Security issues in Distributed Systems

Is Kerberos the Answer?

Types of Distributed Systems
- There are many different types of distributed computing systems and many challenges to overcome in successfully designing one.
- Ideally, this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems.

Architecture of Distributed Systems
- **Client-server** — Smart client code contacts the server for data, then formats and displays it to the user. Input at the client is committed back to the server when it represents a permanent change.

Architecture of Distributed Systems
- **3-tier architecture** — Three tier systems move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are 3-Tier.
**Architecture of Distributed Systems**

- **Tightly coupled (clustered)**
  
  — Refers typically to a set of highly integrated machines that run the same process in parallel, subdividing the task in parts that are made individually by each one, and then put back together to make the final result.

---

**Properties of Distributed Systems:**

**Openness**

- Openness is the property of distributed systems such that each subsystem is continually open to interaction with other systems, i.e. clients or servers. Web Services protocols are standards that enable distributed systems to be extended and scaled. In general, an open system that scales has an advantage over a perfectly closed and self-contained system.

- Consequently, open distributed systems are required to meet the following challenges:

---

**Properties of Distributed Systems:**

**Openness**

- Monotonicity: Once something is published in an open distributed system, it cannot be taken back.

- Pluralism: Different subsystems of an open distributed system include heterogeneous, overlapping and possibly conflicting information. There is no central arbiter of truth in open distributed systems.

- Unbounded nondeterminism: Asynchronously, different subsystems can come up and go down and communication links can come in and go out between subsystems of an open distributed system. Therefore the time that it will take to complete an operation cannot be bounded in advance.

---

**Properties of Distributed Systems:**

**Scalability**

- A scalable system is one that can easily be altered to accommodate changes in the number of users, resources and computing entities affected by it. Scalability can be measured in three different dimensions.
Properties of Distributed Systems: 

**Scalability**

- **Load scalability** — A distributed system should make it easy for us to expand and contract its resource pool to accommodate heavier or lighter loads.
- **Geographic scalability** — A geographically scalable system is one that maintains its usefulness and usability, regardless of how far apart its users or resources are.
- **Administrative scalability** — No matter how many different organizations need to share a single distributed system, it should still be easy to use and manage.
- Note: Some loss of performance may occur in a system that allows itself to scale in one or more of these dimensions.

Security Issues in Distributed Systems

- As computer models began to evolve from central system/dumb terminal to client/server researchers at MIT realized there were a whole new set of issues to resolve.
- Network users were now able to cause mischief, since they now controlled at least a portion of the computer power.
- One of the most important issues they found was the passing of passwords across a network in plaintext.

Kerberos

- The Kerberos protocol was the result of the work on project Athena. The earliest versions were limited to internal use at MIT, mainly for testing and interpretation of the results of development (and testing). Kerberos 4 was the first officially distributed release. It was made available to the public in 1989. Kerberos 5 is the most current version available.

Key Terms Related to Kerberos

- A realm defines the scope of administrative control. It can be thought of as similar to a network domain.
- Principals are associated with all entities within a Kerberos installation. These entities include users, computers, and services (running on application servers). These principals are unique across the installation, and each is associated with a long-term key. A password may be a long-term key. A principal is made up of a username (for users) or a service name (for services), followed by an instance (optional), and the realm name.
Key Terms Related to Kerberos

- The Key Distribution Center, or KDC, is the main piece of the Kerberos puzzle. Each Kerberos realm must have at least one KDC. When using multiple KDCs, it is imperative that they be kept synchronized. Because some of the Kerberos authentication relies on timestamps, KDCs that are out of sync may deny access when the credentials are legitimate. Three logical components make up the KDC—a database containing all of the principals and their keys, the Authentication Server, and the Ticket Granting Server. These three components are most often run together as a single process.

Key Terms Related to Kerberos

- **Authentication Server (AS)**
  - The Authentication Server receives requests to log in to the realm. No password is sent to the AS, as it retrieves password information from the KDC database containing users and passwords. The AS is responsible for creating a Ticket Granting Ticket and sending it back to the client for decryption. Authentication occurs when the client successfully decrypts the message returned by the AS.

- **Ticket Granting Server (TGS)**
  - The TGS provides tickets to the client that are used to access services. The TGS receives two pieces of information in a request from a client—a ticket request including the service principal’s name, and the TGT previously returned by the AS. The TGS also knows about the TGT key and uses it to authenticate the TGT received. If authenticated, the TGS returns a service ticket to the client.

Key Terms Related to Kerberos

- **Ticket**
  - A ticket is a confirmation of a user’s identity. A ticket contains a lot of data, including a session key (an encryption key unique for each session), the user’s principal, a service principal, the ticket valid start time and expiration, and a list of IP addresses the ticket may be used from. The KDC itself provides the session key and the ticket expiration. Ticket expirations created by the KDC are typically ten hours to one day after the valid start time. This way, users do not have to repeatedly log on during the course of a working day, and lengthy attacks using compromised tickets are limited.

- **Credential Cache**
  - Tickets are stored differently dependent upon the specific implementation. The default is to store them in a file, however this is very insecure. Both Microsoft and Apple implementations of Kerberos use a credential cache that is memory based. This forces the credential cache to be destroyed when the login session has concluded.

What’s Kerberos

- Before discussing specific problem areas, it is helpful to review Kerberos Version 4. Kerberos is an authentication system; it provides evidence of a principal’s identity. A principal is generally either a user or a particular service on a home machine. A principal consists of the three-tuple:
  - `<primary name, instance, realm>`.

- If the principal is a user—a genuine person—the primary name is the login identifier, and the instance is either null or represents particular attributes of the user, i.e., root. For a service, the service name is used as the primary name and the machine name is used as the instance, i.e., login.myhost. The realm is used to distinguish among different authentication domains; thus, there need not be one giant—and universally trusted—Kerberos database serving an entire company.

- Kerberos principals may obtain tickets for services from a special server known as the ticket granting server, or TGS. A ticket contains assorted information identifying the principal, encrypted in the private key of the service.
A Typical Ticket in Kerberos

\[\{Tc,s \}Ks = \{s, c, addr, timestamp, lifetime, Kc,s \}Ks\]
- \(c\) client principal
- \(s\) server principal
- \(tgs\) ticket-granting server
- \(Kx\) private key of ‘x’
- \(Kc,s\) session key for ‘c’ and ‘s’
- \(\{\text{info}\}\)Kx info encrypted in key Kx
- \(\{Tc,s \}Ks\) Encrypted ticket for ‘c’ to use ‘s’
- \(\{Ac \}Kc,s\) Encrypted authenticator for ‘c’ to use ‘s’
- \(addr\) client’s IP address

What All This Means

- Since only Kerberos and the service share the private key \(Ks\), the ticket is known to be authentic. The ticket contains a new private session key, \(Kc,s\), known to the client as well; this key may be used to encrypt transactions during the session.
- To guard against replay attacks, all tickets presented are accompanied by an authenticator:
  \[\{Ac \}Kc,s = \{c, addr, timestamp\}Kc,s\]
- This is a brief string encrypted in the session key and containing a timestamp; if the time does not match the current time within the (predetermined) clock skew limits, the request is assumed to be fraudulent.

What All This Means

- For services where the client needs bidirectional authentication, the server can reply with
  \[\{timestamp + 1\}Kc,s\]
- This demonstrates that the server was able to read \(timestamp\) from the authenticator, and hence that it knew \(Kc,s\); that in turn is only available in the ticket, which is encrypted in the server’s private key.

What All This Means

- Tickets are obtained from the TGS by sending a request
  \(s, \{Tc.tgs \}Ktgs, \{Ac \}Kc.tgs\)
- In other words, an ordinary ticket/authenticator pair is used; the ticket is known as the ticket-granting ticket (TGT). The TGS responds with a ticket for server \(s\) and a copy of \(Kc,s\), all encrypted with a private key shared by the TGS and the principal:
  \[\{\{Tc,s \}Ks .Kc,s \}Kc.tgs\]
- The session key \(Kc,s\) is a newly-chosen random key. The key \(Kc.tgs\) and the ticket-granting ticket itself, are obtained at session-start time.
What All This Means

- The client sends a message to Kerberos with a principal name; Kerberos responds with
  \[ \{Kc, tgs \} \{Tc.tgs \} Ktgs \} \]
- The client key \( Kc \) is derived from a non-invertible transform of the user’s typed password. Thus, all privileges depend ultimately on this one key. Note that servers must possess private keys of their own, in order to decrypt tickets. These keys are stored in a secure location on the server’s machine.

The Million Dollar Question

- You must be wondering, if all this takes place before a session starts, how can there be any limitations to Kerberos?
- How can someone compromise security during a session?

Kerberos Limitations

- Although Kerberos was developed to resolve all the issues related to the password-based authentication, many studies reveals that Kerberos is not altogether one hundred percent immune against other form of attacks.

Various Forms of Attacks

- Replay Attack
- Secure Time Services
- Password Guessing Attacks
- Spoofing Login
- Inter-Session Chosen Plain Texts Attacks
Replay Attack

- Kerberos is susceptible to replay attacks. This weakness stem from the use of an authenticator to prevent the attack.

Secure Time Services

- Authenticators rely on machines’ clocks being roughly synchronized. If a host can be misled about the correct time, a stale authenticator can be replayed without any trouble at all. Since some time synchronization protocols are unauthenticated and hosts are still using these protocols despite the existence of better ones, such attacks are not difficult. Spoofing an unauthenticated time service may be a difficult programming task, it is not cryptographically difficult.

Password Guessing Attacks

- A second major class of attack on the Kerberos protocols involves an intruder recording login dialogs in order to mount a password-guessing assault. When a user requests Tc,tgs (the ticket-granting ticket), the answer is returned encrypted with Kc, a key derived by a publicly-known algorithm from the user’s password. A guess at the user’s password can be confirmed by calculating Kc and using it to decrypt the recorded answer. An intruder who has recorded many such login dialogs has good odds of finding several new passwords; empirically, users do not pick good passwords unless forced to

Spoofing Login

- In a workstation environment, it is quite simple for an intruder to replace the login command with a version that records users’ passwords before employing them in the Kerberos dialog. Such an attack negates one of Kerberos’s primary advantages, that passwords are never transmitted in cleartext over a network.
Inter-Session Chosen Plain Texts

Attacks

- A chosen plaintext attack is one where an attacker may choose all or part of the plaintext and, typically, use the resulting cipher text to attack the cipher. Specifically, the encrypted portion of messages of this type have the form:

\[ X = (\text{DATA}, \text{timestamp + direction}, \text{hostaddress}, \text{PAD}) \]

- Since cipher-block chaining (FIPS81, Dave89) has the property that prefixes of encryptions are encryptions of prefixes, if \( \text{DATA} \) has the form

\[ (\text{AUTHENTICATOR}, \text{CHECKSUM}, \text{REMAINDER}) \]

- Then a prefix of the encryption of \( X \) with the session key is the encryption of \( (\text{AUTHENTICATOR}, \text{CHECKSUM}) \), and can be used to spoof an entire session with the server.

RECOMMENDED CHANGES TO THE KERBEROS PROTOCOL

- A challenge/response protocol should be offered as an optional alternative to time-based authentication.

- Use a standard message encoding, such as ASN.1, which includes identification of the message type within the encrypted data.

RECOMMENDED CHANGES TO THE KERBEROS PROTOCOL, Cont'd

- Alter the basic login protocol to allow for handheld authenticators, in which \( \{R\}K_c \), for a random \( R \), is used to encrypt the server’s reply to the user, in place of the key \( K_c \) obtained from the user password. This allows the login procedure to prompt the user with \( R \), who obtains \( \{R\}K_c \) from the handheld device and returns that value instead of the password itself.

RECOMMENDED CHANGES TO THE KERBEROS PROTOCOL, Cont’d

- Mechanisms such as random initial vectors (in place of confounders), block chaining and message authentication codes should be left to a separate encryption layer, whose information-hiding requirements are clearly explicated. Specific mechanisms based on DES should be validated and implemented.
The client/server protocol should be modified so that the multi-session key is used to negotiate a true session key, which is then used to protect the remainder of the session.

Support for special-purpose hardware should be added, such as the key-store. More importantly, future enhancements to the Kerberos protocol should be designed under the assumption that a host, particularly a multi-user host, may be using encryption and key-storage hardware.

To protect against trivial password-guessing attacks, the protocol should not distribute tickets for users (encrypted with the password-based key), and the initial exchange should authenticate the user to the Kerberos server.

Support for optional extensions should be included. In particular, an option to protect against password-guessing attacks via eavesdropping may be a desirable feature.