Unbinding AS-REP from AS-REQ in PKINIT

I. Cervesato[1], A. D. Jaggard[1], A. Scedrov[2], J.-K. Tsay[2], and C. Walstad[2]

IETF-63

Kerberos WG

1 August 2005

[1]Tulane University and [2]University of Pennsylvania

Partially supported by ONR and NSF

Our Work

- Part of ongoing formal analysis of Kerberos 5 suite
  - Previously studied basic and cross-realm authentication
- Attack on pk-init-26 found when KDC uses public-key mode
  - Breaks binding between AS-REQ and AS-REP
  - Prevents full authentication in pk-init-26
- Formal verification of fixes preventing attack

Attack and Fixes (Overview)

- Authentication attack
  - KDC believes he is talking to the attacker
  - Client believes she is talking to the KDC
  - Attacker knows the key shared by the client and KDC
- Possible because the KDC does not sign data identifying the client
  - Attacker constructs AS-REQ based on client's AS-REQ
  - KDC signs data from client, sends in AS-REP to attacker
  - Attacker forwards this to client after learning keys
  - Ran Canetti , consulted on details of spec., independently hypothesized the possibility of an "identity misbinding" attack
- pk-init-27 is intended to defend against this attack
  - KDC signs data derived from client's identity

Consequences of the Attack

- **The attacker knows the keys C uses;** *she may:*
  - *Impersonate servers (in later rounds) to the client C*
  - *Monitor C's communications with the end server*
- *Other notes*
  - *Attacker must be a legal user*
  - *C is authenticated to end server as attacker (not as C)*
  - *DH mode appears to avoid this attack*
    - *Still need to prove formally security for DH*

Formalizing the AS-REQ

- Our formalization of pa-data includes
  - t_C = cusec/ctime (in pkAuthenticator)
  - n_2 = nonce (in pkAuthenticator)
  - $[t_C, n_2]_{skC}$ = signature (in signerInfos) over t_C, n_2 using C's secret key skC
- Our formalization of req-body includes
  - C = cname
  - $\mathcal{T}$ = sname
  - n_1 = nonce

$$t_C, n_2, [t_C, n_2]_{skC}, C, \mathcal{T}, n_1$$

Formalizing the AS-REP

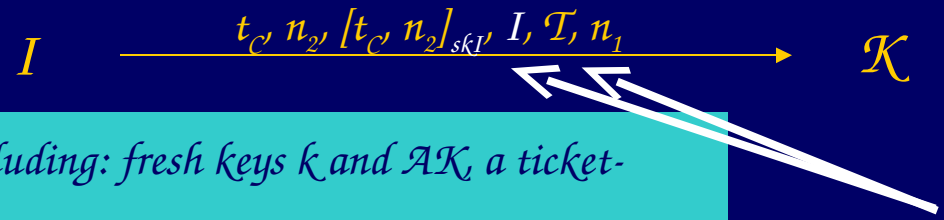
- Our formalization of pa-data includes
  - k = replyKey (in ReplyKeyPack)
  - n_2 = nonce (in ReplyKeyPack), from AS-REQ
  - $[k, n_2]_{skK}$ = signature with K's secret key skK
  - $\{\ldots\}_{pkC}$ is encryption with C's public key pkC
- C = cname in AS-REP
- X = ticket in AS-REP
- Our formalization of enc-part includes
  - AK = key
  - n_1 = nonce
  - t_K = authtime
  - sname = T
  - $\{\ldots\}_k$ is encryption with the reply key k

$$\{k, n_2, [k, n_2]_{skK}\}_{pkC}, C, X, \{AK, n_1, t_K, T\}_k$$

The Attack

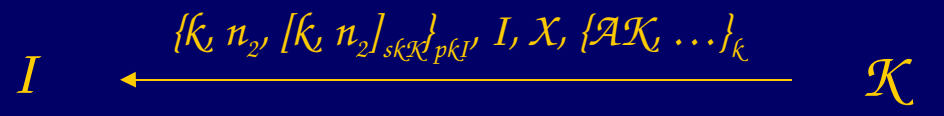
At time t_C, client C requests a ticket for ticket server T (using nonces n_1 and n_2):

The attacker I intercepts this, puts her name/signature in place of C's:



Kerberos server K replies with credentials for I, including: fresh keys k and AK, a ticket-granting ticket X, and K's signature over k,n_2:

(Ignore most of enc-part)



I decrypts, re-encrypts with C's public key, and replaces her name with C's:

- **I knows fresh keys k and AK**
- C receives K's signature over k,n_2 and assumes k, AK, etc., were generated for C (not I)

- Principal P has secret key skP, public key pkP
- $\{msg\}_{key}$ is encryption of msg with key
- $[msg]_{key}$ is signature over msg with key

Consequences of the Attack

- **The attacker knows the keys C uses;** *she may:*
  - *Impersonate servers (in later rounds) to the client C*
  - *Monitor C's communications with the end server*
- *Other notes*
  - *Attacker must be a legal user*
  - *C is authenticated to end server as attacker (not as C)*
  - *DH mode appears to avoid this attack*
    - *Still need to formally prove security for DH*

Desired Authentication Property

If a client C processes a message containing KDC-generated public-key credentials, then some KAS K produced a set of such credentials **for C**.

- *The attack shows this property does not hold in pk-init-26*
- *We believe this property holds if:*
  - *The KAS signs F(C), k, n_2*
  - *The AS-REP is as in pk-init-27*

Preventing the Attack in General

- Sign data identifying client
  - The KDC signs $F(C)$, k, n_2
  - Assume $F(C) = F(C')$ implies $C = C'$
  - AS-REQ message now formalized as

$$\{k, n_2, [F(C), k, n_2]_{sk(K)}\}_{pkC}, C, X, \{AK, n_1, t_K, T\}_k$$

- We have a draft formal proof that this guarantees authentication
  - Does cname/crealm uniquely identify client?
  - Added secrecy properties if $F(C)$ identifies pkC?

pk-init-27 and the Attack

- *In the change implemented in pk-init-27:*
  - *The KDC signs k, cksum (i.e., cksum in place of n_2)*
    - *k is replyKey*
    - *cksum is checksum over AS-REQ*
  - *AS-REP now formalized as*

$$\{k, cksum, [k, cksum]_{skK}\}_{pkC}, C, X, \{AK, n_1, t_K, T\}_k$$

- *We have a formal proof that this guarantees authentication*
  - *Assume checksum is collision-free*
  - *Assume KDC's signature keys are secret*
  - *Plan to carry out a more detailed, cryptographic proof in the future*

ReplyKeyPack in pk-init-26

```
ReplyKeyPack ::= SEQUENCE {
    replyKey  [0] EncryptionKey,
      -- Contains the session key used to encrypt the
      -- enc-part field in the AS-REP.
    nonce                 [1] INTEGER (0..4294967295),
      -- Contains the nonce in the PKAuthenticator of the
      -- request. ... }
```

ReplyKeyPack in pk-init-27

```
ReplyKeyPack ::= SEQUENCE {
    replyKey     [0] EncryptionKey,
       -- Contains the session key used to encrypt the
       -- enc-part field in the AS-REP.
    asChecksum[1] Checksum,
       -- Contains the checksum of the AS-REQ
       -- corresponding to the containing AS-REP.
       -- The checksum is performed over the type AS-REQ.
       -- The protocol key [RFC3961] of the checksum is the
       -- replyKey and the key usage number is 6.
       -- If the replyKey's enctype is "newer" [RFC4120]
       -- [RFC4121], the checksum is the required
       -- checksum operation [RFC3961] for that enctype.
       -- The client MUST verify this checksum upon receipt
       -- of the AS-REP. ... }
```

Future Work

- We will have a technical draft on this in several weeks
  - We will post for comments from WG
- Later: A full analysis and verification of PKINIT
  - Cryptographic proofs
  - We will look at DH mode
- Other parts of Kerberos suite
  - Which protocol components might benefit most from formal analysis?
- We will report results of continuing work to WG
- Thanks to Ran Canetti, Sam Hartman, and Jeffrey Hutzelman for interesting and fruitful discussions

Bonus Slides

After the AS-REQ/-REP

- Both the attacker I and client C know the keys k and AK
  - C believes the KDC produced k and AK for C
- Attacker may monitor communications
  - Attacker must put her name into the TGS-REQ and AP-REQ messages to match the tickets
  - Attacker learns keys in TGS-REP and AP-REP
- Attacker may impersonate servers
  - Instead of forwarding modified –REQ messages, attacker may simply forge –REP messages herself

Proof Sketch for General Defense

- Assume
  - Client receives AS-REP with $[F(C), k, n_2]_{sk_K}$
  - KAS's signature key is secret
  - Signatures are unforgeable
  - $F(C) = F(C')$ implies $C = C'$
- Proof sketch
  - Signature in AS-REP must come from the KAS K
  - K would only produce this signature in response to an AS-REQ containing C' such that $F(C') = F(C)$
  - Collision-freeness of F implies that K created the AS-REP for C

Proof Sketch for pk-init-27

- Assume
  - Client receives AS-REP as in pk-init-27
  - KAS's signature key is secret
  - Signatures are unforgeable
  - Checksums are collision-free
- Proof sketch
  - Signature in AS-REP must come from the KAS K
  - K would only produce this signature in response to an AS-REQ whose checksum is the signed value
  - Collision-freeness of checksums implies that the AS-REQ was as claimed (including C's name)

The Attack (with Certificates)

At time t_C, client C requests a ticket for ticket server T (using nonces n_1 and n_2):

$C \xrightarrow{t_C,\ n_2,\ Cert_C,\ [t_C,\ n_2]_{skC},\ Trust_C,\ C,\ T,\ n_1} I$

The attacker I intercepts this, puts her name/signature in place of C's:

$I \xrightarrow{t_C,\ n_2,\ Cert_I,\ [t_C,\ n_2]_{skI},\ Trust_C,\ I,\ T,\ n_1} K$

Kerberos server K replies with credentials for I, including: fresh keys k and AK, a ticket-granting ticket X, and K's signature over k,n_2:

(Ignore most of enc-part)

$I \xleftarrow{\{k,\ n_2,\ Cert_K,\ [k,\ n_2]_{skK}\}_{pkI},\ I,\ X,\ \{AK,\ \ldots\}_k} K$

I decrypts, re-encrypts with C's public key, and replaces her name with C's:

$C \xleftarrow{\{k,\ n_2,\ Cert_K,\ [k,\ n_2]_{skK}\}_{pkC},\ C,\ X,\ \{AK,\ \ldots\}_k} I$

- **I knows fresh keys k and AK**
- C receives K's signature over k,n_2 and assumes k, AK, etc., were generated for C (not I)

- Principal P has secret key skP, public key pkP
- $\{msg\}_{key}$ is encryption of msg with key
- $[msg]_{key}$ is signature over msg with key

What If skC is Lost?

- Assume client C loses her decryption key
  - C generates new skC/pkC pair
  - KDC K has not yet learned of this update
  - C uses a different key for signatures than for decryption
- Even after fixes described to prevent attack:
  - Attacker may intercept AS-REP (knowing skClost)
  - Attacker re-encrypts using C's new public key pkCnew
  - C is unable to detect tampering
  - Authentication holds, secrecy does not
- Possible fix: sign pkC (or fingerprint?)
  - Loss of pkC is separate problem (and maybe not of concern), but might be addressable when fixing the binding problem

A Secrecy Question

At time t_C, client C requests a ticket for ticket server T (using nonces n_1 and n_2), signing this with her signature key:

Kerberos server K replies with credentials for C, including: fresh keys k and AK, a ticket-granting ticket X, and K's signature over k,cksum (per pk-init-27). The encryption is with C's compromised public key:

I decrypts, re-encrypts with C's new public key, forwards the result to C:

- **I knows fresh keys k and AK**
- C knows that K generated k and AK for C, but does not know that I also knows these

- C uses skCsign for signatures
- $\{msg\}_{key}$ is encryption of msg with key
- $[msg]_{key}$ is signature over msg with key