EAP-IKEv2 の ProVerif による評価結果

国立研究開発法人 情報通信研究機構

1. 基本情報

- ✧ 名前

  The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method

- ✧ 機能

  AP (Access Point) peer と EAP server 間の相互認証・鍵交換プロトコル。

- ✧ 関連する標準

  RFC5106 (https://www.ietf.org/rfc/rfc5106.txt)

2. ProVerif の文法による記述

2.1. プロトコル仕様

```
free c: channel.


type host.
type key.
type pkey.
type skey.
type nonce.
type G.
type Z.


(* DH *)
const g: G.
fun exp(G, Z): G.
equation forall x: Z, y: Z; exp(exp(g, x), y) = exp(exp(g, y), x).


(* SKE *)
```

```
fun encrypt(bitstring, key): bitstring.
reduc forall x: bitstring, k: key; decrypt(encrypt(x, k), k) = x.

(* SIG *)
fun pk(skey): pkey.
fun sign(bitstring, skey): bitstring.
reduc forall m: bitstring, k: skey; getmsg(sign(m, k)) = m.
reduc forall m: bitstring, k: skey; checksign(sign(m, k), pk(k)) = m.

(* PRF *)
fun PRF(bitstring): key.

(* Consts *)

const Req: bitstring [data].
const Res: bitstring [data].
const Suc: bitstring [data].

(* table *)
table keys(host, pkey).

(* events *)
event end().
event beginPeer(host, host, key).
event endPeer(host, host, key).
event beginServer(host, key).
event endServer(host, host, key).

(* honest parties *)
free hostP, hostS: host.

(* free names for describing security *)
```

```
free secretTest, secretS, secretP: bitstring [private].

(* queries *)

query attacker(secretS).
query attacker(secretP).

query p: host, s: host, k: key;
      inj-event(endServer(p, s, k)) ==> inj-event(beginPeer(p, s, k)).



query p: host, s: host, k: key;
      inj-event(endPeer(p, s, k)) ==> inj-event(beginServer(s, k)).

(*
query attacker(secretTest).
*)

let procPeer(P: host, skP: skey) =
    in(c, (ID: host));
    (**)
    in(c, =Req);
    out(c, (Res, ID));
    in(c, (SAi1: bitstring, KEi: G, Ni: nonce));
    new Nr: nonce;
    new DHr: Z;
    let KEr = exp(g, DHr) in
    let SAr1 = SAi1 in
    out(c, (SAr1, KEr, Nr));
    in(c, ct: bitstring);
    let SK = PRF((Ni, Nr, exp(KEi, DHr))) in
    let (S: host, sAUTHi: bitstring) = decrypt(ct, SK) in
```

```
    get keys(=S, pkS) in
    if (SAi1, KEi, Ni, Nr) = checksign(sAUTHi, pkS) then
    event beginPeer(P, S, SK);
    out(c, encrypt((P, sign((SAi1, KEr, Nr, Ni), skP)), SK));
    if S = hostS then
    out(c, encrypt(secretP, SK));
    event endPeer(P, S, SK);
    event end().

let procServer(S: host, skS: skey) =
    in(c, (SAi1: bitstring));
    (**)
    out(c, Req);
    in(c, (=Res, ID: host));
    new DHi: Z;
    let KEi = exp(g, DHi) in
    new Ni: nonce;
    out(c, (SAi1, KEi, Ni));
    in(c, (=SAi1, KEr: G, Nr: nonce));
    let SK = PRF((Ni, Nr, exp(KEr, DHi))) in
    event beginServer(S, SK);
    out(c, encrypt((S, sign((SAi1, KEi, Ni, Nr), skS)), SK));
    in(c, ct: bitstring);
    let (P: host, sAUTHr: bitstring) = decrypt(ct, SK) in
    get keys(=P, pkP) in
    if (SAi1, KEr, Nr, Ni) = checksign(sAUTHr, pkP) then
    out(c, Suc);
    if P = hostP then
    out(c, encrypt(secretS, SK));
    event endServer(P, S, SK);
    event end().
```

```
let keyRegistration =
    in(c, (h: host, k: pkey));
    if h <> hostP && h <> hostS then
    insert keys(h, k).

process
         new skS: skey;
         let pkS = pk(skS) in
         insert keys(hostS, pkS);
         new skP: skey;
         let pkP = pk(skP) in
         insert keys(hostP, pkP);
         (
                  (!procServer(hostS, skS))|
                  (!procPeer(hostP, skP))|
                  (!keyRegistration)
         )
```

SPORE ライブラリのモデル化に沿って記述した。IKE のヘッダは省略した。また、オプション扱いの証明書(CERT ペイロード)及び証明書要求(CERTREQ ペイロード)を省略した。

2.2. 攻撃者モデル

上述の記述に含まれる。

2.3. セキュリティ要件

上述の記述の (* queries *) に該当する。

```
(* (1) *)
query p: host, s: host, k: key;
      inj-event(endServer(p, s, k)) ==> inj-event(beginPeer(p, s, k)).
(* (2) *)
query p: host, s: host, k: key;
      inj-event(endPeer(p, s, k)) ==> inj-event(beginServer(s, k)).
```

(1) ロール S(server)がロール P(peer)を正しく認証すること。すなわち、ロール S(server)が実行を完了したとき、同じパラメータを用いて実行を完了したロール P(peer)が存在する。
(2) ロール P(peer)がロール S(server)を正しく認証すること。すなわち、ロール P(peer)が実行を完了したとき、同じパラメータを用いて実行を完了したロール S(server)が存在する。

ただし、(2)においてロール S がロール P を相手として通信していることまでは要求しない。これを要求する形で安全性が成り立つためにはオプション扱いの{IDr}SK を送信するようにする必要がある。

3. ProVerif による評価結果

3.1. 出力

安全性はともに成り立つ（true）という結果が出力された。

3.2. 攻撃の解説

攻撃は発見されなかった。

4. 形式化

4.1. 方針

MAC と秘密鍵暗号のみを用いるため、そのまま Dolev-Yao モデル上で記述した。

4.2. 妥当性

特になし。

4.3. 検証ツールとの相性

プロトコル仕様を記述するにあたって、Diffie-Hellman 鍵交換の記述に ProVerif の等式推論の機能を使用した。攻撃者モデル、セキュリティ要件を記述するにあたっては特に制限はなかった。

4.4. 検証ツール適用時の性能

検証時間は 0.1 秒未満であった。実行環境は以下のとおり。

- ✧ Intel Core i7 L620 2.00HGz
- ✧ Windows7 上の VirtualBox 仮想マシン上の Ubuntu Linux 12.04.1 LTS
- ✧ メモリ 512MB
- ✧ ProVerif 1.86pl3

5. 備考

本文書は、**総務省「暗号・認証技術等を用いた安全な通信環境推進事業に関する実証実験の請負　成果報告書」**からの引用である。