Network Security (NetSec)

IN2101 – WS 16/17

Prof. Dr.-Ing. Georg Carle

Dr. Heiko Niedermayer
Cornelius Diekmann

Version: December 12, 2016

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Chapter 11: Kerberos and other Frameworks for Client Authentication

Introduction

Learning Goals

Frameworks for Client Authentication

Kerberos

Conclusions – What have we learned

- This course is based to some extend on slides provided by Günter Schäfer, author of the book "Netzsicherheit - Algorithmische Grundlagen und Protokolle", available in German from dpunkt Verlag.
- The English version of the book is entitled “Security in Fixed and Wireless Networks: An Introduction to Securing Data Communications” and is published by Wiley is also available. We gratefully acknowledge his support.
- The slide set has been reworked by Heiko Niedermayer, Ali Fessi, Ralph Holz, Cornelius Diekmann, and Georg Carle.

Chapter 11: Kerberos and other Frameworks for Client Authentication

- Slides called "- Explanation" and usually marked with 🐴 are not for the lecture, but they contain further explanations for your learning at home.
- Parts called "Exercise" are voluntary exercises for discussion in lecture as well as for your reworking of the slides and learning at home.

Agenda

Introduction

Learning Goals

Frameworks for Client Authentication

Kerberos

Conclusions – What have we learned

Chapter 11: Kerberos and other Frameworks for Client Authentication

Introduction

Learning Goals

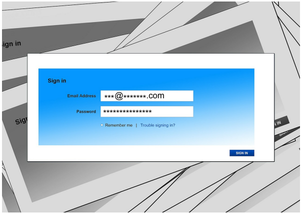
Frameworks for Client Authentication

Kerberos

Conclusions – What have we learned

Introduction

- User Authentication



Sign in

Email Address ***@*******.com

Password ***************

○ Remember me | Trouble signing in?

SIGN IN

Introduction

- User Authentication
  - Username
  - Credential
- Credential usually:
  - Password / Shared Key
  - Signature (less frequent)
- Multiple Factors
  - Know password
  - Possess card
  - ....
  - Benefit of multiple factors is to have multiple lines of defense. Makes attacks more complicated, not impossible.

Login to Services on the Network

- The service needs to know user and a credential.
- In this chapter, we usually assume the use of symmetric cryptography.
- Keys derived from password or other factor.

Chapter 11: Kerberos and other Frameworks for Client Authentication

Introduction

Learning Goals

Frameworks for Client Authentication

Kerberos

Conclusions – What have we learned

Learning Goals

- Basic understanding of the Kerberos protocol
  - Remember and explain
  - Apply what you learned in crypto prots on it and modified protocols
- Ticket concept
  - Remember and explain
  - Apply in modified setting
- Inclusion of Authentication Backends
  - Know of concept, know names of some backends
  - Where does user authentication come from?

Introduction

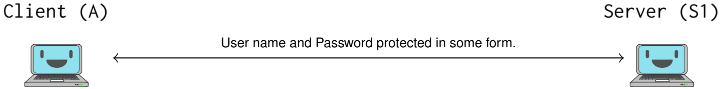
Learning Goals

Frameworks for Client Authentication

Kerberos

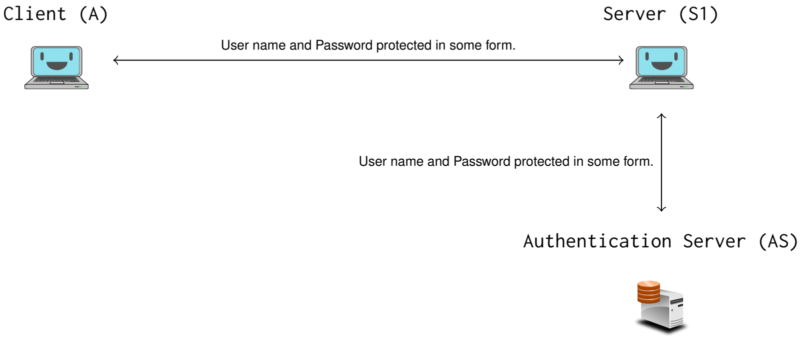
Conclusions – What have we learned

Authenticating to a Server



- A client has accounts at each of the servers it uses.
- Each server has a user database and operates as authentication server.
- Disadvantages: hard to manage, number of passwords, ...

Authenticating to a Server with Authentication Server



Client (A)

Server (S1)

User name and Password protected in some form.

User name and Password protected in some form.

Authentication Server (AS)

- An Authentication Server (AS) manages the accounts.
- Authentication has to happen via the AS.

Authentication with external AS

- Option: S1 uses the credentials provided by A to run a cryptographic protocol with AS.
    - Common practice for local authentication within one infrastructure (Kerberos might be used).
- Alternative (better): End-to-end Authentication between A and AS. S1 relays messages and AS informs S1 about outcome (ACCEPT / DENY).
    - Found in more public infrastructures.
    - Link Layer Access: Extensible Authentication Protocol (EAP), ...
- Alternative (better): A runs protocol with AS, interaction results in information A and S1 can use to mutually authenticate.
    - Can happen before A and S1 interact or in-between.
    - OpenID, OAuth, ...
    - Kerberos (original idea)

Authentication with external AS

- Local infrastructures may want to use different authentication services
  - Make authentication services accessible via a generic API, e.g. PAM, GSSAPI, ...
  - Many authentication services themselves operate primarily as generic transport protocols for cryptographic protocol messages as well as authentication and authorization results (e.g. EAP, Radius, ...).

Chapter 11: Kerberos and other Frameworks for Client Authentication

Introduction

Learning Goals

Frameworks for Client Authentication

Kerberos

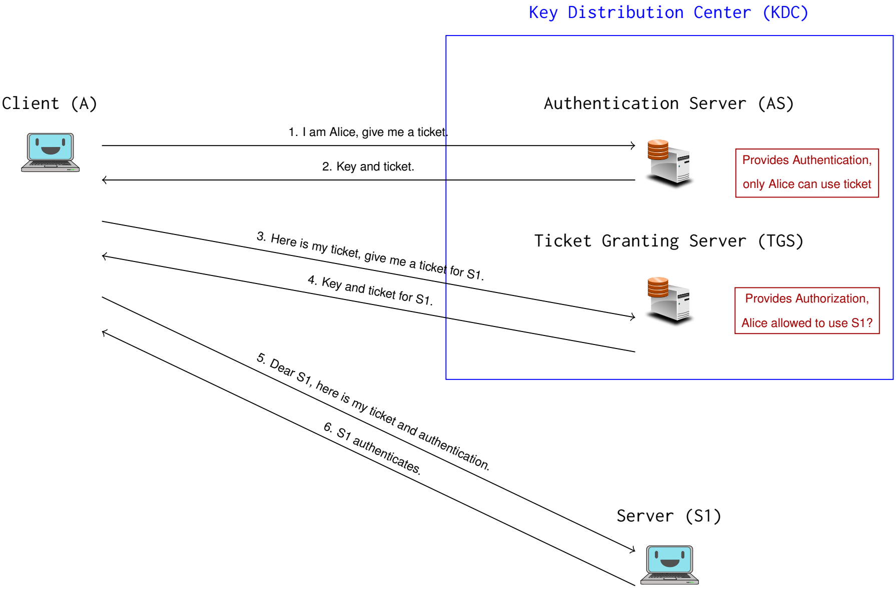
Conclusions – What have we learned

Kerberos

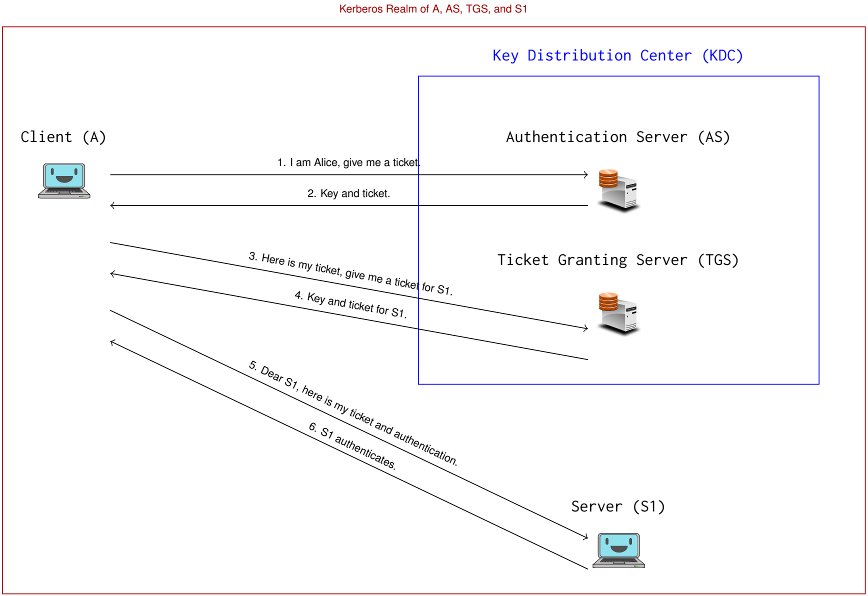
Kerberos is an authentication and access control service for work-station clusters that was designed at the MIT during the late 1980s

- Design goals:
  - Security: eavesdroppers or active attackers should not be able to obtain the necessary information to impersonate a user when accessing a service
  - Reliability: as every use of a service requires prior authentication, Kerberos should be highly reliable and available
  - Transparency: the authentication process should be transparent to the user beyond the requirement to enter a password
  - Scalability: the system should be able to support a large number of clients and servers

Kerberos – Concept


Key Distribution Center (KDC)
Client (A)
Authentication Server (AS)
1. I am Alice, give me a ticket.
2. Key and ticket.
Provides Authentication,
only Alice can use ticket
Ticket Granting Server (TGS)
3. Here is my ticket, give me a ticket for S1.
4. Key and ticket for S1.
Provides Authorization,
Alice allowed to use S1?
5. Dear S1, here is my ticket and authentication.
6. S1 authenticates.
Server (S1)

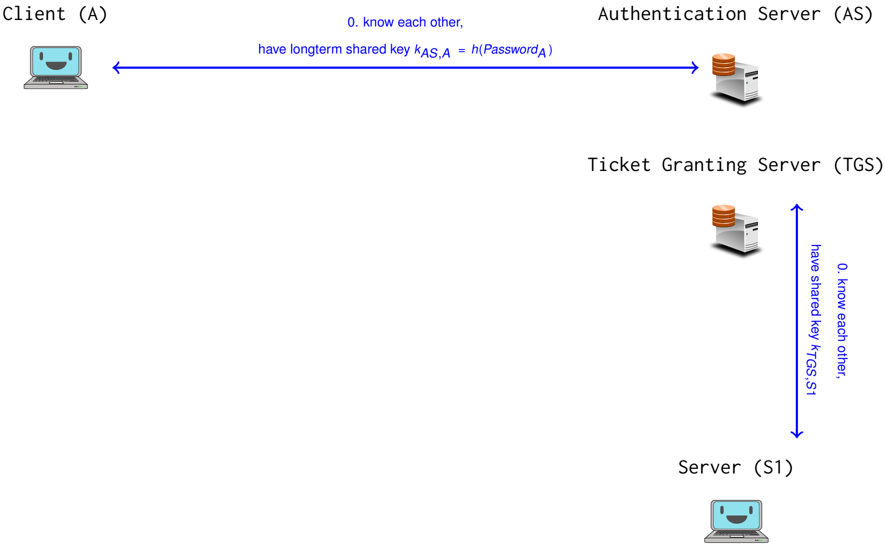
Kerberos – Concept


Kerberos Realm of A, AS, TGS, and S1
Key Distribution Center (KDC)
Client (A)
Authentication Server (AS)
1. I am Alice, give me a ticket.
2. Key and ticket.
Ticket Granting Server (TGS)
3. Here is my ticket, give me a ticket for S1.
4. Key and ticket for S1.
5. Dear S1, here is my ticket and authentication.
6. S1 authenticates.
Server (S1)

Kerberos – Needham-Schroeder

- Conceptually, Kerberos is based on the Needham-Schroeder Symmetric Key Protocol
    - uses timestamps instead of nonces (random numbers)
- Key Distribution Center
    - provides authentication and authorization
    - generates and provides the keys for the next steps
- Ticket
    - like in Needham-Schroeder binds key and identity of client
    - binds ticket to IP address of client
- Realm
    - Kerberos operates in organizational realms
    - Operation is limited to realm
    - Multi-realm possible if realms cooperate
- Password
    - Shared key with AS derived from user password, traditionally $k_{A,AS} = md5(Password_A)$

Kerberos – Concept



Client (A)

Authentication Server (AS)

0. know each other,

have longterm shared key $k_{AS,A} = h(Password_A)$

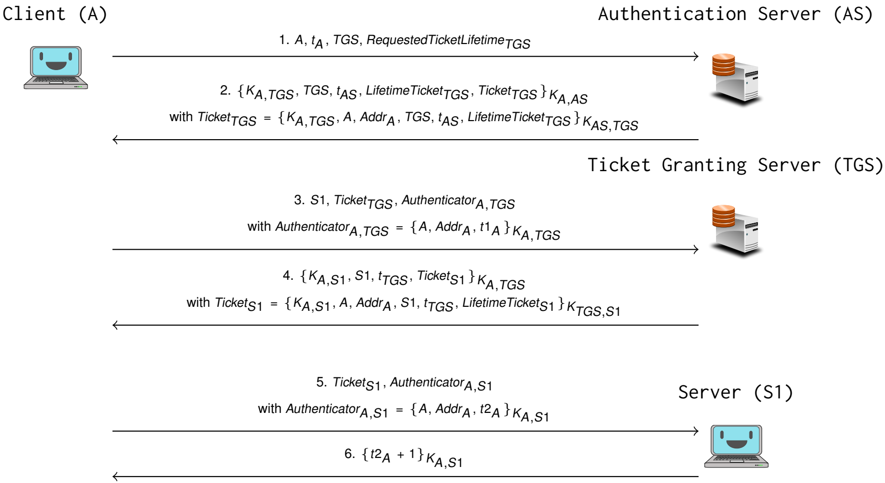
Ticket Granting Server (TGS)

0. know each other,

have shared key $k_{TGS,S1}$

Server (S1)

Kerberos - Protocol



Client (A)

Authentication Server (AS)

1. $A,\ t_A,\ TGS,\ RequestedTicketLifetime_{TGS}$

2. $\{K_{A,TGS},\ TGS,\ t_{AS},\ LifetimeTicket_{TGS},\ Ticket_{TGS}\}K_{A,AS}$

with $Ticket_{TGS} = \{K_{A,TGS},\ A,\ Addr_A,\ TGS,\ t_{AS},\ LifetimeTicket_{TGS}\}K_{AS,TGS}$

Ticket Granting Server (TGS)

3. $S1,\ Ticket_{TGS},\ Authenticator_{A,TGS}$

with $Authenticator_{A,TGS} = \{A,\ Addr_A,\ t1_A\}K_{A,TGS}$

4. $\{K_{A,S1},\ S1,\ t_{TGS},\ Ticket_{S1}\}K_{A,TGS}$

with $Ticket_{S1} = \{K_{A,S1},\ A,\ Addr_A,\ S1,\ t_{TGS},\ LifetimeTicket_{S1}\}K_{TGS,S1}$

Server (S1)

5. $Ticket_{S1},\ Authenticator_{A,S1}$

with $Authenticator_{A,S1} = \{A,\ Addr_A,\ t2_A\}K_{A,S1}$

6. $\{t2_A + 1\}K_{A,S1}$

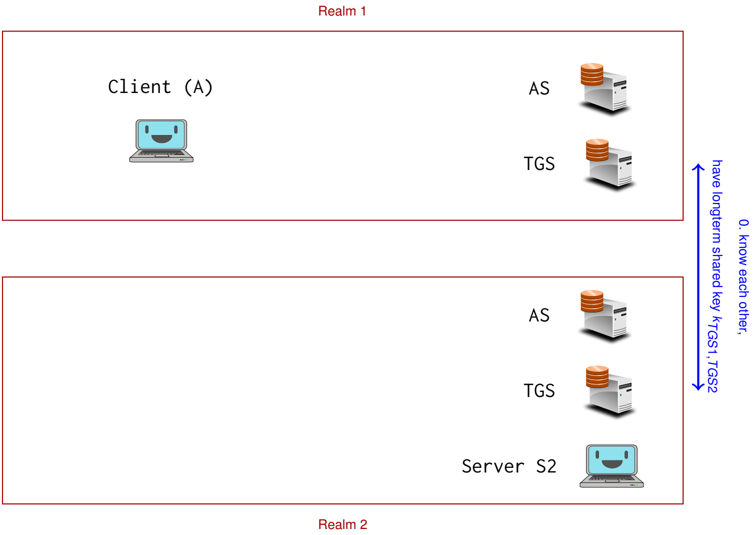
Kerberos – Protocol

- 1. $A \rightarrow AS : A, t_A, TGS, RequestedTicketLifetime_{TGS}$
  - The first message does not use cryptography.
  - Fields are the user name, timestamp, a valid Ticket Granting Server, and the requested lifetime to the tickets.
- 2. $AS \rightarrow A : \{K_{A,TGS}, TGS, t_{AS}, LifetimeTicket_{TGS}, Ticket_{TGS}\}_{K_{A,AS}}$ with $Ticket_{TGS} = \{K_{A,TGS}, A, Addr_A, TGS, t_{AS}, LifetimeTicket_{TGS}\}_{K_{AS,TGS}}$
  - This message is protected with the shared key of A and AS ("password of A"), which is part of the user database of the AS.
  - The first part of the message is information Alice needs to use the ticket, e.g. the new shared key $K_{A,TGS}$ with the TGS.
  - The second part of the message is the ticket, which Alice cannot decrypt or modify.
  - In general, the ticket concepts needs to give the ticket holder enough information to be able to use the ticket. The ticket is protected from the ticket holder. The ticket itself contains similar information, yet for the server that verifies the ticket.

Kerberos – Protocol

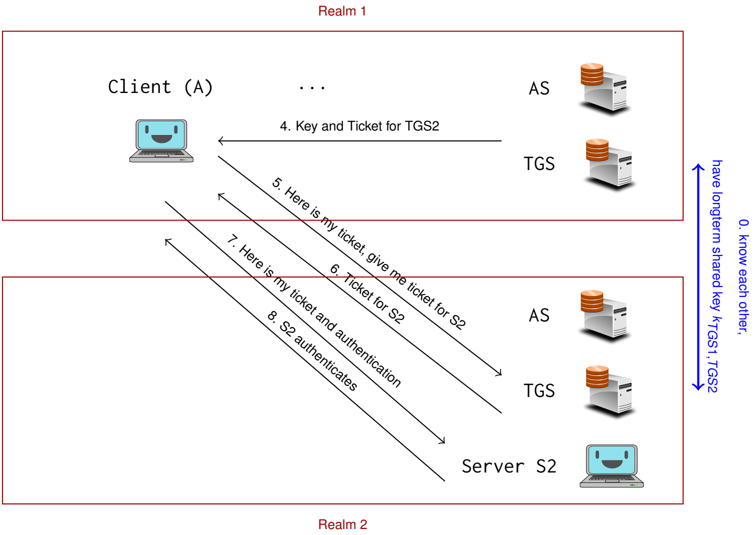
- 3. $A \rightarrow TGS : S1, Ticket_{TGS}, Authenticator_{A,TGS}$ with $Authenticator_{A,TGS} = \{A, Addr_A, t1_A\}_{K_{A,T}}$
  - With the Authenticator Alice shows to the TGS that she is the legitimate ticket holder.
  - She uses the relevant key $K_{A,TGS}$, which is part of the ticket.
  - She has the right IP address $Addr_A$.
  - Authenticator and Ticket are fresh due to fresh enough timestamps.
- 4. $TGS \rightarrow A : \{K_{A,S1}, S1, t_{TGS}, Ticket_{S1}\}_{K_{A,TGS}}$ with $Ticket_{S1} = \{K_{A,S1}, A, Addr_A, S1, t_{TGS}, Life$
  - Similar to 2.
- 5. $A \rightarrow S1 : Ticket_{S1}, Authenticator_{A,S1}$ with $Authenticator_{A,S1} = \{A, Addr_A, t2_A\}_{K_{A,S1}}$
  - Similar to 3.
- 6. $S1 \rightarrow A : \{t2_A + 1\}_{K_{A,S1}}$
  - S1 uses the relevant shared key and answers with Alice's timestamp as nonce. Alice knows she uses the right server.

Kerberos – Multi-Realm



0. know each other,
have longterm shared key $k_{TGS1,TGS2}$

Kerberos – Multi-Realm



Realm 1
Client (A)
. . .
AS
4. Key and Ticket for TGS2
TGS
5. Here is my ticket, give me ticket for S2
6. Ticket for S2
7. Here is my ticket and authentication
8. S2 authenticates
AS
TGS
Server S2
Realm 2
0. know each other,
have longterm shared key $k_{TGS1,TGS2}$

Kerberos – Pre-Authentication Preliminaries

- Weakness Password Authentication
  - Remember message 1: $A, t_A, TGS, RequestedTicketLifetime_{TGS}$
  - Message 1 is not protected. An attacker can request a ticket for someone else. The AS will send message 2 to the attacker.
  - Remember Message 2: $\{K_{A,TGS}, TGS, t_{AS}, LifetimeTicket_{TGS}, Ticket_{TGS}\}_{K_{A,AS}}$ with $K_{A,AS} = h(Password_A)$
  - Now the attacker has ciphertext encrypted with a low-entropy key derived from the password.
  - Attack the key with suitable attack (e.g. dictionary attack)

Kerberos – Pre-Authentication

- Kerberos Pre-Authentication
  - Pre-Authentication is a concept developed for Kerberos[1] to extend the protocol messages (optional).
  - Protocol principles prove their identity before their message is further processed.
  - To avoid the attack above, PA-ENC-TIMESTAMP was proposed in Kerberos 5.
  - Pre-Authentication as generic concept supports all kinds of authentication concepts.
- PA-ENC-TIMESTAMP
  - Add $\{t_A\}_{K_{A,AS}}$ as pre-authentication in message 1.
  - AS will only reply if a current timestamp protected with Alice's key was sent.
  - Thus, ciphertext using key $K_{A,AS}$ will not be sent to the attacker.[2]

[1] IETF RFC 6113

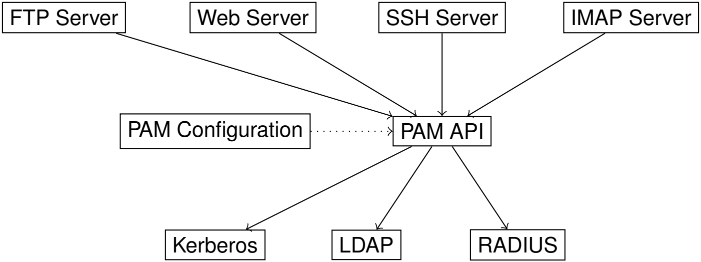
[2] Our Dolev-Yao attacker will see Alice's communication and see such ciphertext nonetheless.

Kerberos – Design Goals?

- Remember the design goals:
    - Security: eavesdroppers or active attackers should not be able to obtain the necessary information to impersonate a user when accessing a service
        - Modern versions of Kerberos use state-of-the-art cryptography, (optional) pre-Authentication helps with password issues
    - Reliability: as every use of a service requires prior authentication, Kerberos should be highly reliable and available
        - The design allows redundant servers and tickets can be reused within their lifetime.
    - Transparency: the authentication process should be transparent to the user beyond the requirement to enter a password
        - Kerberos is a single-sign-on solution. Applications can use tickets within their lifetime.
        - General APIs like PAM help with Kerberos integration in applications.
    - Scalability: the system should be able to support a large number of clients and servers
        - The design allows redundant servers.

Kerberos – Reality Check

- In most environments, Kerberos is used in the backend between Server and Key Distribution Center, but not on client side.
- Pluggable Authentication Modules (PAM) in Linux demonstrate the situation:
  - The services of the system use PAM to do their authentication.
  - PAM provides an API for these services with.
  - Kerberos is one PAM authentication service module, but there are others like LDAP, RADIUS, . . .



Chapter 11: Kerberos and other Frameworks for Client Authentication

Introduction

Learning Goals

Frameworks for Client Authentication

Kerberos

Conclusions – What have we learned

Conclusions – What have we learned

- Authentication Backends
  - User Authentication
  - APIs, Credentials, Transport of Credentials, Protocols
- Kerberos
  - Example for password authentication
  - Ticket Concept
  - Today’s use usually differs from original idea.