Kerberos Network Authentication Service (V5)

basic (core)

Protocol Purpose

Authentication, Authorisation, Key Exchange

Kerberos is a distributed authentication service that allows a process (a client) running on behalf of a principal (a user) to prove its identity to a verifier (an application server, or just server) without sending data across the network that might allow an attacker or the verifier to subsequently impersonate the principal. Kerberos optionally provides integrity and confidentiality for data sent between the client and server.

Definition Reference

- http://www.ietf.org/internet-drafts/draft-ietf-krb-wg-kerberos-clarifications-07.txt

Model Authors

- Haykal Tej, Siemens CT IC 3, 2003
- Sebastian Mödersheim, Computer Security Group, ETH Zürich, January 2004
- AVISPA team (since then)

Alice&Bob style

```
C: Client
A: Authentication Server
G: Ticket Granting Server
S: Server (that the client wants to talk to)

K_AB: key shared or intended to be shared between A and B
      Initially shared: K_CA, K_AG, K_GS
      Established during protocol: K_CG, K_CS

All things marked * are timestamp-related and will be simply replaced
```

```
with fresh text.

Macros:
Ticket_1 := { C,G, K_CG, Tstart*, Texpire* }K_AG
Ticket_2 := { C,S, K_CS, Tstart2*, Texpire2* }K_GS

1. C -> A : C,G,Lifetime_1*,N_1
2. A -> C : C, Ticket_1, { G, K_CG, Tstart*, Texpire*, N_1 }K_CA

3. C -> G : S,Lifetime_2*,N_2,Ticket_1, { C,T* }K_CG
4. G -> C : C, Ticket_2, { S, K_CS, Tstart2*, Texpire2*, N_2 }K_CG

5. C -> S : Ticket_2, { C, T2* }K_CS
6. S -> C : { T2* }K_CS
```

Model Limitations

Ticket Caching is not performed, so only weak authentication is provided. It is rumoured that implementations do not perform ticket caching.

Problems considered: 8

Attacks Found

None

Further Notes

Agents involved: Client, Authentication Server (AS), Ticket Granting server (TGS), Server where the client needs to authenticate (Server)

HLPSL Specification

```
% Authentication Server
```

```
role kerberos_A (A, C, G : agent,
                 Snd, Rcv   : channel (dy),
                 K_CA, K_AG : symmetric_key)
played_by A
def=

  local St               : nat,
        K_CG             : symmetric_key,
        N1, Lifetime_1   : text,
        Tstart, Texpire : text

  const k_cg : protocol_id,
        sec_a_K_CG : protocol_id

  init  St := 0

  transition

   1. St = 0  /\ Rcv(C.G.Lifetime_1'.N1') =|>
      St':= 1 /\ Tstart' := new()
              /\ Texpire' := new()
              /\ K_CG' := new()
              /\ Snd(C.{C.G.K_CG'.Tstart'.Texpire'}_K_AG.
                       {G.K_CG'.Tstart'.Texpire'.N1'}_K_CA)
              /\ witness(A,C,k_cg,K_CG')
              /\ witness(A,G,k_cg,K_CG')
              /\ secret(K_CG',sec_a_K_CG,{A,C,G})

end role
```

```
% Ticket Granting Server
role kerberos_G (G, A, S, C  : agent,
                 Snd, Rcv    : channel (dy),
                 K_AG, K_GS  : symmetric_key)
played_by G
def=

  local St                                : nat,
        K_CG                              : symmetric_key,
      K_CS                                : symmetric_key,
      Lifetime_2, Tstart, Texpire, T, N2 : text,
      Tstart2, Texpire2                  : text

  const t1,k_cs : protocol_id,
      sec_g_K_CG, sec_g_K_CS : protocol_id

  init  St := 0

  transition

   1. St = 0  /\
       Rcv(S.Lifetime_2'.N2'.{C.G.K_CG'.Tstart'.Texpire'}_K_AG.{C.T'}_K_CG') =|>
      St':= 1 /\ K_CS' := new()
              /\ Tstart2' := new()
              /\ Texpire2' := new()
              /\ Snd(C.
                   {C.S.K_CS'.Tstart2'.Texpire2'}_K_GS.
                     {S.K_CS'.Tstart2'.Texpire2'.N2'}_K_CG')
              /\ wrequest(G,C,t1,T')
              /\ wrequest(G,A,k_cg,K_CG')
              /\ witness(G,S,k_cs,K_CS')
              /\ witness(G,C,k_cs,K_CS')
              /\ secret(K_CG',sec_g_K_CG,{A,C,G})
              /\ secret(K_CS',sec_g_K_CS,{G,C,S})

end role
```

```
% Server
role kerberos_S (S, G, C  : agent,
                 Snd, Rcv : channel (dy),
                 K_GS     : symmetric_key)
played_by S
def=

  local St                   : nat,
      Tstart2, Texpire2, T2 : text,
      K_CS                  : symmetric_key
  const t2a, t2b : protocol_id,
        sec_s_K_CS : protocol_id

  init  St := 0

  transition

   1. St = 0  /\ Rcv({C.S.K_CS'.Tstart2'.Texpire2'}_K_GS.{C.T2'}_K_CS') =|>
      St':= 1 /\  Snd({T2'}_K_CS')
              /\ witness(S,C,t2a,T2')
              /\ wrequest(S,G,k_cs,K_CS')
              /\ wrequest(S,C,t2b,T2')
              /\ secret(K_CS',sec_s_K_CS,{G,C,S})

end role
```

```
% Client
role kerberos_C (C, A, G, S : agent,
                 Snd, Rcv   : channel (dy),
                 K_CA       : symmetric_key)
played_by C
def=

  local St                                : nat,
      K_CG, K_CS                          : symmetric_key,
      T, T2 : text,
      Tstart, Texpire, Tstart2, Texpire2    : text,
      Ticket_1, Ticket_2 : {agent.agent.symmetric_key.text.text}_symmetric_key,
      N1, N2  : text

  const t1, k_cg, k_cs, t2a, t2b : protocol_id,
        sec_c_K_CG, sec_c_K_CS : protocol_id,
        cLifetime_1, cLifetime_2: text

  init  St := 0

  transition

   1. St = 0  /\ Rcv(start) =|>
      St':= 1 /\ N1' := new()
```

```
             /\ Snd(C.G.cLifetime_1.N1')

   2. St = 1  /\ Rcv(C.Ticket_1'.{G.K_CG'.Tstart'.Texpire'.N1}_K_CA) =|>
      St':= 2 /\ N2' := new()
              /\ T' := new()
              /\ Snd(S.cLifetime_2.N2'.Ticket_1'.{C.T'}_K_CG')
              /\ witness(C,G,t1,T')
              /\ wrequest(C,A,k_cg,K_CG')
              /\ secret(K_CG',sec_c_K_CG,{A,C,G})

   3. St = 2  /\ Rcv(C.Ticket_2'.{S.K_CS'.Tstart2'.Texpire2'.N2}_K_CG)  =|>
      St':= 3 /\ T2' := new()
              /\ Snd(Ticket_2'.{C.T2'}_K_CS')
              /\ witness(C,S,t2b,T2')
              /\ wrequest(C,G,k_cs,K_CS')
              /\ secret(K_CS',sec_c_K_CS,{G,C,S})

   4. St = 3  /\ Rcv({T2}_K_CS) =|>
      St':= 4 /\ wrequest(C,S,t2a,T2)

end role
```

```
role session( C, A, G, S                             : agent,
              K_CA, K_AG, K_GS                       : symmetric_key)
def=

   local S_C, R_C, S_A, R_A, S_G, R_G, S_S, R_S : channel (dy)

   composition

        kerberos_C(C,A,G,S,S_C,R_C,K_CA)
     /\ kerberos_A(A,C,G,S_A,R_A,K_CA,K_AG)
     /\ kerberos_G(G,A,S,C,S_G,R_G,K_AG,K_GS)
     /\ kerberos_S(S,G,C,S_S,R_S,K_GS)

end role
```

```
role environment() def=

  const  c, a, g, s, i           : agent,
         kca, kag, kgs, kia      : symmetric_key

  intruder_knowledge = {c,a,g,s,kia
                       }

  composition
        session(c,a,g,s,kca,kag,kgs)
 /\     session(i,a,g,s,kia,kag,kgs)

end role
```

```
goal

  %secrecy_of K_CG, K_CS
  secrecy_of sec_a_K_CG,
             sec_g_K_CG, sec_g_K_CS,
             sec_s_K_CS,
             sec_c_K_CG, sec_c_K_CS

  %Kerberos_C weakly authenticates Kerberos_A on k_cg
  weak_authentication_on k_cg
  %Kerberos_G weakly authenticates Kerberos_A on k_cg
  weak_authentication_on k_cg

  %Kerberos_C weakly authenticates Kerberos_G on k_cs
  weak_authentication_on k_cs
  %Kerberos_S weakly authenticates Kerberos_G on k_cs
  weak_authentication_on k_cs

  %Kerberos_C weakly authenticates Kerberos_S on t2a
  weak_authentication_on t2a
  %Kerberos_S weakly authenticates Kerberos_C on t2b
  weak_authentication_on t2a

  %Kerberos_G weakly authenticates Kerberos_C on t1
  weak_authentication_on t1
```

```
end goal
```

```
environment()
```

References