CSE543 - Introduction to Computer and Network Security

Module: Authentication

Professor Trent Jaeger
What is Authentication?

• Short answer: establishes identity
  ‣ Answers the question: To whom am I speaking?
• Long answer: evaluates the authenticity of identity proving credentials
  ‣ Credential – is proof of identity
  ‣ Evaluation – process that assesses the correctness of the association between credential and claimed identity
  • for some purpose
  • under some policy (what constitutes a good cred.?)
Why authentication?

• Well, we live in a world of rights, permissions, and duties?
  ‣ Authentication establishes our identity so that we can obtain the set of rights
  ‣ E.g., we establish our identity with Tiffany’s by providing a valid credit card which gives us rights to purchase goods ~ physical authentication system

• Q: How does this relate to security?
Why authentication (cont.)?

• Same in online world, just different constraints
  ‣ Vendor/customer are not physically co-located, so we must find other ways of providing identity
    • e.g., by providing credit card number ~ electronic authentication system
  ‣ Risks (for customer and vendor) are different
    • Q: How so?

• Computer security is crucially dependent on the proper design, management, and application of authentication systems.
What is Identity?

• That which gives you access ... which is largely determined by context
  ‣ We all have lots of identities
  ‣ Pseudo-identities

• Really, determined by who is evaluating credential
  ‣ Driver’s License, Passport, SSN prove ...
  ‣ Credit cards prove ...
  ‣ Signature proves ...
  ‣ Password proves ...
  ‣ Voice proves ...

• Exercise: Give an example of bad mapping between identity and the purpose for which it was used.
Credentials

• ... are evidence used to prove identity
• Credentials can be
  ‣ Something I am
  ‣ Something I have
  ‣ Something I know
Something you know …

- Passport number, mothers maiden name, last 4 digits of your social security, credit card number
- Passwords and pass-phrases
  - Note: passwords are generally pretty weak
    - University of Michigan: 5% of passwords were goblue
    - Passwords used in more than one place
  - Not just because bad ones selected: If you can remember it, then a computer can guess it
    - Computers can often guess very quickly
    - Easy to mount offline attacks
    - Easy countermeasures for online attacks
“Salt”ing passwords

• Suppose you want to avoid an offline dictionary attack
  ‣ bad guy precomputing popular passwords and looking at the password file

• A salt is a random number added to the password to differentiate passwords when stored in `/etc/shadow`

\[
salt_1, h(salt_1, pw_1) \\
\]
\[
salt_i, h(salt_2, pw_2) \\
\]
\[
salt_i, h(salt_3, pw_3) \\
\]
\[
... \\
\]
\[
salt_n, h(salt_n, pw_n) \\
\]

• consequence: guesses each password independently
• The rule of seven plus or minus two.
  ‣ George Miller observed in 1956 that most humans can remember about 5-9 things more or less at once.
  ‣ Thus is a kind of maximal entropy that one can hold in your head.
  ‣ This limits the complexity of the passwords you can securely use, i.e., not write on a sheet of paper.
  ‣ A perfectly random 8-char password has less entropy than a 56-bit key.

• Implication?
A question?

• Is there going to come a day where all passwords are useless?
  ‣ Suppose I can remember 16 bytes of entropy (possible?)
  ‣ Won’t there come a day when all passwords are useless?
    • Moore’s law and its corollaries?
Answer: no

- Nope, you just need to make the process of checking passwords more expensive. For example, you can repeat the salted hash many times ...
  - Linear cost speedup?

\[ salt_i, h^{100}(salt_i, pw_i) \]
Something your have …

• Tokens (transponders, …)
  ‣ Speedpass, EZ-pass
  ‣ SecureID

• Smartcards
  ‣ Unpowered processors
  ‣ Small NV storage
  ‣ Tamper resistant

• Digital Certificates (used by Websites to authenticate themselves to customers)
  ‣ More on this later …
A (simplified) sample token device

• A one-time password system that essentially uses a hash chain as authenticators.
  ‣ For seed \( S \) and chain length \( l \), epoch length \( x \)
  ‣ Tamperproof token encodes \( S \) in firmware

\[
pw_i = h^{l-i}(S)
\]

• Device display shows password for epoch \( i \)
• Time synchronization allows authentication server to know what \( i \) is expected, and authenticate the user.

• **Note**: somebody can see your token display at some time but learn nothing useful for later periods.
Something your are …

• Biometrics measure some physical characteristic
  ‣ Fingerprint, face recognition, retina scanners, voice, signature, DNA
  ‣ Can be extremely accurate and fast
  ‣ Active biometrics authenticate
  ‣ Passive biometrics recognize

• Issues with biometrics?
  ‣ Revocation – lost fingerprint?
  ‣ “fuzzy” credential, e.g., your face changes based on mood ...
  ‣ Great for physical security, not feasible for on-line systems
Biometrics Example

• A fingerprint biometric device (of several)
  ‣ record the conductivity of the surface of your finger to build a “map” of the ridges
  ‣ scanned map converted into a graph by looking for landmarks, e.g., ridges, cores, ...
Fingerprint Biometrics (cont.)

• Graph is compared to database of authentic identities
• Graph is same, the person deemed “authentic”
  ‣ This is a variant of the graph isomorphism problem
  ‣ Problem: what does it mean to be the “same enough”
    • rotation
    • imperfect contact
    • finger damage

• *Fundamental Problem*: False accept vs. false reject rates?
Web Authentication

• Authentication is a bi-directional process
  ‣ Client
  ‣ Server
  ‣ Mutual authentication

• Several standard authentication tools
  ‣ Basic (client)
  ‣ Digest (client)
  ‣ Secure Socket Layer (server, mutual)
  ‣ Cookies (indirect, persistent)

• Q: Are cookies good credentials?
Basic Authentication

GET /protected/index.html HTTP/1.0

HTTP/1.0 401 Unauthorized
WWW-Authenticate: Basic realm="Private"

GET /protected/index.html HTTP/1.0
Authorization: Basic JA87JKAs3NbBDs

CLIENT

CLIENT
Setting up Basic auth in Apache

• File in directory to protect (.htaccess)

AuthType Basic
AuthName Patrick’s directories (User ID=mcdaniel)
AuthUserFile /usr/mcdaniel/www-etc/.htpw1
AuthGroupFile /dev/null
require valid-user

• In /usr/mcdaniel/www-etc/.htpw1
  mcdaniel:l7FwWEqjyzmNo
  generated using htpasswd program

• Can use different .htaccess files for different directories
Basic Authentication Problems

• Passwords easy to intercept
• Passwords easy to guess
  ‣ Just base-64 encoded
• Passwords easy to share
• No server authentication
  ‣ Easy to fool client into sending password to malicious server
• One intercepted password gives adversary access to many pages/documents
Digest Authentication

GET /protected/index.html HTTP/1.1

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
realm="Private" nonce="98bdc1f9f017.."

GET /protected/index.html HTTP/1.1
Authorization: Digest
username="lstein" realm="Private"
nonce="98bdc1f9f017.." response="5ccc069c4.."
Challenge/Response

• Challenge \textit{nonce} is a one time random string/value
  \[ nonce = H(IP\text{address} : \text{timestamp} : \text{server secret}) \]

• Response: challenge hashed with uname & password
  \[ response = H(H(\text{name} : \text{realm} : \text{password}) : \text{nonce} : H(\text{request})) \]

• Server-specific implementation options
  ‣ One-time nonces
  ‣ Time-stamped nonces
  ‣ Method authentication digests
Advantages of Digest over Basic

- Cleartext password never transmitted across network
- Cleartext password never stored on server
- Replay attacks difficult
- Intercepted response only valid for a single URL
- Shared disadvantages
  - Vulnerable to man-in-the-middle attacks
  - Document itself can be sniffed
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/Vista network authentication
  ‣ Old Windows authentication was a cruel 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 $ke$, 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

  1. User authentication/obtain session key (and ticket granting ticket) key from Key Distribution Center
  2. 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?
Phase 1 (obtaining a TGT)

- $T_{\text{exp}}$ - time of expiration
- $n$ - nonce (random, one-use value: e.g., timestamp)

\[
[A, TGS, T_{\text{exp}}, n] \\
E(k^A, [k^{A,TGS}, TGS, T_{\text{exp}}, n]), E(K^{TGS}, [A, k^{A,TGS}, T_{\text{exp}}])
\]
Phase 2 (authentication/key dist.)

\[
[B, \text{Time}_{\text{exp}}, n, E(k_A^{A,TGS}, [B, \text{Time}_{\text{exp}}, n])],
E(K_{TGS}^{TGS}, [A, k_A^{A,TGS}, \text{Time}_{\text{exp}}])
\]

\[
E(k_A^{A,TGS}, [k_A^{A,B}, B, \text{Time}_{\text{exp}}, n]),
E(k_B^{B}, [A, k_A^{A,B}, \text{Time}_{\text{exp}}]),
E(k_A^{A,B}, [A, \text{Time}_{\text{exp}}, n]),
E(k_B^{B}, [A, k_A^{A,B}, \text{Time}_{\text{exp}}])
\]
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