocol.

1. Principal ***A*** establishes a connection with ***B*** for this protocol.
2. Principal ***A*** sends the message: ***I,A,B,*** $\{ N_a , I, A, B \}_{kas}$ to ***B***.
3. When ***B*** wants to establish the connection with the server, the hacker manages to gain control over an entity through which information is being transmitted and uses *Hunt* type software to assume the identity of the server.
4. Principal ***B*** sends the message ***I,A,B,*** $\{ N_a , I , A, B \}_{kas}$, $\{ N_b , I , A, B \}_{kbs}$ to the server impersonated by the hacker.
5. The hacker returns almost the same message, without the sub-message ***A,B***.
6. Principal ***B*** finds the session key, which is the same as the message ***I,A,B.***
7. Principal ***B*** sends the message ***I,*** $\{ N_a , I , A, B \}_{kas}$ to principal ***A*** to end the protocol.
8. He finds the session key ***I,A,B*** and the hacker, with the message that ***B*** sent him, easily finds the session key.

9. ***A*** and ***B*** start exchanging secret encrypted information with the session key. Using his sniffer placed between the two principals, the hacker is able to decrypt all the information as if it were being transmitted on the network unscrambled.

How to use it

To my knowledge, no software exists to carry out this type of attack and exploit this type of vulnerability in this protocol or other security protocols. With respect to attacks, it is easy for a knowledgeable hacker who controls the router through which information between ***A*** and ***B*** is transmitted to control the information using a filter and thereby retrieve the session key.

There might be no software to carry out this type of attack, but there is a tool, designed at Laval University in Québec City, that was developed to perform automatic verification of security protocols. We only have to provide this tool with a protocol specification similar to the format of the Otway-Rees Protocol specification. We will discuss how this tool works in the next section.

Signature of the attack

Network point of view

The signature of this attack is relatively easy for a configurable intrusion detection system to identify. Even if he successfully steals the session key, in all these types of attacks, the hacker is only able to generate a single session key, the session key ***I,A,B***. Therefore, if the network intrusion detection system is able to follow the details of an Otway-Rees Key Exchange Protocol session and sees that the message $\{ N_a , I , A, B \}_{kas}$ sent with message 1 is in fact the same message contained in the second part of messages 3 and 4 or that message $\{ N_b , I , A, B \}_{kbs}$ sent in message 2 is the same as the third part of message 3, it can determine that there is definitely a problem, since the probability that the bit string representing the session key would be the same as that representing message ***I,A,B*** is very low. In fact, there is almost no chance of this situation occurring, which means the rate of false positives for this type of attack is very low. As well, the server should not be permitted to generate this type of key, thus enhancing the signature detection activity. In short, a network-type detection intrusion system can, without decrypting any information at all, detect this type of attack.

The intrusion detection system may also be able to verify if all protocol steps have been carried out and, if not, warn that there is a problem.

So an intrusion detection system that can monitor a session of this protocol has the possibility to keep in memory, for a session ***S***, all the messages that have been transmitted for this session between the principals. By comparing some part of message 1 and some part of message 4 one will be able to detect very easily this attack. The rules

that may be added to an intrusion detection system able to follow a session of the protocol may be written in pseudo code, as follows:

IDS rules to prevent the man-in-the-middle attack against Otway-Rees

```
If (S.Message1.kas) = (S.Message3.kas) or
   (S.Message1.kas) = (S.Message4.kas)
        alarm administrator "Stolen Key"
Else if (S.Message2.kbs) = (S.Message3.kbs)
        alarm administrator "Stolen Key"
```

In this case, the specification ***S.MessageN.K*** specifies the sub-message encrypted with the key ***K*** in the message ***N*** from the session ***S***. Even if these messages are encrypted, if the part of the message 1 encrypted with the permanent key of ***A*** is the same as the part of message 4 encrypted with the same key, this means that the key is the message ***I,A,B*** for the current session.

Host point of view

From the host's point of view, the protocol may include a mechanism to protect against this attack. Indeed, if one of the principals notices that the session key is in fact the same key, the same message as ***I,A,B***, from a binary point of view, he can conclude that a hacker is attempting to steal the session key if, in addition, the server is not able to generate this key. The pseudo code of it goes like this :

Program adjustment to prevent the man-in-the-middle attack against Otway-Rees

```
If sessionKey = sessionNumber + firstPrincipal.name + secondPrincipal.name
        drop connection
        alarm user "Stolen Key"
```

How to protect against Otway-Rees Key Exchange Protocol Exploit

Implementation level

This attack has a very specific and precise signature, irrespective of how the hacker orchestrates the attack. The fact that, during a specific protocol session, the session key is the message ***I,A,B,*** from a binary point of view, clearly identifies this

attack signature. However the hacker proceeds, the only key that he is able to steal as session key is the message ***I,A,B,*** which he retrieves from the network.

Therefore, as mentioned in the section on the attack signature, one way to protect against this attack is to configure the network-oriented or host-oriented intrusion detection system to follow Otway-Rees Protocol sessions. Based on the messages exchanged, it must then verify if the session key is in fact the same, from a binary point of view, as the message consisting of the session number and the identity of the two principals.

As we saw in the section on the attack signature, the intrusion detection system is able to detect this attack even if the message is encrypted. From the host's point of view, it is also very easy for each of the principals, before validating the session key, to determine, from a binary point of view, if the key is in fact the message consisting of the session number, their identity, and the identity of the principals with whom they have initiated a protocol session.

Design level

From a software perspective, it would be possible to solve this problem by adding code to prevent this situation and to configure our intrusion detection systems to handle this, etc. However, it should be remembered that this vulnerability is related to the specification and it is at this level that we can find a better way of preventing groups of hacker from exploiting this vulnerability. Therefore, it is necessary to find ways of creating protocols that have no vulnerabilities from the specification point of view, or have verification tools to check them.

No tools yet exist to create custom security protocols with no specification vulnerabilities. However, some research is being conducted in this area but there is still work to accomplish.

Several tools for verifying protocols do exist. In particular, the LSFM group at the Computer Science Department of Laval University in Québec City has developed a tool to perform automatic verification of security protocols (reference for the tool designers is found in [4]). Supplied only with a protocol specification similar in format to the Otway-Rees Protocol specification, the tool is able to turn back all protocol attacks based on a specification vulnerability. The operation of this software is simple. The tool is given the protocol specification and the protocol's principal who will be attacked. After the option "Check Flaw" is chosen, the tool starts processing, as shown in the following figure:

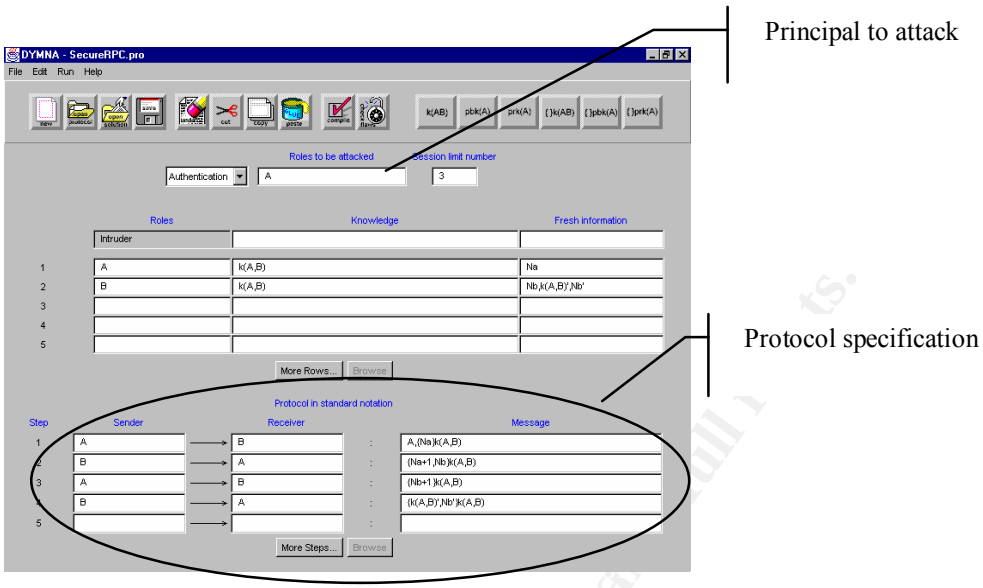

Figure 7 Graphical interface of the security protocols analyzer.

The software then reports all the specification-based attacks against this protocol that it detected. Thus, according to the attack, we can define the protocol's problems and weaknesses to make an adjustment to the specification, making the protocol more secure. It does not suggest solutions to a problem, but generally provides good information about the problem. The following graphic interface shows the attacks against the protocol:

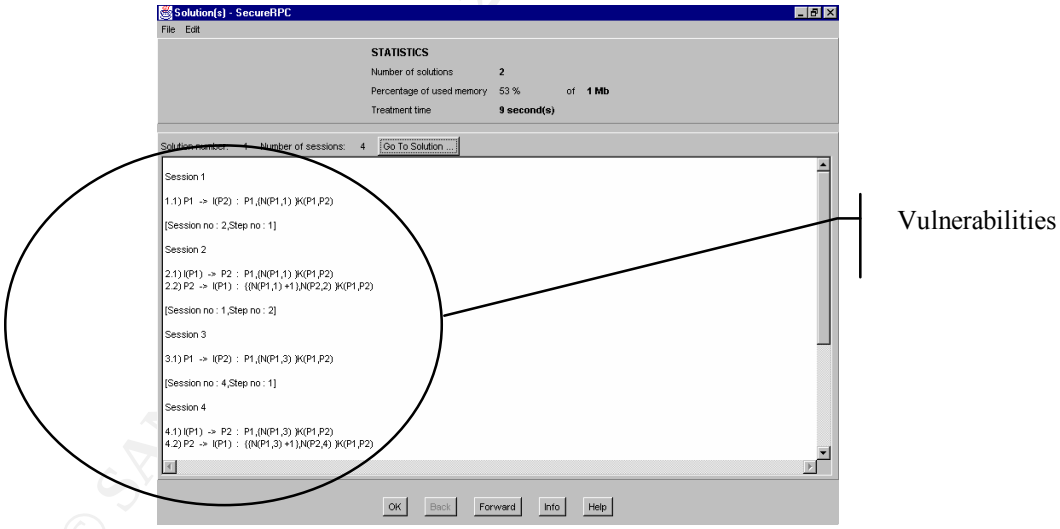

Figure 8 Graphical interface giving the solution to a user.

This type of software may be very useful in identifying the design vulnerabilities affecting current means of exchanging keys, performing authentication, etc.

The code for the tool that we just discussed is unfortunately not distributed. For more information, you may contact its authors at Laval University.

Source code

Given that, to my knowledge, no software exists that is able to threaten and attack this protocol, no source code is available. However, if a hacker controls an entity such as a router that is able to filter information and make decisions on messages, and if this entity is located on the link through which information is transmitted in an Otway-Rees Protocol, it would be easy to write a short program to carry out this attack. For the sake of concision, the pseudo code for the program specification allowing to reproduce the attack against the two parties of the key exchange protocol is not presented here. However, with the pseudo code presented here, it will be easy to understand how such a program would be developed. For the attack against the initiator of Otway-Rees Key Exchange Protocol, the program could contain the following instruction:

Pseudo code

```
get setup for the man-in-the-middle attack

if packet.protocol = Otway-Rees
        if packet.data = Otway-Rees.message1
                firstPrincipal = packet.data.firstPrincipal
                secondPrincipal = packet.data.secondPrincipal
                sessionNumber = packet.data.sessionNumber
                messagetoSend = packet.data – firstPrincipal – secondPrincipal
                sessionKey = sessionNumber + firstPrincipal + secondPrincipal
                Send messagetoSend to the firstPrincipal
                Send sessionKey to the hacker
Else if
        Don't touch it or send it to the right person
```

Additional information

There are many types of specification vulnerabilities in security protocols (e.g. key exchange protocols, authentication protocols, e-commerce protocols, etc.). A good summary and other attacks against protocols may be found in [3,5,6,7,12]. It should be remembered that a specification vulnerability implies that, even if implementation is perfect, there will be an implementation vulnerability. Effective security tools are essential in a world in which information is no longer a privilege but a necessity for tomorrow.

References

[1] B. Schneier: *Applied Cryptography*. Wiley, Second Edition, 1994.

[2] http://www.cri.ca/kra/index.html

[3] J. Clark. Attacking Authentication Protocols. *High Integrity Systems*, 1 (5) :465-474, March 1996

[4] http://www.ift.ulaval.ca/~lsfm/

[5] J. Clark and J. Jacob. On Security of Recent Protocols. Information Processing Letters, 156 (3): 151-155, November 1995.

[6] J. Clark and J. Jacob. A Survey of Authentication Protocols, November 1997. Unpublished article available at http://dcpu1.cs.york.ac.uk/simjeremy.

[7] J. Clark and J. Jacob. Draft: Implementation Dependencies and Assumptions in Authentication Protocols. Unpublished article available at http://dcpu1.cs.york.ac.uk/simjeremy, February 1997.

[8] S. Northcutt: Network intrusion Detection: An Analyst's Handbook. New Riders, First Edition, 1999.

[9] http://www.incrypt.com/mitma.html

[10] http://tcad.stanford.edu/archive/srp-dev/msg00068.html

[11] http://java.sun.com/security/ssl/API_users_guide.html

[12] F. Massicotte: Une théorie pour le type de faille dans les cryptoprotocoles, Master Thesis, Laval University, Québec, Canada 237 pages.

Last Updated: July 27th, 2017

Upcoming Training

Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Memphis SEC504	Memphis, TN	Aug 21, 2017 - Aug 26, 2017	Community SANS
Mentor Session AW - SEC504	Milwaukee, WI	Aug 23, 2017 - Sep 29, 2017	Mentor
Mentor Session AW - SEC504	New York, NY	Aug 24, 2017 - Sep 08, 2017	Mentor
Mentor Session - SEC504	Denver, CO	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS vLive - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	SEC504 - 201709,	Sep 05, 2017 - Oct 12, 2017	vLive
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, Ireland	Sep 11, 2017 - Sep 16, 2017	Live Event
Mentor AW - SEC504	Santa Clara, CA	Sep 11, 2017 - Sep 22, 2017	Mentor
Mentor Session - SEC504	Arlington, VA	Sep 20, 2017 - Nov 01, 2017	Mentor
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, Netherlands	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Columbia SEC504	Columbia, MD	Sep 25, 2017 - Sep 30, 2017	Community SANS
Mentor Session - SEC504	Boston, MA	Sep 26, 2017 - Nov 07, 2017	Mentor
Mentor Session AW - SEC504	Houston, TX	Oct 02, 2017 - Dec 11, 2017	Mentor
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Mentor Session - SEC504	Columbia, SC	Oct 03, 2017 - Nov 14, 2017	Mentor
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event
Community SANS Chicago SEC504	Chicago, IL	Oct 09, 2017 - Oct 14, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event