Network Security

Network Authentication, Passwords and Kerberos

Kenneth Thilakarathna

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

Intended Learning Outcomes (ILOs)

The lesson introduces you to Passwords and Network Authentication.

After completing this session, related activities, and assignments, you should be able to:

- **LO-01** Describe security threats, mechanisms, protocols and services in computer networks
  - **LO-01-01** Explain network authentication protocols and identify potential vulnerabilities.
  - **LO-01-02** Explore further knowledge on network authentication protocols.

Intended Learning Outcomes (ILOs)

- **LO-02** Analyze and evaluate the implementation and functioning of network applications and decide on their suitability from the security point of view.
  - **LO-02-01**Analyse and evaluate network authentication protocols

Lesson Plan

- Identification vs Authentication
- Activity on identification
- Three Factors in authenticating a user
- Secret verification through direct presentation - Passwords
- Secret verification through result of challenge - SecureID
- Assignment @ home on network authentication
- Network authentication on open networks - Kerberos

Identification vs Authentication

... identifying the user (who he / she is?) by username, email, etc.

... verify the identity, if the user is really who he / she claims to be

User Identification - X.500 Directory

Identification attributes.

- Country
- State
- City
- Address
- Surname
- First Name
- Common Name
- Tel. Number
- Email
- Serial Number
- etc.

Web entity Activity @ In-class (5 Minutes)

Check the X.509 certificate of a web site you access through HTTPS and find the attributes.

User Identification - X.500 Directory

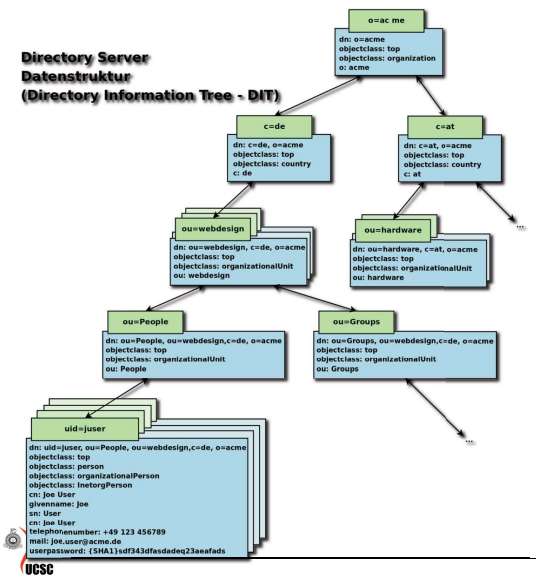
DIT - Directory Information Tree

LDAP - Lightweight Directory Access Protocol

Example Distinguished Name (DN)
CN=kenneth,OU=CIS,O=UCSC,C=LK

DIT Example

Ref: https://de.wikipedia.org/wiki/Datei:Datenstruktur.png



User Authentication

We can authenticate a user based on three (03) factors

- Something the user knows:
  - Day to Day: Knocking pattern to open the door at home
  - In computers: password
- Something the user has:
  - Day to Day: Physical key to the door at home
  - In computers: A smart card
- Something the user is:
  - Day to Day: Calling off with the voice
  - In computers: Fingureprint

How reliable these factors are?

User Authentication

Threats against authentication factors

- Something the user knows:
  - passwords
  - Threat: can be guessed, shared, stolen
- Something the user has:
  - key, smart card
  - Threat: can be stolen
- Something the user is:
  - biometrics
  - Threat: can be copied sometimes. Once copied, you may not have options

Multi-factor authentication

Factors may be combined to reduce the probability of compromization.

ATM has two factor authentication:

- ATM card - something you have
- PIN - something you know

Does two passwords give you two factor authentication? NO

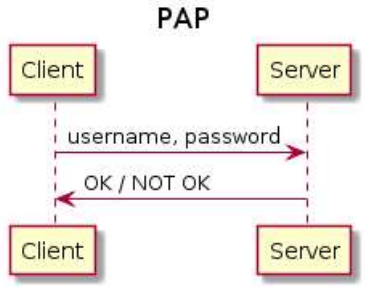
Verification in authentication

Secrets are verified through:

- direct presentation
- result of challenge
- implicit (cryptographical means)

Direct Presentation (PAP)

Password Authentication Protocol:



- Unencrypted, reusable passwords
- Insecure and open network
- Password file must be protected from open access
  - but administrator can see everyone's passwords

Weaknesses of passwords

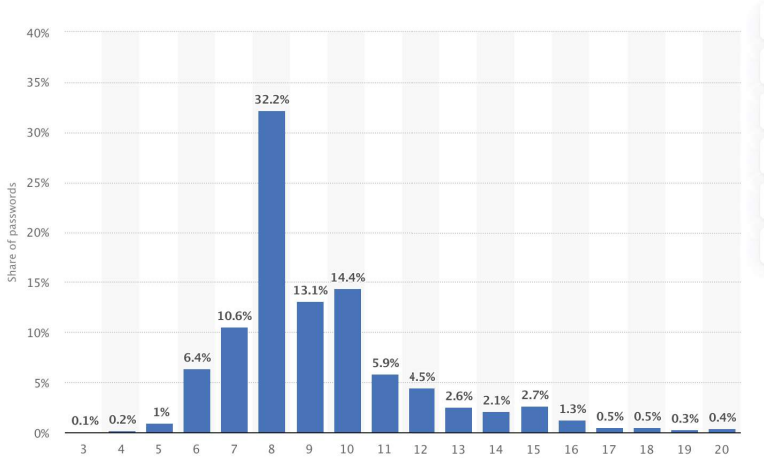
Can be learned by unauthorized people in many ways.

Possibly insecure for a number of reasons

- Obsolete systems - Unix password length is less than 8 characters
- Probability - foo, neo, mom (trying all english words take less than 90 seconds)
- If we check all combinations of eight letters or less we typically account for 50% of all passwords
- Using a GPU computing system, it takes only 5 days or less to bruteforce 8 character password

Password length by number of characters

Average number of characters of leaked user passwords worldwide as of 2017. Ref: statista.com



Problem

Open access to the password file

- What if the password file isn't sufficiently protected and an intruder gets hold of it?
- Even if trusted admin sees your password, this might also be your password on other systems.

Solution:

- Store a hash of the password in a file. Then given a file, you don't get the password directly.
- Have to resort to a dictionary, rainbow table or brute-force attack.
- Example, passwords hashed with SHA-512 hashes (SHA-2)
- salt can be used to guard against rainbow table attacks

Storing passwords

- Plaintext system password lists
  - Need to be heavily protected
  - Would generally require very special implementation in the operating system for the protection
- Encrypted / Hashed system password lists
  - Can be in plain view
  - Need to prevent 1 to 1 mapping so that one user cannot accidentally discover another user's password

Techniques of obtaining passwords

- Try default password used with standard accounts shipped with computer
- Exhaustively try all short passwords
- Try words in directory or a list of likely passwords
- Collect information about users and use these items as passwords: phone number, pet names, spouse name, birth days, etc.
- Use a Troja Horse to bypass restrictions on access
- Tap the line between a remote user and the host system

Password Selection Strategies

- Computer generated passwords
  - users have difficulty remembering them
  - need to write it down
  - have a history of poor acceptance
- Eliminate guessable passwords while allowing the user to select a memorable password

Password Selection Strategies

- Reactive password checking
  - the system periodically runs its own password cracker to find guessable passwords
  - the system cancels passwords that are guessed and notifies user
  - Consumes resources
  - Can be used to victimise users by malicious internal users
  - E.g. John the ripper, Cain and abel, etc. can be used as crackers
- Proactive password checking
  - The system checks at the time of selection if the password is allowable
  - With guidance of the system, users can select memorable passwords that are difficult to guess.

Problem

Network sniffing

- Password can be stolen by observing a user's session in person or over a network.

Solution:

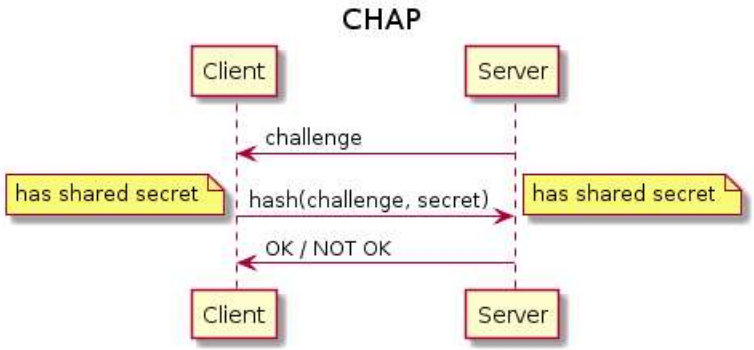
- Use one-time passwords
- Use an encrypted communication channel

One-time password

- Use a different password each time
  - generate a list of passwords
  - use an authentication card (dedicated hw/sw). Usually, keys may be generated using time + initiation key + algorithm

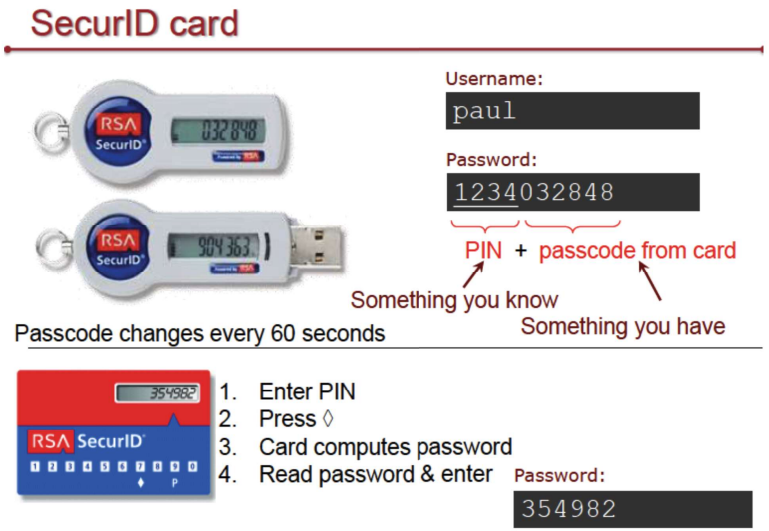
Result of Challenge (CHAP)

Challenge-Handshake Authentication Protocol:



- The challenge is a nonce (random bits)
- We create a hash of the nonce and the secret
- An intruder does not have the secret and cannot do this.

Secure ID



Secure ID

- An intruder (sniffing the network) does not have the information to generate the password for future logins.
  - Needs the seed number (in the card), the algorithm (in the card), and the PIN (from the user)
- An intruder steels your card cannot log in
  - Needs a PIN
  - the benefit of two factor authentication
- An intruder sees your PIN cannot log in
  - Needs the card
  - the benefit of two factor authentication
- BUT ...
  - Vulnerable to man-in-the-middle attacks.
    - attacker acts as the application server.
    - user does not have a chance to authenticate server

What is Kerberos?

A secret key based service for providing user and service authentication to each other in (open) insecure networks.

Developed at M.I.T (US) in the mid 1980s.

There are number of implementations. Mainly:

- Microsoft
- MIT

Kerberos is a trusted third party:

- Knows all (user and services) passwords
- Responsible for:
  - Authentication : validating an identity
  - Authorization : deciding whether someone can access a service
  - Key exchange : giving both parties an encryption key(securely)

Problem

- Access services deployed on a distributed manner throughout an open network (insecure)
- Being able to authenticate requests for services
- Servers being able to restrict access to authorised users
- Users being able to authenticate servers / services

Solution

Require the user to prove his or her identity for each service invoked.

Also require that servers prove their identity to clients.

Challenge: There can be middle parties listening to traffic or probing to get access to services by means such as replay, spoofing as another workstation, etc.

Requirements

- Secure: should not be able to get necessary information to impersonate a legitimate user
- Reliable: Lack of availability of the solution means lack of availability of the services that a legitimate user can access
- Transparent: authentication should be transparent to the legitimate user after the initial successful authentication for a defined period of time. e. g. similar to single sign on
- Scalable: should be able to cater a large number of clients and servers in a distributed network environment.

Solution 01: Do it yourself

Each server keeps track of their clients and handles their authentication

Not scalable

Solution 02: Simple Centralised Authentication Server

$C \Rightarrow AS : ID_C||P_C||ID_V$

$AS \Rightarrow C : Ticket$

$C \Rightarrow V : ID_C||Ticket$

Abbreviations:

$Ticket = E(K_v, [ID_C||AD_C||ID_V])$

$V = Server$

$P_C = Password\ of\ user\ on\ C$

$AD_C = Network\ Address\ of\ C$

$K_v = Secret\ encryption\ key\ shared\ by\ AS\ and\ V$

Problems of Solution 02

- Password goes in plain text
- User will need to enter password for multiple times may be for same service or to access different services
- Ticket can be highjacked and no time constraints thus may be vulnerable to replay attacks

Solution 03: More Secure Centralised Authentication Server

Once per user login session:

$C \Rightarrow AS : ID_C || ID_{tgs}$

$AS \Rightarrow C : E(K_c, Ticket_{tgs})$

Once per type of service:

$C \Rightarrow TGS : ID_C || ID_V || Ticket_{tgs}$

$TGS \Rightarrow C : Ticket_v$

Once per type of service:

$C \Rightarrow V : ID_C || Ticket_v$

Abbriviations:

K_c = *A key derived from C's password such that C can use its password to decrypt a message encrypted by the key*

$Ticket_{tgs} = E(K_{tgs}, [ID_C || AD_C || ID_{tgs} || TS_1 || Lifetime_1])$

$Ticket_v = E(K_v, [ID_C || AD_C || ID_V || TS_2 || Lifetime_2])$

Problems addressed in Solution 03

- Password goes in plain text
- User will need to enter password for multiple times may be for same service or to access different services
- Ticket has time constraints

Problems in Solution 03

- Tickets can be highjacked and used for replay attack for a limited amount of time. Therefore, service need to know if the service request user is same user whom that ticket was issued.
- Additionally, user should be able to verify the service if it is legitimate.

Kerberos Version 4

Once per user login session: AS exchange obtaining TGT (Ticket Granting Ticket)

$C \Rightarrow AS : ID_C||ID_{tgs}||TS_1$

$AS \Rightarrow C : E(K_c, [SK_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Ticket_{tgs}])$

Once per type of service: TGS exchange obtaining SGT (Service Granting Ticket)

$C \Rightarrow TGS : ID_V||Ticket_{tgs}||Authenticator_c$

$TGS \Rightarrow C : E(SK_{c,tgs}, [SK_{c,v}||ID_v||TS_4||Ticket_v)$

Once per type of service: Obtaining service

$C \Rightarrow V : Authenticator_{c1}||Ticket_v$

$V \Rightarrow C : E(K_{c,v}, [TS_5 + 1])$

Abbriviations:

$Ticket_{tgs} = E(K_{tgs}, [SK_{c,tgs}||ID_C||AD_C||ID_{tgs}||TS_2||Lifetime_2])$

$Ticket_v = E(K_v, [SK_{c,v}||ID_C||AD_C||ID_v||TS_4||Lifetime_4])$

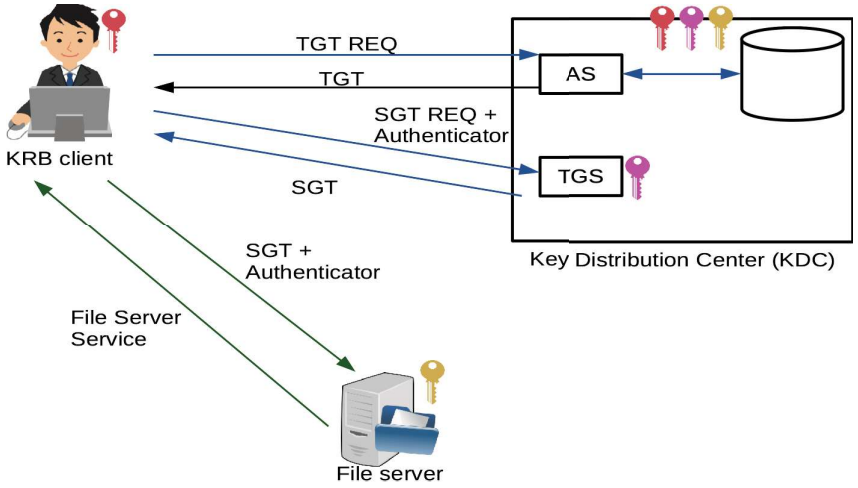
$Authenticator_c = E(SK_{c,tgs}, [ID_C||AD_C||TS_3])$

$Authenticator_{c1} = E(SK_{c,v}, [ID_C||AD_C||TS_5])$

Terms

- AS : Authentication Server
- TGS : Ticket Granting Service
- TGT : Ticket Granting Ticket
- SGT : Service Granting Ticket
- KDC : Key Distribution Centre
- UPN : User Principal Name
- SPN : Service Principal Name

Kerberos Protocol



Kerberos Realm

All users should be registered with KDC and their ID and hashed password should be in Kerberos server.

All services/servers should be registered with KDC and they must share a secret key.

Such an environment is referred as a "Kerberos realm". A Kerberos realm is a set of managed nodes that share the same Kerberos database

Kerberos Principal

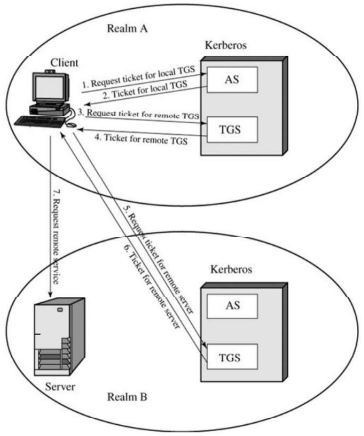
"Kerberos principal", which is a service or user that is known to the Kerberos system. Kerberos principle consist of a service or user name, an instance name, and a realm name

Example: kenneth/PC1.example.com@UC_REALM

Users in one realm can have the services of another realm the Kerberos server in each realm should share a secret key with the server in the other realm. The two Kerberos servers should be registered with each other.

Kerberos remote realm service access

It is achieved by registering remote KDC as a service in local TGS so that local KDC will do the initial authentication and then passes a service request to remote TGS.



Kerberos V4 vs V5

V4 require use of DES encryption where V5 use encryption type identifier so that any encryption technique may be used

V4 require IPv4 where V5 network addresses are tagged with type and length so that any type can be used.

V4, the maximum ticket lifetime is 1280 minutes where in V5 it is made more flexible.

V4 does not support authentication forwarding where V5 supports so that delegated authentication can be done.

In V5, inter-realm authentication is made more scalable

Got rid of double encryption performed in V4 when providing tickets to clients.

Kerberos features in summary

- Avoid sending passwords in clear text through an insecure network.
- Provide "Single-sign-on" capability i.e. use the credential once and access many services.
- Delegated authentication
  - User → Print Server
  - On behalf of user: Print Server → File server to get the file to be printed.
- Secure authentication over insecure network
- Mutual authentication, i.e. allow identification of not only user but also the service
- Centralised user account administration

Kerberos V4 vs V5

Only double encryption changes are depicted
AS exchange obtaining TGT (Ticket Granting Ticket)
v4:

$$AS \Rightarrow C : E(K_c, [SK_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Ticket_{tgs}])$$

V5:

$$AS \Rightarrow C : Realm_c||ID_c||Ticket_{tgs}||E(K_c, [SK_{c,tgs}||ID_{tgs}||Times||Nonce_1||Realm_{tgs}])$$

Once per type of service: TGS exchange obtaining SGT (Service Granting Ticket)
V4:

$$TGS \Rightarrow C : E(SK_{c,tgs}, [SK_{c,v}||ID_v||TS_4||Ticket_v)$$

V5:

$$TGS \Rightarrow C : Realm_c||ID_c||Ticket_v||E(K_c, [SK_{c,v}||ID_v||Times||Nonce_2||Realm_v])$$

$$Ticket_{tgs} = E(K_{tgs}, [...])$$

$$Ticket_v = E(K_v, [...])$$

Kerberos: some details

- Kerberos require reasonably synchronised clocks across the users and services
- Everyone trusts the KDC
- The users key is derived from a password by appending salt (UPN or SPN) and parsing it through a hash function (implementation dependent)
- Session keys are large random numbers

Disadvantages of Kerberos

- Vulnerable to password guessing attacks. Need strong passwords
- Use hardware pre authentication – E.g. SmartCards, HSM

More reading

- PKINIT
  - PKINIT is a pre-authentication mechanism for Kerberos 5 which uses X.509 certificates to authenticate the KDC to clients and vice versa.
  - Ref: http://web.mit.edu/kerberos/krb5-1.13/doc/admin/pkinit.html

References and Mandatory reading

- Stallings Cryptography and Network Security (4th Edition) Chapter 14.1
- Abusing Microsoft Kerberos: https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf