Risks of the Passport Single Signon Protocol

On-line access with passwords/PINs

- Credit card accounts
- Bank account
- Brokerage accounts – trading, balances, 401k
- Newspapers, NYT, wall street journal
- Yahoo portfolio, iwon.com
- Internal employee services at work
- Hotjobs.com, monster.com, computerjobs.com
- Amazon.com, barnesandnoble.com, orkut
Dilemma

- Use same password for all accounts?
  - merchants learns passwords to other sites
- Use different passwords for all accounts?
  - how do you remember them?
  - write them down?
    - what if you lose the sheet of paper?
    - what if someone else sees it?
- Use a “high security” password, a “medium security” password, and a “low security” password
  - low security is okay, but need unique high security ones

Possible solution – single signon

- Enter a password once
- Use fact that authenticated to obtain credentials
- Present credentials automatically
- User does not need to enter any more passwords

- Requires single administrative domain or cross domain trust
  - cannot log into some place that never heard of you from some place you know
Single signon example - kerberos

- Users authenticate to Kerberos
- Kerberos issues tickets with session keys for services
- Clients send ticket and authenticator to service
  - authenticator is proof of freshness and knowledge of session key
- Server verifies authenticator and communicates using session key
- User only types in one password for Kerberos

Single signon on the web

- Users often have to enter login & password
  - cumbersome
  - dangerous (JavaScript Trojan login window)
- Can use cookie mechanism to store credentials on the client
- No way to implement authenticator without client software, e.g. plugin
- Example is Microsoft passport server
- Useful tool, but not without risk
Single server example

- Server has a master symmetric key, MK
- Users logs in using basic authentication
- Server takes user pw and encrypts with MK: \( \{pw\}_{MK} \)
- \( \{pw\}_{MK} \) is stored in a cookie on client
- subsequent accesses use cookie instead of bothering user
- cookie expires after the session
  - as soon as user accesses another web site

Architecture for multiple servers

- A “trusted” credentials server
- A relationship between each end server and trusted server
  - share symmetric keys
  - usually some payment to trusted server
- Users are known to trusted server
  - have an account
  - share some authentication information
Single signon (cont.)

1. Request page
2. Auto redirect
3. Redirect
4. Request credentials
5. Login & password
6. Redirect w/tokens in header
7. Request page w/credentials
8. Set cookie

Compare to Kerberos

- No way to implement authenticator
- If cookies are stolen, can spoof user
  - attacks have been shown where cookies can be stolen
- Client never knows the session key
- Merchant cookies are not Kerberos tickets
  - merchant cookies encrypted for the same site
  - Users must enter password to obtain Kerberos tickets
- Passport server cookies are the keys to the castle
  - can automatically provide auth info to any participating site
Key management

- single key used to encrypt all cookies in Passport (according to white paper)
- better to have a master key generate unique key per client
  - $k_1 = 3DES(MK, \text{Client}_1)$
  - $k_2 = 3DES(MK, \text{Client}_2)$
  - etc.
- MS claims to have some key mgmt, and promised documentation, but never delivered

Bogus merchant threat

- Merchant poses as a passport server
- Pretends to redirect to passport but instead
  - passsport.com, passpart.com, passpor.com
  - no SSL, just bogus site, spoof DNS
- user enters credentials, and bogus merchant learns them
- Basic problem with SSL and the web
- also possible with active attack exploiting the redirects to bogus passport site
Active attack

Flaw discovered

- Netscape 4.7
- Option to only return cookies to originating web server
- When user signs out of passport
  - feedback that passport credentials are being removed
  - feedback that passport credentials are being removed
  - generic MS web page
- then, when typing in hotmail.com into browser, automatically logged in.
- Attack fixed the day we told them about it
User interface - signout

Challenges to single signon on the web

- Reliance of web on DNS
- Reliance of SSL on DNS and users
  - users do not verify certificates
  - illegitimate certificates easy to obtain (58 root keys in Netscape 4.7)
- Key management
  - need authenticator to avoid stolen cookie attacks
  - not likely to happen – just as difficult to protect keys as cookies
- User interface – make sure users understand what is going on.
Other risks of Passport

• Passport server presents single point of attack/failure
  – no information in white paper about replication, which carries its own risks (duplication of private keys)
• Requiring cookies can lead to privacy compromises
• attacks, such as famous Hotmail attack, could lead to free credentials
• cookiemonster type attacks could lead to denial of service, by deleting or replacing cookies

Conclusions

• Passport provides a solution to a very difficult problem – single signon
• Passport carries with it the same risks as an SSL service
  – bogus merchants
  – reliance on DNS
• Passport had a serious flaw in it that was fixed when we pointed it out to them
• Kerberos has advantages because it is not restricted to existing web technologies