



# Kerberos Project

*A technical presentation on Kerberos and the  
ROSPA Meta-network*

Tillman Hodgson

tillman@seekingfire.com

ROSPA



# INTRODUCTION

# What's this presentation about?

- I'm here to talk about the ROSPA Kerberos project
- Topics I'll cover:
  - A 5 minute security primer
  - What Kerberos is
  - What Kerberos does
  - How Kerberos works: The analogy
  - How Kerberos works: The gritty details
  - How we implemented Kerberos
  - How you can get involved



# The Four Horsemen of Security

- Network security can be divided into 4 topics:
  - *Authentication*: deals with confirming the identity of a communication partner
  - *Authorization*: has to do with restricting access to information by unauthorized entities
  - *Non repudiation*: deals with signatures, it uniquely identifies the sender of a message or file
  - *Integrity control*: assures that vital data has not been modified
- Kerberos deals *only* with authentication

# Who are Alice and Bob?

- Cryptographers traditionally used the names Alice and Bob to refer to two parties who want to communicate securely over an insecure channel
- Alice wants to manage her money, and Bob is her banker
- Eve is someone who wishes to intercept and read their communications
- Note that Alice and Bob have never met and don't trust each other

# What is authentication?

- Authentication is the act of verifying the identity of the communication partner
  - Alice wishes to deal with Bob, her banker
  - In real life Bob and Alice can authenticate each other by recognizing faces, voices or handwriting . . . over network none of these options are available
  - Authentication: How can Bob be sure that the request to transfer all of Alice's money to a secret Swiss bank account came from Alice and not from Eve?

# Secrets and Keys

- Kerberos uses secret-key rather than public-key cryptography
- In secret-key cryptography, Alice and Bob must share a secret
- Another word for this shared secret is "password" or "key"
- This secret both proves their identity and forms a seed for encryption



# WHAT KERBEROS IS

# Description

- Kerberos is a cross-platform secure network authentication protocol
- The protocol is currently at version 5, described in RFC 1510
- Kerberos provides authentication for client/server applications (such as FTP and telnet) by using secret-key cryptography
- Kerberos can be described as a identity-verifying proxy or as a trusted third-party authentication system

## Description (cont)

- The name Kerberos comes from Greek mythology; it is the three-headed dog that guards the entrance to Hades
- Kerberos was created by MIT, as part of the Athena project, as a solution to network security problems
- Several free implementations of this protocol have been developed, including the original MIT and the newer Heimdal



# WHAT KERBEROS DOES

# What can it do?

- Kerberos provides only one function — the secure authentication of users on the network
- Within that one function it solves three problems:
  - Allows a single, centralized, password store
  - Prevents passwords from being transmitted over the network
  - Frees users from repeatedly authenticating every time they use a service

## What can it do? (cont)

- Allow a client and a server to *mutually* authenticate each other
  - This prevents server/service spoofing
  - Most other authentication systems don't do this
- Optionally allows all data traffic to be encrypted
  - This means that traditionally insecure applications like `rsh`, `rcp`, `telnet`, `ftp` (and others) are at least as safe as SSH!

# What can't it do?

- Kerberos does *not* provide:
  - Authorization functions (what actions those users are able to perform)
  - User meta information like home directories, UID and GID
  - Auditing functions (which user did what and when)

# What about Authorization?

- Unlike other centralized authentication services (LDAP, NIS, NIS+, etc), Kerberos deals only with authentication
- Kerberos does not provide meta information like home directories, UID and GID
- Kerberos is generally combined with another tool such as LDAP, NIS, /etc/passwd, etc
- The various combinations are powerful because Kerberos does the security aspect of authentication very well and a choice in Authorization tools can be very flexible



# HOW KERBEROS WORKS: THE ANALOGY

# The Scenario

- Alice wants to access her bank accounts
  - Alice represents a client application, such as `telnet`
- Bob is the Alice's bank manager
  - Bob represents a server application, such as `telnetd`
- Eve is a villain who wants to transfer Alice's money to her own account
- Ken D. Cristoff is a mutual friend of Alice and Bob
  - Ken represents a trusted authentication service, such as a Kerberos key distribution center (KDC)
- Alice, Bob, Eve and Ken are all *very* good at math

# Trouble at the Bank

- Alice wants to be able to call Bob to access her account, but:
  - *Only* Alice should be able to access the account
  - The only phone she has access to is very noisy and she can't hear his voice clearly
  - Alice is lazy and doesn't want to authenticate every time
- Alice wants to make sure that she's really talking to Bob (and not Eve)
- Bob wants to make sure that he's really talking to Alice (and not Eve)

# The First Attempt

- Alice and Bob decide that they need to share a secret, a password
  - Every time Alice calls Bob, she must say a password for Bob to allow access to the bank account
- Eve answers Bobs phone and pretends to be Bob
  - Alice gives her password, and Eve can now access Alice's accounts

# The First Attempt: Analysis

- Critical failure: It's not *mutual* authentication, Alice has no way of knowing that it was Bob who answered the phone
- This is how traditional services like `telnet` work, spoofing the service is fairly trivial

# The Second Attempt

- Alice and Bob decide that they need to share two secrets
  - Every time Alice calls Bob, Bob must give Alice his password before she'll give hers
- Eve calls Bob pretending to be Alice and gets Bobs password; she then answers Bobs phone when Alice calls and gives Bobs password to Alice — Alice gives Eve her password (believing it to be Bob), and Eve can now access Alice's accounts

# The Second Attempt: Analysis

- Critical failure: The passwords are given “in the clear”, they’re vulnerable to being stolen
- Clear-text passwords are how most traditional services work, and they’re vulnerable to packet sniffers and other tools
- On an ethernet network, where the communications medium is shared, it’s like an old telephone party line — any number of people can be listening in on the conversation

# The Third Attempt

- Alice and Bob decide that they need to use encryption when exchanging the password
  - Every time Alice calls Bob, Bob encrypts Alice's name and a random number using the password
  - Alice then decrypts it using the password, and if she gets her name as the result she knows that it really is Bob
  - Alice sends Bob the random number... since only she could have obtained it, Bob knows it really is Alice
- Alice is lazy and doesn't want to do this every time she calls Bob; she also deals with other banks and doesn't want a different password for every bank

# The Third Attempt: Analysis

- Critical failure: The authentication scheme isn't a persistent and it's not centralized (i.e., it's not single-sign-on)
- This is how Challenge-Response authentication systems work
- Danger: a user with many passwords will resort to writing them down in order to remember them (Eve would love that)
- Danger: a user may use the same password with every service... what if Eve is a manager at another bank that Alice uses?

# The Fourth Attempt

- Alice and Bob decide to get Ken involved
  - Ken shares a secret with Alice, and a secret with Bob
  - Alice calls Ken and authenticates as in the 3rd example, Ken gives Alice a ticket encrypted with Bob's passwords
  - Alice calls Bob and reads the ticket to him; Bob decrypts the ticket with his password, if successful he knows that Ken vouches for Alice (and Alice knows that only Bob could decrypt the ticket)
  - If Alice calls Bob again, she gives him the same ticket
  - If Alice wants to call another bank, she calls Ken and obtains another ticket

# The Fourth Attempt: Analysis

- This is similar to how Kerberos works
- There is still some issues:
  - What if some banks don't support Kerberos?
  - What if Ken can't be trusted?
  - What if Alice does something silly (like writing her password down)?
- A vulnerability that Kerberos must address is: What if Eve hears the ticket and simply parrots it to Bob? This is called a replay attack.

# Preventing Replay Attacks

- Preventing replays requires a small change in our method:
  - When Ken encrypts a ticket for Alice, he gives her a *session key* too
  - Ken also gives Alice a copy of the session key encrypted with Bobs password
  - When Alice calls Bobs, she encrypts her ticket *along with the current time* using the session key and gives Bob that *as well as* his copy of the session key
  - Bob decrypts his copy of the session key, and then uses the session key to decrypt the ticket

# Preventing Replay Attacks (cont)

- Preventing replays requires a small change in our method:
  - Bob keeps track of the timestamps he's seen, and won't allow repeats
  - If Alice calls Bob again, she gives him the same ticket but has to reencrypt it with the session key and the time each time
  - If Alice wants to call another bank, she calls Ken and obtains another session key



# HOW KERBEROS WORKS: THE GRITTY DETAILS

# Introduction

- The following description of how Kerberos works was taken from The Moron's Guide to Kerberos.
- Alice is the user, Bob is the service, Ken is the KDC
- Both the user and the service are required to have keys registered with the KDC
- The user's key is derived from a password that he chooses; the service key is randomly selected
- Imagine that messages are written on paper, and are "encrypted" by being locked in a box by means of a key

# Tips

- While following this, continually ask yourself:
  - Who encrypted (locked) a particular ticket (box)?
  - Who can decrypt (unlock) it?
  - How can you know that the ticket (box) isn't an illegitimate copy?
  - Every time a ticket (box) travels across the network, what would happen if somebody saw it?

# Logging onto to the network

- In order to “log onto the network”, or use Kerberos as a single-sign-on system, Alice needs to get a special Ticket Granting Ticket (TGT) from the KDC
- To start off, Alice asks the KDC for a TGT

# The KDC processes the TGT request

- The KDC looks up the user (Alice) in its database, then generates a session key for use between the user and itself
- It then encrypts the session key with the user's password (creating the TGT)
- It then sends the encrypted session key to the user
- The user can now talk to the KDC securely at will

# Requesting access to a service

- First the user (Alice) sends a message to the KDC: “I, Alice, would like to talk to the server named Bob”
- When the KDC receives this message, it makes up two copies of a *brand new* key called the session key, which will be used in the direct exchange between user and service
- Note that session keys are create new each time they are needed

# Building the request

- The KDC puts one of the session keys in box 1, along with a piece of paper with the name “Bob” written on it
- It locks box 1 with Alice’s key (so only Alice can open it)
- It puts the other session key in box 2, along with a piece of paper with the name “Alice” on it
- It locks box 2 with Bob’s key (so only Bob can open it)
- It gives both boxes to Alice

# Checking the ticket

- Alice unlocks box 1 with her key, extracts the session key and the paper
- Note that Alice can't open box 2
- Note that box 2 is often called a *ticket*
- Alice checks that the paper from box 1 has the name of the service she asked for (if it doesn't, the KDC and her aren't agreeing on the key)
- If the paper checks out, Alice trusts the session key and is ready to contact Bob

# Using the ticket: Authenticators

- Alice puts a piece a paper with the current time on it into a new box (box 3) and locks it with the session key
  - Note that box 3 is often called an *authenticator*
- Alice gives both boxes 2 and 3 to Bob
  - Bob opens box 2 with his key and gets the paper with the name “Alice” as well as the session key
  - Bob uses the session key to open box 3 and extract the current time
  - Bob now knows that Alice really is Alice and that the ticket wasn't replayed

## And she's in!

- Bob is satisfied that the request came from Alice, and let's her in!
- Since Alice and Bob share a secret session key, they can use it to encrypt all the data traffic as well



# QUESTIONS?



# GETTING INVOLVED

# Getting Involved ...

- Visit us at <http://www.rosipa.ca> and check out the Projects section
- Sign up for the projects mailing list
- It really helps to be on the ROSIPA VPN Meta-Network
- Start using Kerberos!