

Kerberos

CPSC/ELEN 689

Philip Mattingly

Mohit Dewan

Overview

- Introduction / History
- Motivation
- Design
- Working / Mechanism
- Weaknesses
- Attacks
- Interoperability
- Conclusion



Introduction

- Kerberos is a trusted third party authentication service for computer networks
- Available as open source or in supported commercial software
- Still under active development

History

- Developed at MIT in the mid 1980's as part of the Athena project
- Name of a 3 headed dog that guards the entrance to the underworld (Hades) – comes from Greek mythology



Motivation

- Problems with password based authentication
 - Insecurity: passwords can be collected by eavesdropping, timing attacks on *ssh* etc.
 - Inconvenient: users don't want to enter a password each time they want to access a network service
- Mutual Authentication: users and servers authenticate each other

Design/Goals

- Network can not be trusted
- Hosts are secure

- User must identify itself once at the beginning of a workstation session (login session)
- Passwords are never sent across the network in clear text (or stored in memory)

- Every user has a password
- Every service has a password
- The only entity that knows all the passwords is the *Authentication Server*

What Kerberos is NOT

- A directory service
- An authorization service
- An auditing service
- An accounting service

Working

■ Primary Components

- Key Distribution Center (KDC)
- Users and services (collectively called *principals*)
- Tickets
- Realms

KDC

- Contains principal keys
 - User keys: Hashed passwords
 - Service keys: random bit-strings
- Provides authentication services by an Authentication Server (AS)
- Provides key distribution functionality by a Ticket Granting Server (TGS)
- Clients and services trust the KDC
 - Foundation of Kerberos security

KDC

- KDCs must be on dedicated, secured machines
- Physical security is important
- Kerberos administrative functions may be performed remotely
- KDC is replicated - one master per realm and $N \geq 0$ slaves
 - Manual intervention needed to turn slave into master, but all data is present on slaves

Principals

- Entity that receives services from the KDC
 - workstation user
 - network service (POP, CVS, FTP)
 - A single server can run multiple services, but each service has its own key
- Each principle shares a secret key with the KDC
 - Used for encrypting & decrypting data
 - Also used to create session keys

Tickets

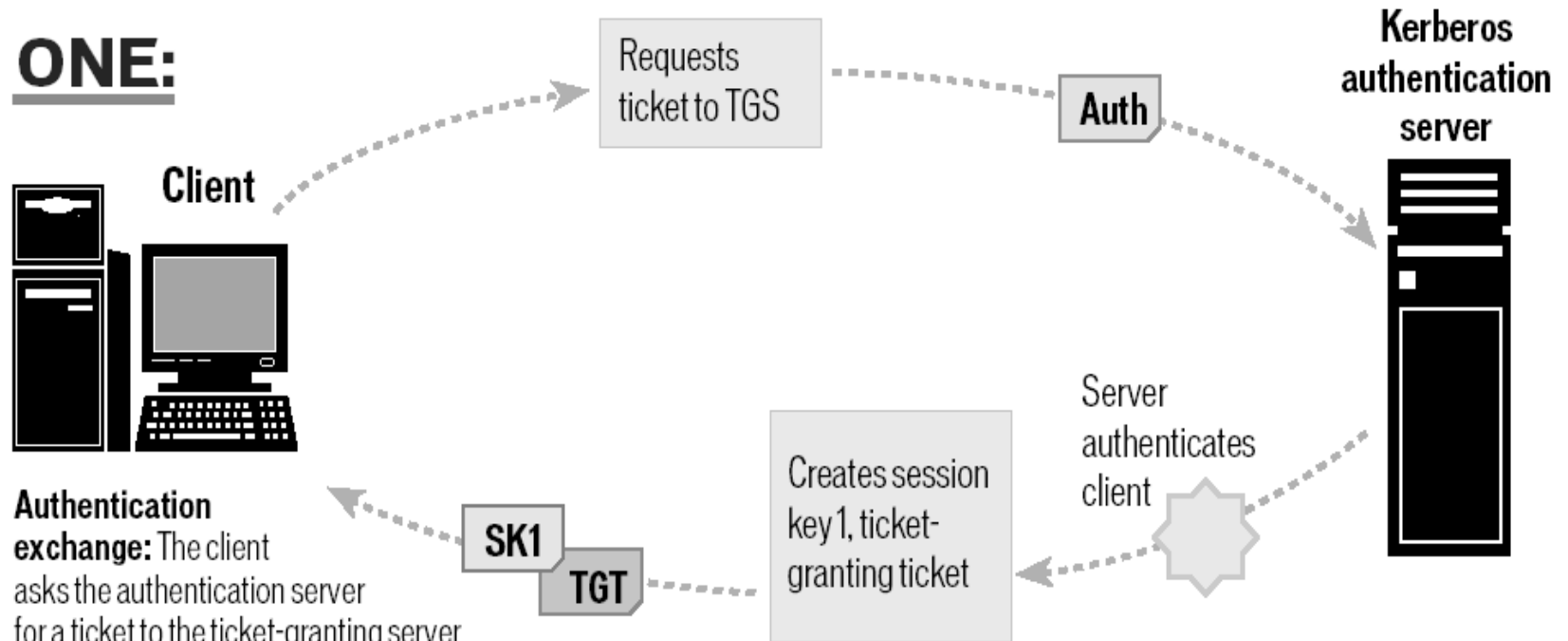
- Each request for a service requires a ticket (from KDC)
- A ticket provides a single client with access to a single server

How Kerberos works

- Uses series of encrypted messages to prove to a verifier (server) that the principal (client) is running on behalf of a particular user
- Incorporates notion of tickets, Ticket Granting Service (TGS), usage of timestamps, approaches to cross-realm authentication etc.

Kerberos – mechanism

ONE:

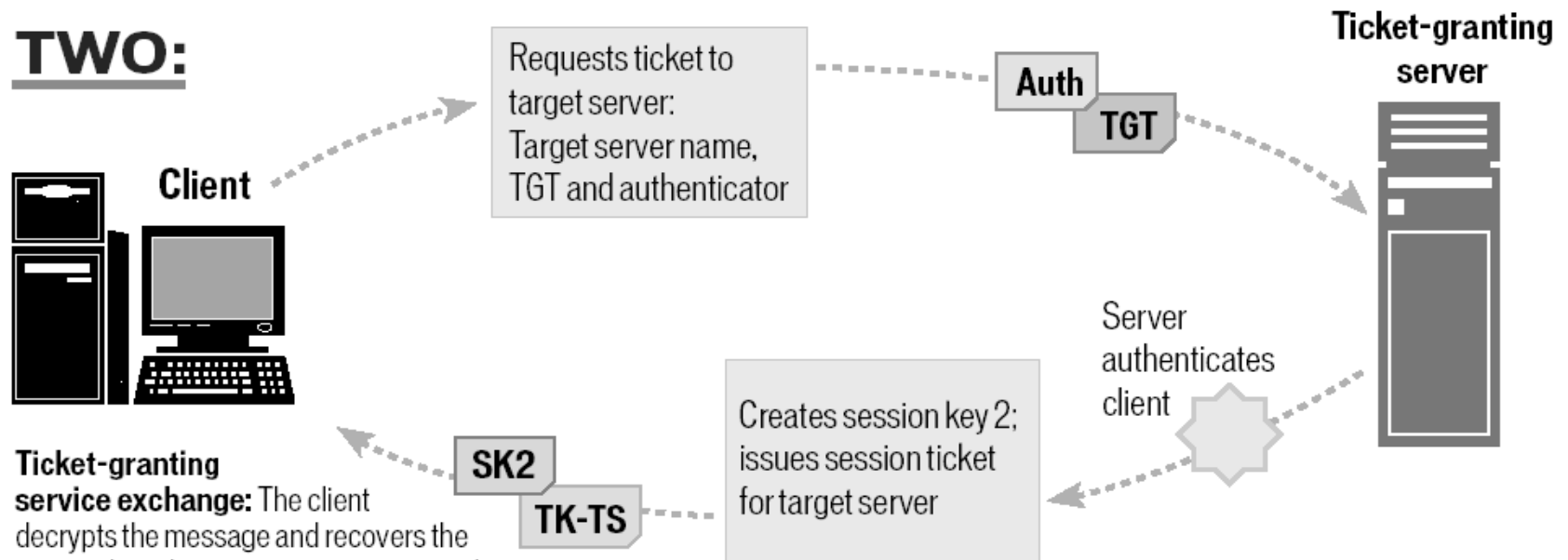


Authentication

exchange: The client asks the authentication server for a ticket to the ticket-granting server (TGS). The authentication server looks up the client in its database, then generates a session key (SK1) for use between the client and the TGS. Kerberos encrypts the SK1 using the client's secret key. The authentication server also uses the TGS's secret key (known only to the authentication server and the TGS) to create and send the user a ticket-granting ticket (TGT).

Kerberos – mechanism

TWO:

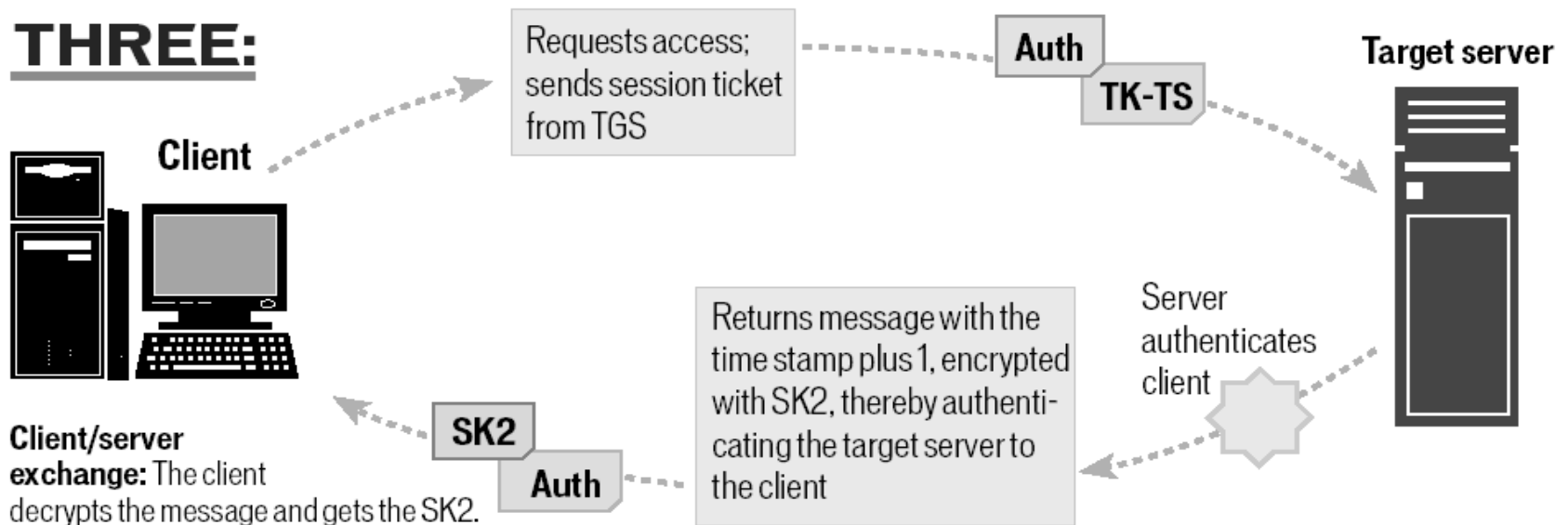


Ticket-granting

service exchange: The client decrypts the message and recovers the session key, then uses it to create an authenticator containing the user's name, IP address and a time stamp. The client sends this authenticator, along with the TGT, to the TGS, requesting access to the target server. The TGS decrypts the TGT, then uses the SK1 inside the TGT to decrypt the authenticator. It verifies information in the authenticator, the ticket, the client's network address and the time stamp. If everything matches, it lets the request proceed. Then the TGS creates a new session key (SK2) for the client and target server to use, encrypts it using SK1 and sends it to the client. The TGS also sends a new ticket containing the client's name, network address, a time stamp and an expiration time for the ticket – all encrypted with the target server's secret key – and the name of the server.

Kerberos – mechanism

THREE:



Client/server

exchange:

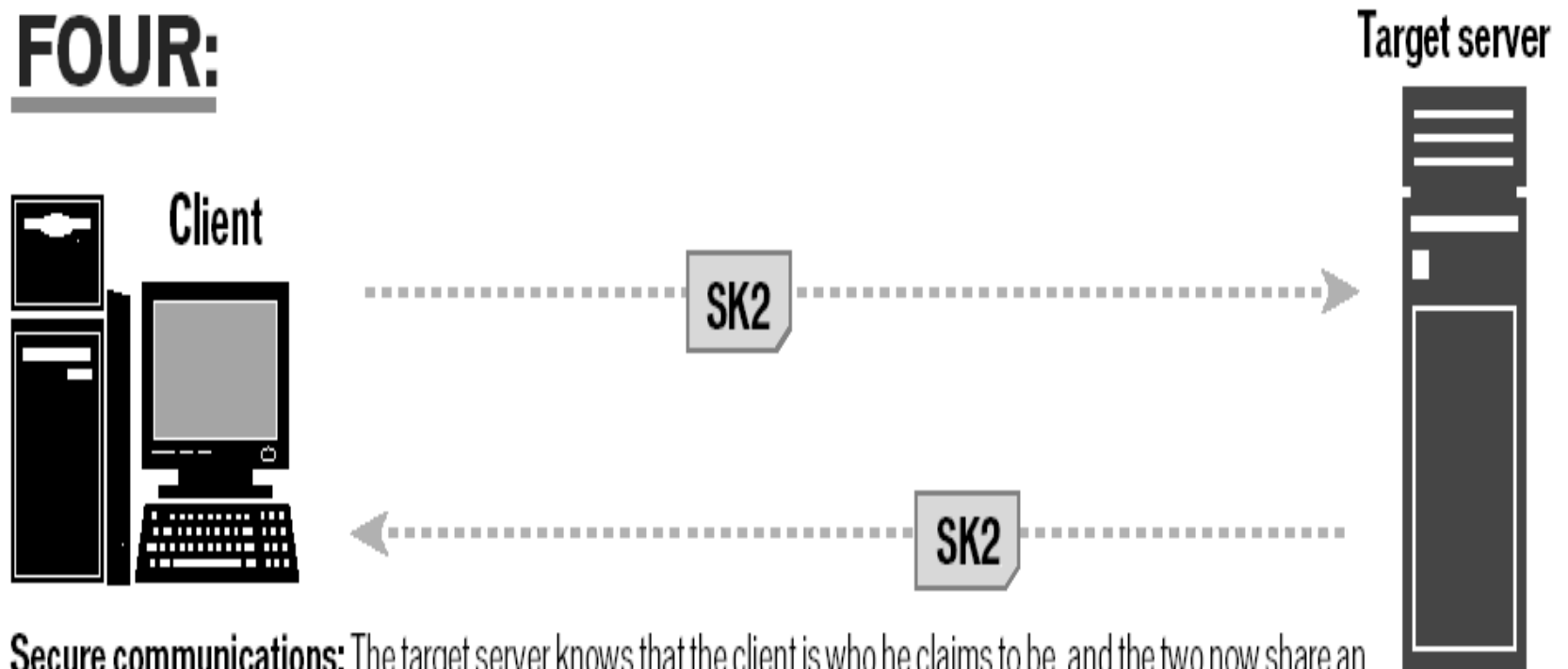
The client decrypts the message and gets the SK2.

Finally ready to approach the target server, the client

creates a new authenticator encrypted with SK2. The client sends the session ticket (already encrypted with the target server's secret key) and the encrypted authenticator. Because the authenticator contains plaintext encrypted with SK2, it proves that the client knows the key. The encrypted time stamp prevents an eavesdropper from recording both the ticket and authenticator and replaying them later. The target server decrypts and checks the ticket, authenticator, client address and time stamp. For applications that require two-way authentication, the target server returns a message consisting of the time stamp plus 1, encrypted with SK2. This proves to the client that the server actually knew its own secret key and thus could decrypt the ticket and the authenticator.

Kerberos – mechanism

FOUR:



Secure communications: The target server knows that the client is who he claims to be, and the two now share an encryption key for secure communications. Because only the client and target server share this key, they can assume that a recent message encrypted in that key originated with the other party.

Observations

- Kerberos depends on authenticators and tickets.
- Tickets held on a workstation cannot be decrypted on the workstation
- Tickets are associated with session keys that are generated by a random mechanism when the ticket is generated.
- Tickets are reusable, tickets usually have a lifetime of 8 hours. Authenticators are not reusable and must be created afresh for each connection to a server.
- A server history of recent requests should result in the detection of duplicate requests due to stolen tickets.
- Tickets and authenticators contain network addresses so stolen tickets can't be used elsewhere without address spoofing.

Cross Realm Authentication

- Principals in one realm can authenticate to principals in another realm.
- The two realms share a special cross-realm secret
- Version 5 supports **transitive** cross-realm authentication
- Realms in which you share a cross-realm secret with cannot acquire a ticket for a user in your local realm.
 - Defaults to NOT trusting any foreign realms

Weaknesses

- Does not provide Authorization
 - only Confidentiality & Integrity
- KDC is a single point of failure
- Authentication Service on the KDC must be able to handle request in timely manner
- Secret keys are temporarily store on end-user workstation

Weaknesses

- Open Architecture
- Kerberos is vulnerable to password guessing
 - Dictionary attack
- Network traffic is not protected
 - Session replay
- Time based

Security Threats

■ Eavesdropping

- obtaining data without authority

■ Masquerading

- attacks sent and received using the false identity of a legitimate user

■ Message Tampering

- stops a legitimate message

Attacks – Denial of Service

- Simple replay attacks
 - Resource starvation
 - Simple denial of service
- Buffer Overrun and Under run
 - Segment that processed principal names.
 - Coupled with certain platforms and implementations of *malloc* it would bring down KDC.
 - As recent as 3/21/2003

Attacks

- Chosen Plain-Text
 - Cryptographic weakness in implementation
 - During the seeding of the crypto functions
- Cut-N-Paste
 - Allowed a client principal to fabricate a krb4 ticket
 - In the triple-DES functions
 - Works in a krb5 site with backwards compatibility

Attacks

■ Realm Hopping

- Usually does not trust multiple hop cross realm authentication
- Trust can not be transitive

■ Ticket Splicing

- If client principal names are chosen carefully then this session key will line up with a DES block boundary
- Attacker has legitimate ticket t1. Then captures ticket t2 from other principal.
Creates a new tickets t2 by replacing t2's session key block from t1

Upgrades

- *key salt* algorithm has been changed to use the entire principal name.
 - Kerberos 4 it was never used
 - key compromise in one realm would result in a key compromise in the other realm.
 - Kerberos 5 the complete principal name including the realm is used as the salt
 - the same password will not result in the same encryption key in different realms or with two different principals in the same realm

Upgrades

- The network protocol has been completely redone and now uses ASN.1
- There is now support for forwardable, renewable, and postdatable tickets
- A generic crypto interface module
 - Other encryption algorithms beside DES can be used
 - There is now support for replay caches, so authenticators are not vulnerable to replay

The Competition: SSL

| SSL | Kerberos |
|--|---|
| Uses public key encryption | Uses private key encryption |
| Is certificate based (asynchronous) | Relies on a trusted third party (synchronous) |
| Ideal for the WWW | Ideal for networked environments |
| Key revocation requires Revocation Server to keep track of bad certificates | Key revocation can be accomplished by disabling a user at the Authentication Server |
| Certificates sit on a users hard drive (even if they are encrypted) where they are subject to being cracked. | Passwords reside in users' minds where they are usually not subject to secret attack. |
| Uses patented material, so the service is not free. Netscape has a profit motive in wide acceptance of the standard. | Kerberos has always been open source and freely available. |

Interoperability

■ LDAP

- Designed to distribute information
- Acts as a naming and authorization service

■ AFS (Andrew File System) support

■ Windows Active Directory

■ Crypto-card support

Conclusion

- Deployed correctly and in conjunction with other tools Kerberos can be an enterprise solution for a secure environment.

Questions

- Those of you with questions, please ask them
- The rest of you may watch a dancing monkey



References

- <http://web.mit.edu/kerberos/>
- Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition, by Bruce Schneier (Wiley, 1995)
- Strong Authentication – System Design and Deployment: Matt Crawford, Fermilab, Computer Security Team
- <http://www.scit.wlv.ac.uk/~jphb/comms/kerberos.html>
- Kerberos: Jean-Anne Fitzpatrick, Jennifer English
- <http://www.kerberos.isi.edu/>
- <http://www.ornl.gov/~jar/CISreport.html>
- <http://www.securiteam.com/unixfocus/5IP0E2K9FE.html>
- <http://www.saintcorporation.com>