Lecture 9 - Authentication

CSE497b - Spring 2007
Introduction to Computer and Network Security
Professor Jaeger
www.cse.psu.edu/~tjaeger/cse497b-s07/
Implementing Authentication Protocols

• Authentication
  – verifying identity (prove possession of a secret)
  – mutual authentication
  – key distribution (secret for secure communication)

• Leverage constructions to achieve *authenticity, confidentiality, and integrity*
  – Signatures
  – HMAC

• Protocols
  – Needham-Schoeder
Kerberos

- History: from UNIX to Networks (late 80s)
  - Solves: password eavesdropping
  - Online authentication
    - Variant of Needham-Schroeder protocol
  - Easy application integration API
  - First *single sign-on system* (SSO)
  - Genesis: rsh, rcp
    - authentication via assertion

- Most widely used (non-web) centralized password system in existence (and lately only ..)
- Now: part of Windows 2K, XP network authentication
  - Windows authentication was a joke.
An aside …

- **Authentication**
  - Assessing identity of users
  - By using credentials …

- **Authorization**
  - Determining if users have the right to perform requested action (e.g., write a file, query a database, etc.)

- **Kerberos** authenticates users, but does not perform any authorization functions …
  - … beyond identify user as part of Realm
  - Typically done by application.

- **Q:** Do you use any “Kerberized” programs?  
  - How do you know?
The setup ...

• The players
  – Principal - person being authenticated
  – Service (verifier) - entity requiring authentication (e.g, AFS)
  – Key Distribution Center (KDC)
    • Trusted third party for key distribution
    • Each principal and service has a Kerberos password known to KDC, which is munged to make a password key, e.g., $k^A$
  – Ticket granting server
    • Server granting transient authentication

• The objectives
  – Authenticate Alice (Principal) to Bob (Service)
  – Negotiate a symmetric (secret) session key $k^{AB}$
The protocol

• A two-phase process
  – User authentication/obtain session key (and ticket granting ticket) key from Key Distribution Center
  – Authenticate Service/obtain session key for communication with service

• Setup
  – Every user and service get certified and assigns password
A Kerberos Ticket

• A Kerberos ticket is a token that …
  – Alice is the only one that can open it
  – Contains a session key for Alice/Bob ($K^{AB}$)
  – Contains inside it a token that can only be opened by Bob

• Bob’s Ticket contains
  – Alice’s identity
  – The session key ($K^{AB}$)

• Q: What if issuing service is not trusted?
The simplified Kerberos protocol

1) Ticket?

2) Ticket-Granting-Ticket

3) Bob?

4) Ticket (Bob)

5) Ticket (Bob)
Kerberos Ticket Granting Tickets

• Alice requests a Kerberos session
  – Enters her password

• Her workstation forwards a request for a TGT
  – In clear (w/o password)

• KDC generates a TGT
  – \{K_{AT} + TGT + details to prevent replay\}K_{A}
  – The TGT contains session state: Alice, session key, expiration time
  – All are encrypted with TGS key (KDC master key)

• Q: Why is TGT encrypted with Alice’s key?
Service Session

• Alice wants to establish a session with a service Bob
  – She uses the TGT for each session

• Alice sends
  – The identity of the service: Bob
  – The TGT
  – And an authenticator to prove that her workstation knows the current session key

• Authenticators
  – Encrypted timestamp of the current time: \{time\}K_{AT}

• Receives a service session key and a ticket for Bob
Cross-Realm Kerberos

• Extend philosophy to more servers
  – Obtain ticket from TGS for foreign Realm
  – Supply to TGS of foreign Realm
  – Rinse and repeat as necessary

  • “There is no problem so hard in computer science that it cannot be solved by another layer of indirection.”
    – David Wheeler, Cambridge University (circa 1950)
Kerberos Reality

• V4 was supposed to be replaced by V5
  – But wasn’t because interface was ugly, complicated, and encoding was infuriating

• Assumes trusted path between user and Kerberos

• Widely used in UNIX domains

• Robust and stable implementation

• **Problem**: trust ain’t transitive, so not so good for large collections of autonomous enterprises
Kerberos Security

• Key storage issues
  – KDC is the focal point of security
  – However, user passwords and session keys may be stolen on compromised clients
  – Password cracking was done on Windows Kerberos messages

• Timestamps are an issue (not nonces like NH)
  – Don’t have to track what nonces have been used
  – Authenticators use timestamps as challenge-responses
  – However, timestamps are accepted with range of minutes

• Some crypto attacks have been proposed

• Despite these, Kerberos broadly used
  – Not the lowest hanging fruit
Needham-Schroeder Public Key

• Did anyone build a public key version of Kerberos?
  – No

• Ill-fated existence
  – “Proven correct” in 1990
  – Flaw found in 1995
  – Led to work on protocol analysis tools

\[
\begin{align*}
X.1 & : A \rightarrow I & A + I + \{N_a, A\}_{K_I^+} \\
Y.1 & : I(A) \rightarrow B & A + B + \{N_a, A\}_{K_B^+} \\
Y.2 & : B \rightarrow I(A) & B + A + \{N_b, N_a\}_{K_A^+} \\
X.2 & : I \rightarrow A & I + A + \{N_b, N_a\}_{K_A^+} \\
X.3 & : A \rightarrow I & A + I + \{N_b\}_{K_I^+} \\
Y.3 & : I(A) \rightarrow B & A + B + \{N_b\}_{K_B^+}
\end{align*}
\]
Secure SHe ll

• Secure login, file transfer, X11, TCP/IP over Internet

• Replaces old insecure protocols for such things that used passwords in cleartext

• Uses strong cryptography for communication
  – RSA is used for key exchange and authentication
  – Symmetric algorithms for data security
Basic SSH Protocol

• (1) Client opens connection to server
• (2) Server sends public *host key*
  – Enables approval of new hosts
  – Rejects changed host keys
  – Notifies on expired host keys
• (3) Client generates random number as session key
  – Encrypts for the server using the host key
• (4) Server decrypts the session key
  – Confirms receipt (authenticating itself to the client)
• (5) Client can then authenticate using traditional means
  – E.g., Password
SSH Security

• Client encrypts session key in server’s host key
  – Q: Does this guarantee integrity?
  – Q: Can you prove that this is not susceptible to man-in-middle attacks?

• In SSH v2, communication is protected via HMAC-SHA1
  – You should be able to write these messages
SSH Services

• Value of SSH comes from the services that it runs...
  – Remote services
    • scp, sftp, ...
  – Support for connections
    • X11 forwarding, TCP forwarding, ...

• Over a secure channel...
  – Using strong crypto

• And it’s straightforward to setup the server and easy for clients
  – Has to deal with a modest number of error cases
SSH Vulnerabilities

• The communication is secure, so what to attack...

• Several problems: circa 2001-2002
  – Buffer Overflows (sshd runs as root)
    • Several of these
  – Integer overflows
  – Confuse the program (ssh-agent on client runs as root)
  – Also, attack the client side (run as client)
  – DoS attacks

• OpenSSH system has been rearchitected

• Q: We’ll talk about how to fix these problems later...
Take Away

• Systems for authentication have been constructed
  – Powerful, broadly used
  – Cryptography is generally above reproach
  – System challenges
    • Kerberos timestamps
    • Key storage
    • System security

• Communication is probably not the weakest link