PANA 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

  Protocol for Carrying Authentication for Network Access (PANA)

- 機能

  UDP/IP 上でのネットワークアクセス認証・鍵交換プロトコルであり、Extensible Authentication Protocol (EAP) のメッセージを運ぶ。

- 関連する標準

  RFC5191 (https://tools.ietf.org/html/rfc5191)

2. ProVerif の文法による記述

本文書では、EAP-AKA が利用されているとする。

2.1. プロトコル仕様

```
free c: channel.


type key.
type nonce.
type host.
type number.


fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].


(* Hash and MAC *)
  (* for EAP-AKA *)
fun SHA1(bitstring): bitstring.
fun FIPS_PRF_K_AUT(bitstring): key.
fun FIPS_PRF_MSK(bitstring): key.
fun PANA_PRF(bitstring): key.

fun HMAC_SHA1(key, bitstring): bitstring.
fun f1(key, nonce): bitstring.
fun f2(key, nonce): bitstring.
fun f3(key, nonce): bitstring.
fun f4(key, nonce): bitstring.
fun f5(key, nonce): bitstring.
  (* for PANA *)
fun PANA_Hash(bitstring): bitstring.

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

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

(* sequence number *)
const Zero: bitstring [data].
fun s(number): number.

const PANASuccess: bitstring [data].
const EAPSuccess: bitstring [data].

free hostA, hostP: host.

(* events *)
event beginAuthenticator(nonce, nonce, key, bitstring,
                         key, bitstring, bitstring).
event endAuthenticator(host, host, nonce, nonce, key, bitstring,
                       key, bitstring, bitstring).
```

```
event beginPeer(host, host, nonce, nonce, key, bitstring,
                key, bitstring, bitstring).
event endPeer(nonce, nonce, key, bitstring,
              key, bitstring, bitstring).

free secretAuthenticator, secretPeer: bitstring [private].

(*queries*)
query attacker(secretAuthenticator);
      attacker(secretPeer).

query p: host, a: host, cn: nonce, an: nonce, msk: key, id: bitstring,
      authkey: key, prfalg: bitstring, intalg: bitstring;
      inj-event(endAuthenticator(p, a, cn, an, msk, id, authkey,
                                 prfalg, intalg)) ==>
      inj-event(beginPeer(p,  a,  cn,  an,  msk,  id,  authkey,  prfalg,
intalg)).

query cn: nonce, an: nonce, msk: key, id: bitstring,
      authkey: key, prfalg: bitstring, intalg: bitstring;
      inj-event(endPeer(cn, an, msk, id, authkey, prfalg, intalg)) ==>
      inj-event(beginAuthenticator(cn,  an,  msk,  id,  authkey,  prfalg,
intalg)).

let procAuthenticator(a: host) =
    in(c, (PANA_PAA_ReqId: number,
           EAPReqId: number,
           PRF_Algorithm: bitstring, Integrity_Algorithm: bitstring));
    (* PANA-Client-Initiation *)
    in(c, =Zero);
    (* PANA Algorithm Negotiation *)
    let I_PAR = (PANA_PAA_ReqId, PRF_Algorithm, Integrity_Algorithm) in
    out(c, I_PAR);
    in(c, I_PAN: bitstring);
    if (PANA_PAA_ReqId, PRF_Algorithm, Integrity_Algorithm) = I_PAN
then
    (* EAP-Request/Identity *)
    new PAA_nonce: nonce;
    out(c, (s(PANA_PAA_ReqId), PAA_nonce, EAPReqId));
    in(c, (=s(PANA_PAA_ReqId), PaC_nonce: nonce));
    (* EAP-Response/Identity *)
    in(c, (PANA_PaC_ReqId: number, =EAPReqId, p: host));  (* PANA-Req
*)
    out(c, PANA_PaC_ReqId);                                (* PANA-Res
*)
    get keys(=p, =hostA, k) in
    new at_rand: nonce;
    let ck = f3(k, at_rand) in
    let ik = f4(k, at_rand) in
    let ak = f5(k, at_rand) in
    let at_autn = f1(k, at_rand) in
    let mk = SHA1((p, ik, ck)) in
    let k_aut = FIPS_PRF_K_AUT(mk) in
    let msk = FIPS_PRF_MSK(mk) in
    let at_mac = HMAC_SHA1(k_aut, (at_rand, at_autn)) in
    (* EAP-Request/AKA-Challenge *)
    out(c, (s(s(PANA_PAA_ReqId)), s(EAPReqId), (at_rand, at_autn,
at_mac)));
    in(c, =s(s(PANA_PAA_ReqId)));
    (* EAP-Response/AKA-Challenge *)
    in(c, (=s(PANA_PaC_ReqId), =s(EAPReqId),
           (at_res: bitstring, at_mac2: bitstring)));
    out(c, s(PANA_PaC_ReqId));
    if at_mac2 = HMAC_SHA1(k_aut, at_res) &&
       at_res = f2(k, at_rand) then
    (* EAP-Success *)
    new KeyId: bitstring;
    let m = (s(s(s(PANA_PAA_ReqId))), PANASuccess, EAPSuccess, KeyId)
in
    let msk = FIPS_PRF_MSK(mk) in
    let PANA_AUTH_KEY
        = PANA_PRF((msk, I_PAR, I_PAN, PaC_nonce, PAA_nonce, KeyId)) in
    let AUTH = PANA_Hash((PANA_AUTH_KEY, m)) in
    event beginAuthenticator(PaC_nonce, PAA_nonce, msk, KeyId,
                              PANA_AUTH_KEY,            PRF_Algorithm,
Integrity_Algorithm);
    out(c, (s(s(s(PANA_PAA_ReqId))), PANASuccess, EAPSuccess, KeyId,
AUTH));
    in(c, (=s(s(s(PANA_PAA_ReqId))), AUTH2: bitstring));
    if AUTH2 = PANA_Hash((PANA_AUTH_KEY, s(s(s(PANA_PAA_ReqId))))) then
    (* for security *)
    if p = hostP then
    out(c, encrypt(secretAuthenticator, msk));
    event endAuthenticator(p, a, PaC_nonce, PAA_nonce, msk, KeyId,
                            PANA_AUTH_KEY,              PRF_Algorithm,
Integrity_Algorithm).

let procPeer(p: host) =
    in(c, (a: host, PANA_PaC_ReqId: number,
           PRF_Algorithm: bitstring, Integrity_Algorithm: bitstring));
    (* PANA-Client-Initiation *)
    out(c, Zero);
    get keys(=p, =a, k) in
    (* PANA Algorithm Negotiation *)
    in(c, I_PAR: bitstring);
    let (PANA_PAA_ReqId: number, =PRF_Algorithm, =Integrity_Algorithm)
```

```
        = I_PAR in
    let I_PAN = (PANA_PAA_ReqId, PRF_Algorithm, Integrity_Algorithm) in
    out(c, I_PAN);
    (* EAP-Request/Identity *)
    in(c, (=s(PANA_PAA_ReqId), PAA_nonce: nonce, EAPReqId: number));
    new PaC_nonce: nonce;
    out(c, (s(PANA_PAA_ReqId), PaC_nonce));
    (* EAP-Response/Identity *)
    out(c, (PANA_PaC_ReqId, EAPReqId, p));                        (*
PANA-Req *)
    in(c, (=PANA_PaC_ReqId));                                     (*
PANA-Res *)
    (* EAP-Request/AKA-Challenge *)
    in(c, (=s(s(PANA_PAA_ReqId)), =s(EAPReqId),
           (at_rand: nonce, at_autn: bitstring, at_mac: bitstring)));
    out(c, s(s(PANA_PAA_ReqId)));
    let ck = f3(k, at_rand) in
    let ik = f4(k, at_rand) in
    let ak = f5(k, at_rand) in
    let mk = SHA1((p, ik, ck)) in
    let k_aut = FIPS_PRF_K_AUT(mk) in
    if at_mac = HMAC_SHA1(k_aut, (at_rand, at_autn)) &&
       at_autn = f1(k, at_rand) then
    let at_res = f2(k, at_rand) in
    let at_mac2 = HMAC_SHA1(k_aut, at_res) in
    let msk = FIPS_PRF_MSK(mk) in
    (* EAP-Response/AKA-Challenge *)
    out(c, (s(PANA_PaC_ReqId), s(EAPReqId),
            (at_res, at_mac2)));
    in(c, =s(PANA_PaC_ReqId));
    (* EAP-Success *)
    in(c, (=s(s(s(PANA_PAA_ReqId))), =PANASuccess, =EAPSuccess,
          KeyId: bitstring, AUTH: bitstring));
    let m = (s(s(s(PANA_PAA_ReqId))), PANASuccess, EAPSuccess, KeyId)
in
    let PANA_AUTH_KEY
        = PANA_PRF((msk, I_PAR, I_PAN, PaC_nonce, PAA_nonce, KeyId)) in
    if AUTH = PANA_Hash((PANA_AUTH_KEY, m)) then
    let AUTH2 = PANA_Hash((PANA_AUTH_KEY, s(s(s(PANA_PAA_ReqId))))) in
    event beginPeer(p, a, PaC_nonce, PAA_nonce, msk, KeyId,
                        PANA_AUTH_KEY,              PRF_Algorithm,
Integrity_Algorithm);
    out(c, (s(s(s(PANA_PAA_ReqId))), AUTH2));
    (* for security *)
    if a = hostA then
    out(c, encrypt(secretPeer, msk));
    event endPeer(PaC_nonce, PAA_nonce, msk, KeyId,
                  PANA_AUTH_KEY, PRF_Algorithm, Integrity_Algorithm).

let keyRegistration =
    in(c, (h1: host, h2: host, k: key));
    if (h1, h2) <> (hostP, hostA) then
    insert keys(h1, h2, k).

process
         new Kpa: key;
         insert keys(hostP, hostA, Kpa);
         ((!procPeer(hostP))|
          (!procAuthenticator(hostA))|
          (!keyRegistration))
```

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 attacker(secretAuthenticator);
      attacker(secretPeer).

(* (2) *)
query p: host, a: host, cn: nonce, an: nonce, msk: key, id: bitstring,
      authkey: key, prfalg: bitstring, intalg: bitstring;
      inj-event(endAuthenticator(p, a, cn, an, msk, id, authkey,
                           prfalg, intalg)) ==>
      inj-event(beginPeer(p, a, cn, an, msk, id, authkey, prfalg,
intalg)).

(* (3) *)
query cn: nonce, an: nonce, msk: key, id: bitstring,
      authkey: key, prfalg: bitstring, intalg: bitstring;
      inj-event(endPeer(cn, an, msk, id, authkey, prfalg, intalg)) ==>
      inj-event(beginAuthenticator(cn, an, msk, id, authkey, prfalg,
intalg)).
```

(1) EAP-AKA の結果生成されたマスターセッション鍵(MSK)の秘匿性。
(2) サーバによるクライアントの認証及び PANA のセキュリティアソシエーションの一致。
(3) クライアントのセキュリティアソシエーションと一致するセキュリティアソシエーションをもつ参加者の存在。

なお、EAP-AKA ではクライアントによるサーバの認証はできないことが明らかなため、その評価はしなかった。

3. ProVerif による評価結果

3.1. 出力

いずれも安全であること (true) が確認できた。

```
RESULT inj-event(endPeer(cn,an,msk_33,id,authkey,prfalg,intalg)) ==>
inj-event(beginAuthenticator(cn,an,msk_33,id,authkey,prfalg,intalg))
 is true.
```

```
RESULT inj-event(endAuthenticator(p_7810,a_7811,cn_7812,an_7813,
msk_7814,id_7815,authkey_7816,prfalg_7817,intalg_7818)) ==>
inj-event(beginPeer(p_7810,a_7811,cn_7812,an_7813,
msk_7814,id_7815,authkey_7816,prfalg_7817,intalg_7818)) is true.
RESULT not attacker(secretAuthenticator[]) is true.
RESULT not attacker(secretPeer[]) is true.
```

3.2. 攻撃の解説

前述のとおり、攻撃は発見されなかった。

4. 形式化

4.1. 方針

特になし。

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、攻撃者モデル、セキュリティ要件を ProVerif で記述するにあたって、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 0.5 秒であった。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。