HW1-HW2 Overview

COM-402: Information Security and Privacy
Authentication

Request

Response
Authentication - hw1 ex1

- Password check on client side (Javascript!)
  - Javascript code visible to client
Javascript code (1)
var enc = superencryption(username, mySecureOneTimePad) ;
if (enc != password) {
    alert("I didn't say it would be easy, Neo. I just said it would be the truth.");
    return;
}

function superencryption(msg, key) {
    if (key.length < msg.length) {
        var diff = msg.length - key.length;
        key += key.substring(0, diff);
    }

    var amsg = msg.split(" ").map(ascii);
    var akey = key.substring(0, msg.length).split(" ").map(ascii);
    return btoa(amsg.map(function(v, i) {
        return v ^ akey[i];
    }).map(toChar).join(" "));
}
Javascript code (3)

var enc = superencrytion(username, mySecureOneTimePad);
if (enc != password) {
    alert("I didn't say it would be easy, Neo. I just said it would be the truth.");
    return;
}

e.preventDefault();
var mySecureOneTimePad = "Never send a human to do a machine's job";
var username = $('username').val();

function superencrytion(msg, key) {
    if (key.length < msg.length) {
        var diff = msg.length - key.length;
        key += key.substring(0, diff);
    }

    var amsg = msg.split(" ").map(ascii);
    var akey = key.substring(0, msg.length).split(" ").map(ascii);
    return btoa(amsg.map(function(v, i) {
        return v ^ akey[i];
    }).map(toChar).join('"'));
}
Fixing hw1 ex1 in hw2 ex1

- Implement password check on server side
Password check on server side (1)

Interface for running python web applications
Password check on server side (2)

● **Use POST HTTP method**
  ○ Submits data to be processed to a specific resource

● **Here the resource is /hw2/ex1**

● **Steps**
  ○ Key at least as long as e-mail
  ○ Bitwise xor e-mail with key
  ○ Convert result of xor to bytes (characters)
  ○ Base64 encode the bytes
Cookies - Kirill
Cookies

- For storing info across user’s sessions (HTTP is stateless)
- Stored on client machines
- Used for
  - Authentication
  - Personalization
  - Tracking

Credit: Wikipedia. HTTP cookie
Cookies

● Usually include
  ○ Name
  ○ Domain and Path
  ○ Expiration date

● But may also include browsing activity, account information, state, etc.
Cookies - Storing state in browser

Dansie Shopping Cart (2006)

```
<FORM METHOD=POST
   ACTION="http://www.dansie.net/cgi-bin/scripts/cart.pl">

  Black Leather purse with leather straps
  <INPUT TYPE=HIDDEN NAME=name VALUE="Black leather purse">
  <INPUT TYPE=HIDDEN NAME=price VALUE="20.00">
  <INPUT TYPE=HIDDEN NAME=sh VALUE="1">
  <INPUT TYPE=HIDDEN NAME=img VALUE="p">
  <INPUT TYPE=HIDDEN NAME=custom1 VALUE="P with leather straps">

  <INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">

</FORM>
```

Change this to 2.00
Bargain shopping!
HW1 - ex2

Steps to solve:

● Decode from base64
● You obtain smth like:
  'hero@epfl.ch,1488477881,com402, hw1,ex1,user'

● Substitute ‘user’ by ‘administrator’
● Encode back into base64 and paste into your browser state
● Go to /hw1/ex2/list
● Hack & Spy
HW2 - ex3

- When having received a POST request at “/ex3/login”, check “user” and “pass” and prepare an appropriate string:
  - administrator,timestamp,com402,hw2,ex3,administrator or
  - name,timestamp,com402,hw2,ex3,user

- Compute HMAC of the prepared string with your password of hw1/ex1 encoded in utf-8 as a secret key $K$ (use python3 module `hmac`)

  $$\text{HMAC}(K, m) = H((K' \oplus \text{opad}) || H((K' \oplus \text{ipad}) || m))$$

- Send a response with a cookie <your_string, HMAC>.
- Expect a POST request to "/ex3/list" with the cookie. Upon reception, check whether HMAC is correct and return a corresponding status code.
Key Agreement Protocols

- Key agreement protocol - parties agree on a shared key in such a way that both parties influence the key
  - Example: Diffie-Hellman key exchange
    - Alice and Bob have g and p
    - Alice: $A = g^a \mod p$, send A to Bob
    - Bob: $B = g^b \mod p$, send B to Alice
    - Shared key $K = A^b \mod p = B^a \mod p = g^{ab} \mod p$
    - Problem - No authentication -> vulnerable to MiTM attacks

- Potential solutions:
  - Public-key crypto: digitally signed keys (Certificate authorities, TLS, HTTPS,...)
  - Password authenticated key exchange protocols (PAKE)
PAKE

- PAKE - two parties establish a shared key based on their knowledge of a password in such a way that MiTM attacker can’t participate in the method
- Two main purposes of PAKE:
  - Generate a cryptographically secure shared key from a low-entropy password
  - Prove the knowledge of a password without sending the actual password
SRP

- Secure remote password protocol (SRP)
  - Augmented PAKE - server doesn’t store password-equivalent data
- Client and server have an established shared password
  - Goal: User wants to prove to the server that it knows the password without sending it
SRP

- Server stores users passwords as: 
  \{username, pass\_verifier, salt\}
- pass\_verifier = \nu\text{ in the diagram}
SRP

- Authentication is usually initiated by the user

Client

---------
U = <username>

---------

Host

--> s = <salt from passwd file>
SRP

- After this exchange the user and the server should have the same secret session key $S$
- To finish authentication they need to prove to each other that they indeed have the same key

\[
\begin{align*}
a &= \text{random()} \\
A &= g^a \mod N \\
\quad\rightarrow \quad v &= \text{<stored password verifier>} \\
b &= \text{random()} \\
\quad\leftarrow B &= (v + g^b) \mod N \\
p &= \text{<raw password>} \\
x &= \text{SHA}(s \mid \text{SHA}(U \mid ":" \mid p)) \\
S &= (B - g^x)^{a + u \cdot x} \mod N \\
S &= (A \cdot v^u)^b \mod N
\end{align*}
\]
PKI - Ceyhun

- For PAKE you trust the server and authenticate yourself
  - But do you always know who you talk with?
- Sending password in plain-text is not secure
  - Hey man, can you hack my girlfriend's Facebook password?
  - Used to be easy, shouldn't be easy
- First step self-signed certificate
  - A website signs a certificate proving that it holds the key
  - MITM?
  - TOFU?
PKI Infrastructure
Is it secure?

- Better than nothing for sure
- Kind of works
- What if a CA’s signing key is compromised (DigiNotar)
- What if A CA is coerced to sign something for _ _ _
Decentralization