Network Security
Web Security and SSL/TLS

Angelos Keromytis
Columbia University

Web security issues

- Authentication (basic, digest)
- Cookies
- Access control via network address
- Multiple layers
  - SHTTP
  - SSL (TLS)
  - IPsec

Vulnerabilities

- Revealing private information on server
  - Information about host
- Server logs
- Intercept of client information (passwords, credit card numbers)
- DoS
- Confusion
  - User interface exploits
- Program execution
- Javascript vulnerabilities
  - Cross-site scripting

cgi-bin problems

- cgi-bin, server-side includes
- Server starts privileged, switches to non-privileged mode
- Random/hand-crafted arguments to cgi-bin
  - Usually scripts, meta-characters
- Perl in "taint" mode
- SQL injection

HTTP access control - basic

- Client attempts GET/PUT...
- Server returns

```
HTTP/1.0 401 Unauthorized
WWW-Authenticate: Basic realm="Columbia CS Pages"
```

- Client tries again with

```
Authorization: Basic base64(user:password)
```

- Passwords in the clear
- Repeat for each access

HTTP access control - digest

- Again, client attempts GET/PUT...
- Server declines, provides:
  - Realm: displayed to user
  - Domain: URIs, remembered by client
  - Nonce: calculated by server, H(client-IP, timestamp, server secret)
    - Does not require server state
  - Opaque: returned unchanged by client
  - Algorithm: digest, checksum (MD5)

HTTP access control - digest (2)

- Client tries again, providing response:
  - Same nonce, opaque data
  - Response: H(H(A1), nonce, H(A2))
  - Digest: H(H(A1), nonce, method, data, info, H(body))
- info = H(URI, type, length, coding, modified, expires)
- A1 = (user, realm, password)
- A2 = (method, URI)
- Digest useful for POST/PUT operations
- Server only needs H(A1), not password itself
  - Stolen H(A1) good for realm only

HTTP access control - digest (3)

- On successful request, client is given next nonce, digest
  - Avoid 401 on next request
  - Protects digest of HTTP body
- Subject to man-in-the-middle by proxy
- Hash is sufficient to gain access (to one realm only)
  - Must have unique realms
- No server authentication

SSL overview

- Secure Socket Layer
  - SSL 3.0 has become TLS standard (RFC 2246) with small changes
- Provide secure channel (byte stream)
  - Any TCP-based protocol
  - https:// URIs, port 443
  - NNTP, SIP, SMTP...
- Optional server authentication with public key certificates
  - Common on commercial sites

SSL overview (cont.)

- Optional client authentication
- Hash: combined MD5 and SHA1
- Encryption optional (with session key)
    - Default algorithms: DES40, DES, RC2, RC4, 3DES

SSL cipher suites

- Diffie-Hellman key exchange
- RSA
- Fortezza

SSL basics

- Layered protocol
  - Application-layer fragmentation, blocks of max 16KB
  - Data compression
  - MIC is H(message, session key)
  - Encryption with client or server "write" key
  - Transmit over TCP
- Stateful
  - Handshake to setup keys, algorithms
- Different encryption/MAC keys in each direction

SSL messages

- Alert: notification of error
- ApplicationData: actual data
- Certificate: sender’s X.509 certificate/public key
- CertificateRequest: request that client sends certificate
- CertificateVerify: digital signature
- ChangeCipherSpec: start using agreed-upon algorithms

SSL messages (2)

- ClientHello: here’s what I want and can do (algorithms)
- ClientKeyExchange: client’s keys
- Finished: all done
- HelloRequest: server asks client to start negotiation
- ServerHello: server capabilities (algorithms)
- ServerHelloDone: server done
- ServerKeyExchange: server’s key

SSL handshake

- Client->Server: Supported ciphers, nonce
- Server->Client: chosen cipher, nonce, certificate(s)
- Client->Server: Encrypted pre-master key

- Compute keys

- Client->Server: MAC of previous messages
- Server->Client: MAC of previous messages

SSL handshake

- Server->Client: HelloRequest (*)
- C->S: ClientHello
- S->C: ServerHello, Certificate (*), ServerKeyExchange (*), CertificateRequest (*), ServerHelloDone
- C->S: Certificate (*), ClientKeyExchange, CertificateVerify (*), ChangeCipherSpec, Finished
- S->C: ChangeCipherSpec, Finished

- "Finished" messages are encrypted
- (*) optional payload

Session keys

- 48-byte pre-master key Sp generated by client
- Compute:
  - MD5(Sp, SHA1("A", Sp, Nc, Ns))
  - MD5(Sp, SHA1("BB", Sp, Nc, Ns))
  - MD5(Sp, SHA1("CCC", Sp, Nc, Ns))
- Concatenate to get master secret
- Session key
  - Do the above again (replace Sp with master key)
  - Cut out pieces for server/client MAC/encryption keys and IVs

Record protocol

- Used to transfer actual data
- (Type, Version, Length) header, followed by data
- MIC follows, and any padding (if encryption is used)
- At the end of data exchange, close_notify alert is sent

More advanced features

- Session resumption
  - Session vs. connection
- Ephemeral RSA
  - Create temporary key, sign with long-term key
  - Include in ServerKeyExchange message to client
  - Remnant from export-restriction days
- Re-handshake
  - Change ciphers, re-authenticate
  - Handshake protected by existing SSL session

More advanced features (2)

- Server-gated cryptography
  - Again, remnant from export-restriction days
  - Client can do full crypto if talking to properly authorized server
  - Special indication in server certificate
  - Hacked...

More advanced features (3)

- Diffie-Hellman
  - Perfect forward secrecy
  - Needed with non-encrypting PK algorithms (e.g., DSA)

  - Ephemeral DH keys
    - Sign with RSA/DSA key
    - Send with ServerKeyExchange
    - Client sends DH value in ClientKeyExchange
  - Long-term DH keys (embedded in certificate)

More advanced features (4)

- Kerberos support
  - ClientKeyExchange also contains ticket
- Fortezza
  - Hardware cryptographic accelerator with key escrow

SSL security

- Good randomness
  - Netscape used rand(getpid() + gettimeofday())...
- Protect server’s private key
- Check the certificate chain
  - Domain name embedded in certificate (hack!)
  - Revocation!
- Algorithm selection

Client authentication

- Username/password over SSL
- Client certificate authentication
    - Not common