

Solution to Homework 1

CS259 - Security Analysis of Network Protocols

Winter 2008

Problem 1 Two invariants in the model specify the mutual authentication property we are interested in verifying. When translated to English, the “initiator correctly authenticated” invariant states that whenever a responder i completes a session, apparently with some initiator j , then it must be that j has completed a session, apparently with i . The meaning of the other invariant is analogous. For both invariants (independently), determine if they are satisfied in the provided Mur ϕ model of the NS protocol.

Solution In this problem you only had to run the supplied Mur ϕ code to check if the authentication invariants hold for the NS protocol. Invariant “responder correctly authenticated” holds while the invariant “initiator correctly authenticated” fails. Inspection of the trace leads to an attack that can be described using the arrows-and-messages diagram as follows:

$$\begin{array}{ll} A \rightarrow I : \{|A, N_A|\}_{K_I} & I(A) \rightarrow B : \{|A, N_A|\}_{K_B} \\ & B \rightarrow I(A) : \{|N_A, N_B|\}_{K_A} \\ I \rightarrow A : \{|N_A, N_B|\}_{K_A} & \\ A \rightarrow I : \{|N_B|\}_{K_I} & I(A) \rightarrow B : \{|N_B|\}_{K_B} \end{array}$$

Problem 2 In this problem we investigate whether NS or NSL can be used as key-exchange protocols. Specifically, we want to check if the nonces exchanged in the protocol remain secret.

1. Write down the Mur ϕ invariant “initiator secrecy” modelling the following property: if some initiator i completes a session with an honest responder then the intruder does not know the initiator’s nonce. Also, write down the analogous invariant “responder secrecy”.
2. For both NS and NSL protocol, test if these invariants are satisfied.

Solution Invariant modelling secrecy of the initiator’s nonce can be written down using Mur φ as follows:

```
invariant "initiator secrecy"
  forall i: InitiatorId do
    ini[i].state = I_COMMIT & ismember(ini[i].responder, ResponderId)
    ->
    forall j: IntruderId do
      int[j].nonces[i] = false
    end;
  end;
```

Invariant modelling secrecy of the responder’s nonce can be written down using Mur φ as follows:

```
invariant "responder secrecy"
  forall i: ResponderId do
    res[i].state = R_COMMIT & ismember(res[i].initiator, InitiatorId)
    ->
    forall j: IntruderId do
      int[j].nonces[i] = false
    end;
  end;
```

For the NS protocol invariant “responder secrecy” fails and the inspection of the trace leads to the same attack described in the previous problem. Invariant “initiator secrecy” holds for the NS protocol. For the NSL protocol both invariants hold.

Problem 3 Consider the following variant(s) of the Needham-Schroeder(-Lowe) protocol(s) where the last message is sent unencrypted:

$A \rightarrow B : \{ A, N_A \}_{K_B}$	$A \rightarrow B : \{ A, N_A \}_{K_B}$
$B \rightarrow A : \{ N_A, N_B \}_{K_A}$	$B \rightarrow A : \{ B, N_A, N_B \}_{K_A}$
$A \rightarrow B : N_B$	$A \rightarrow B : N_B$
NS Variant	NSL Variant

1. Modify the existing Mur φ code to model the variant protocols. Check if the initiator and responder authentication properties hold. If not, describe the attack(s) found.
2. As you can see, the secrecy of the nonce N_B no longer holds at the end of the protocol. Confirm if Mur φ comes up with the expected attack on secrecy in both protocols. What about the secrecy of N_A ?

3. Though the secrecy of N_B does not hold at the end, it might hold at intermediate points. Check and describe upto what state, for both initiator and responder, there is no attack found on secrecy of N_B .

Solution There are several ways of modifying the existing code to do this. One simple way is to let the attacker learn the nonce if the message is of type M_Nonce irrespective of the key used and let the receiver be receptive to a message of type M_Nonce in the R_WAIT state if the nonce is it's own, ignoring the key.

For the NS Variant protocol, initiator authentication fails, while no attack is found on responder authentication. The attack on initiator authentication is analogous to Lowe's attack on NS:

$$\begin{array}{ll}
A \rightarrow I : \{|A, N_A|\}_{K_I} & I(A) \rightarrow B : \{|A, N_A|\}_{K_B} \\
& B \rightarrow I(A) : \{|N_A, N_B|\}_{K_A} \\
I \rightarrow A : \{|N_A, N_B|\}_{K_A} & \\
A \rightarrow I : N_B & \\
& I(A) \rightarrow B : N_B
\end{array}$$

For the NSL Variant protocol, no attack is found on any of the authentication properties.

The trivial responder secrecy attack is found by Mur φ :

$$\begin{array}{ll}
A \rightarrow I \rightarrow B : \{|A, N_A|\}_{K_B} & A \rightarrow I \rightarrow B : \{|A, N_A|\}_{K_B} \\
B \rightarrow I \rightarrow A : \{|N_A, N_B|\}_{K_A} & B \rightarrow I \rightarrow A : \{|B, N_A, N_B|\}_{K_A} \\
A \rightarrow I \rightarrow B : N_B & A \rightarrow I \rightarrow B : N_B \\
(I \text{ learns } N_B) & (I \text{ learns } N_B)
\end{array}$$

NS Variant attack

NSL Variant attack

No attack is found on the secrecy of N_A for both the NS and NSL Variant protocols.

From the initiator's point of view, if it is interacting with an honest responder, no attack is found on the secrecy of N_B till the L_WAIT state, inclusive. From the responder's point of view, if it is interacting with an honest initiator, no attack is found on the secrecy of N_B till the R_SLEEP state, inclusive. This is for both the NS Variant and NSL Variant protocols. Although this looks "obvious", it is by no means trivial to actually *prove* that secrecy of N_B holds till the L_WAIT state. In fact, it takes a fair bit of technical machinery to achieve this. The complexity arises because an arbitrary number of concurrent sessions of the protocol might be going on in the network with each agent possibly

performing multiple roles. Remember how “obviously” secure the original NS protocol looked, before the “Lowe” attack came out! For the responder’s point of view, N_B is not even generated till after the R_SLEEP state - so secrecy is easy to deduce. After these states attacks are found along similar lines as the previous paragraph.