CS259 - Security Analysis of Network Protocols

Winter Quarter, 2006
Solutions for Homework #1

January 22, 2006

Problem 1

In this problem you only had to run the supplied Murφ 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:

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

Problem 2

Invariant modelling secrecy of the initiator's nonce can be written down using Murφ 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 Murφ 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

In the first part of the problem, you are required to modify the model so that initiators only initiate conversation with honest responders. The modification involves changing the guard of the first rule for the intitiator as follows:

```
-- initiator i starts protocol with responder or intruder j (step 3)
ruleset i: InitiatorId do
  ruleset j: AgentId do
    rule 20 "initiator starts protocol (step 3)"

      ini[i].state = I_SLEEP &
  --  !ismember(j,InitiatorId) &               -- ORIGINAL: responders and intruders
       ismember(j,ResponderId) &                  -- NEW : only responders
      multisetcount (l:net, true) < NetworkSize

==>
...
```

All four invariants are satisfied for the NS protocol with this modification.

In the second part of the problem, you are required to modify the model so that attackers only intercept messages that are destined to them. The modification is complicated by the fact that the adversary model is optimized. The optmimization reduces the number of states that need the be checked by having honest principals accept messages *only* from the intruder. Thus, the intruder is the only principal who directly reads messages from honest principals. It takes a little thought to see that this optmization does not preclude any attacks. The first of our modifications effectively rerverses this optimization. We modify the initiator and responder rules to accept messages from honest peers. One such modification is shown below.

```
-- initiator i reacts to nonce received (steps 6/7)
ruleset i: InitiatorId do
  choose j: net do
    rule 20 "initiator reacts to nonce received (steps 6/7)"

      ini[i].state = I_WAIT &
      net[j].dest = i -- &
    --  ismember(net[j].source,IntruderId)  --This line is not commented
                                            --out in the optimized code.
    ==>
....
```

Second, we modify the attacker code, so that it only receives messages that are destined to it.

```
-- intruder i intercepts messages
ruleset i: IntruderId do
  choose j: net do
      rule 10 "intruder intercepts"

        (!ismember (net[j].source, IntruderId))  &  -- not for intruders' messages
        ismember(net[j].dest,IntruderId)   -- This line not in the original code
      ==>
```

As in the case of the NS protocol, "initiator correctly authenticated" and "responder secrecy" fail, and the other invariants hold.

Problem 4

In this problem you had to modify the model so that the intruder can change agent names inside encrypted messages, even without knowing the decryption key. One way to achieve this is to modify the "intruder sends recorded message" as follows:

```
ruleset i: IntruderId do
    choose j: int[i].messages do
        ruleset k: AgentId do
            ruleset a: AgentId do
                rule 90 "intruder sends recorded message"
                    !ismember(k, IntruderId) &
                    multisetcount (l:net, true) < NetworkSize
                ==>
                var
                    outM: Message;
                begin
                    outM        := int[i].messages[j];
                    outM.source := i;
                    outM.dest   := k;
                    if outM.mType = M_NonceAddress then
                        outM.nonce2 := a
                    end;
                    if outM.mType = M_NonceNonceAddress then
                        outM.address := a
                    end;
                    multisetadd(outM,net);
                end;
            end;
        end;
    end;
end;
```

With the improved intruder, all invariants fail. Invariants "initiator correctly authenticated" and "responder secrecy" fail with an attack that is similar to the previously described attack on NS protocol:

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

Invariants "responder correctly authenticated" and "initiator secrecy" fail with an attack that can be described as follows:

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