Secure Web Authentication with Mobile Phones Using Keyed Hash Authentication

Authors

Eric Gieseke, John McLaughlin

egieseke@connecterra.com, john_n_mclaughlin@yahoo.com

CSCI E 170 Computer Security and Usability
Final Project
Harvard University Extension
January 11, 2005

Introduction

This project extends the work of Min Wu, Simson Garfinkel, and Rob Miller on the topic of Secure Web Authentication with Mobile Phones {1}.

In summary, the original project explored the use of a mobile phone to authenticate a user logging in to a secure web site (e.g. Fidelity) from a non secure Web terminal (e.g. public kiosk). The user's mobile phone is employed as a security proxy so that the user is not required to type their password into the non secure terminal. A security proxy server mediates the login process between the non secure terminal, the mobile phone, and the target web site. The security proxy server is trusted and contains the user's password required by the target server.

In the original work, the security proxy server sent a SMS messages to the user's phone. The message contained a list of keywords. The user selected the keyword that matched the keyword displayed in the browser. A message was then sent back to the server containing the selected keyword. The security proxy server then proceeded to log in to the target server, at which point the user was authenticated on the target server, and could proceed working from the non secure terminal.

While the original project provided a successful proof of concept, there were 2 problems:

1. The SMS messages often take in excess of 1 minute to arrive, delaying the login process and negatively affecting the usability of the system.

2. It is fairly easy to spoof the security proxy server. This could be used to trick the user into entering confidential information, a credit card number for example.

Our project addresses these issues by replacing the use of SMS with HTTP communication between the mobile phone and the security proxy server. A Java based MIDP application installed on the mobile phone supports the interaction between the phone and the security proxy. The MIDP application also provides a graphical user interface for interaction with the user. This modification results in sub second response time and greatly improves the usability of the system.

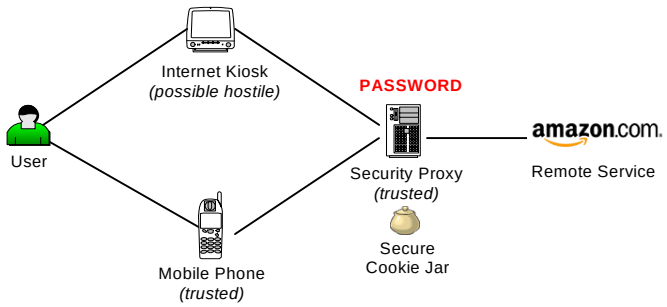
To prevent spoofing, a keyed hash (SHA1) exchange is employed to authenticate the security proxy server and the mobile client. The use of SSL was considered, but the keyed hash exchange was ultimately chosen for its simplicity and ease of configuration.

The Security Proxy Server mediates the interaction between the non secure terminal, mobile phone, and target server. It also supports the form based interaction required to log in at each target server. Since

there is no standard API supporting security proxies, each server requires special code to support the form based login.

The project goal is to use the mobile security proxy to log into at least 2 different target servers (e.g. Amazon, Yahoo).

System Architecture



System Components

The system is composed of 4 main components. The non secure web browser, the Security Proxy Server, the mobile phone hosting the authentication application, and the target web server. The end user interacts with the non secure web browser to initiate a connection with the security proxy server, and then authenticates using the authentication application installed on the mobile phone. After the user authenticates, the security proxy server automatically logs into the target server and redirects the users browser to the target server.

Web Interface

The non secure browser provides a host for web access for the user. The browser and the host machine are considered hostile. Threats include keyboard sniffing, and software designed to capture user inputs and information. The goal of the Mobile Security Proxy is to limit

Mobile Authentication Application

User's personal mobile phone hosting MIDP/Java based Mobile Security Application. Provides secure authentication mechanism for user. Communicates with Security Proxy Server via HTTP over the carrier network.

Security Proxy Server

Stores user names and passwords for target services. Supports interaction with Mobile Security Application. Provides secure proxy for http traffic between non secure Web Browser and target server(traps cookies, and performs http url rewrites)

Target Server

Web Site or service requiring user authentication to access services.

Mobile Authentication Client

The Mobile Authentication client resides on the user's personal phone. The application provides a user interface for interaction with the user and negotiates the authentication with the Security Proxy Server.

User Interface

The primary design considerations were usability and security. For usability, we focused on keeping the interface as simple as possible. This was achieved by not exposing the details of the authentication process. We also leveraged the authentication techniques previously used by Wu {1}. A list of phrases is displayed and the user is prompted to select the phrase matching the phrase displayed in the web browser. Selection of the correct phrase results in session authentication. In our testing, this technique proved to be simple and straightforward, and very usable. Another advantage of this approach is that it does not require any key typing, which can be quite tedious and error prone. Application response time to button clicks is sub 2 seconds, including actions that require communication with the server. These times were measured using the emulator. We expect faster speeds on the actual phone, since the emulator tends to be slower. Fast response time will further enhance the user experience.

Communication

Communication between the Mobile Authentication Client and the Security Proxy Server is through HTTP Get request over the carrier's data network, through a gateway connecting to the Internet, and then to the Security Proxy Server. Response time is consistently sub second. Stability of connection may be an issue in remote regions or inside of buildings.

Cryptography

The application requires cryptography functions to support keyed hash computation and data encryption. These cryptographic functions are not supported by the libraries packaged by MIDP. A lightweight cryptographic package from IAIK SIC was selected to support these functions:

Generation of a random client nounce
Computation of keyed hash using the SHA1 algorithm.
Conversion between hex and byte format.

The keys, hashes, and nounces were transferred over the wire in hex format to support the HTTP protocol.

Remote Authentication Process

The following screen captures document a successful authentication session.

Remote Termination Process

The following screen captures document a successful session termination.

Protocol

This following is a description of the interaction between the objects involved in the MIDP Mobile Security Proxy. The objects include:

User: end user who uses the mobile security proxy to login to a target server requiring user authentication.

Non Secure Web Browser: web browser from where the user initiates the connection to the target server.

Mobile Phone: java enabled mobile phone hosting MIDP/Java based Mobile Security Application.

Security Proxy Server: Stores user names and passwords for target services. Supports interaction with Mobile Security Application. Provides secure proxy for http traffic between non secure Web Browser and target server(traps cookies, and performs http url rewrites)

Target Service: Web Site or service requiring user authentication.

Assumptions:

1. Client and server authentication is supported using a keyed hash exchange. If either authentication fails, then the session authentication will be aborted.
2. Security proxy server will store the user name and password for each target service.
3. Each user will be assigned a unique client id or user name. This will be used to identify the user when initiating session between non secure web browser and security proxy server. This user name will also be used to identify the mobile security proxy aplication to the security proxy server. The client id is configurable.
4. A shared secret will be assigned the user. The shared secret will be stored on the mobile security proxy within the record management store. The stored value will be encrypted using a key known only by the mobile security application. The Security Proxy Server will maintain a shared key for each client id. The shared key is configurable and may be updated.
5. A session key will be generated by the server and passed to the client. The session key will be used to encrypt data between the client and the server after the client and server authenticity has been established.
6. Session approval will use a secret pass phrase. This phrase will be displayed to the user via the non secure web browser. A list of possible phrases including the secret pass phrase will be given to the user as a list to choose from on the mobile phone. The user will select the matching phrase on his phone to approve the session.
7. Session may be terminated either through the web browser or from the mobile phone.

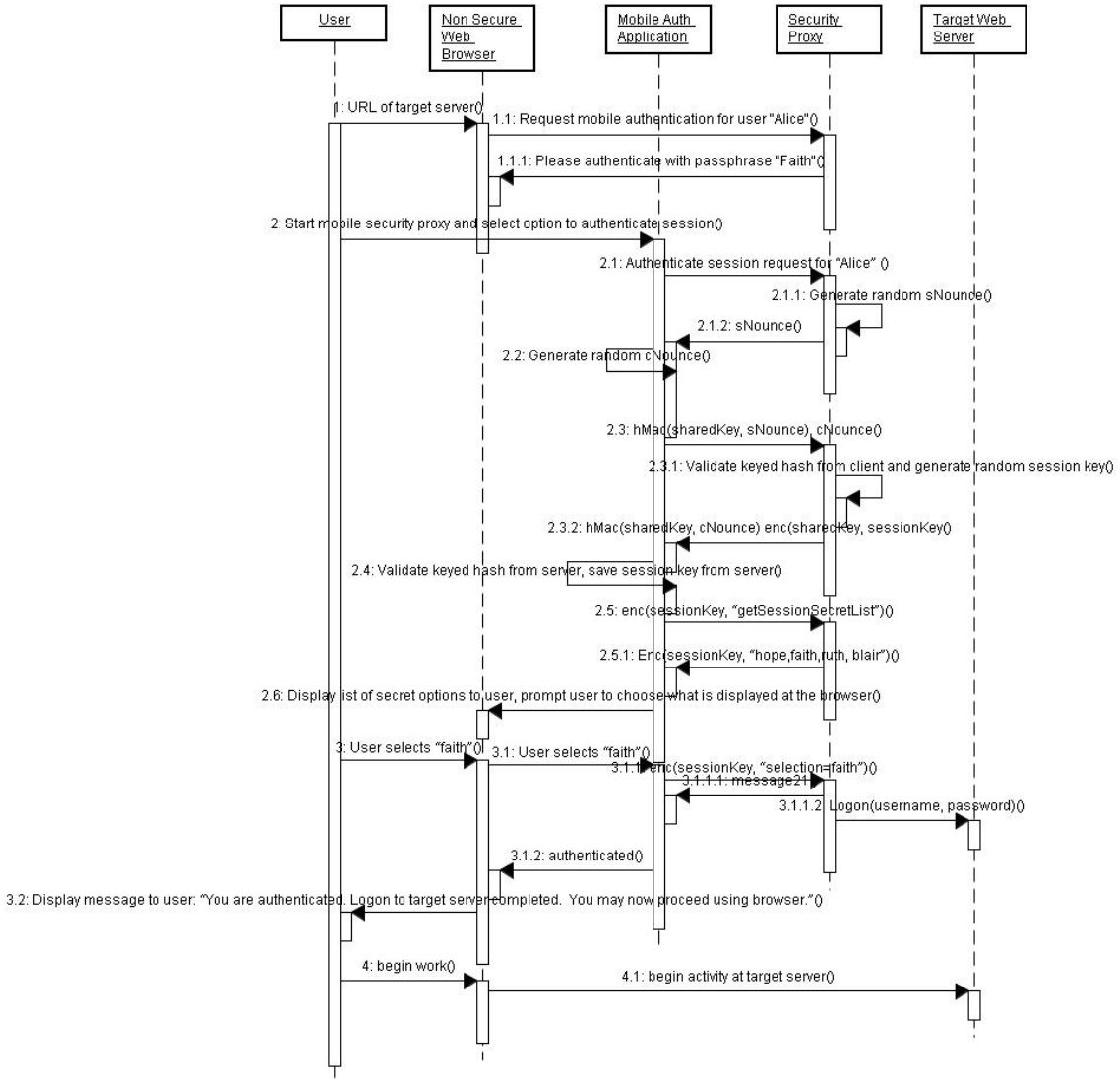


Figure 1 Sequence Diagram showing component interaction

Mobile Security Proxy Protocol

The following protocol documents the message syntax for interaction between the mobile authentication application and the Security Proxy Server. All message exchange use synchronous HTTP GET request. The server does not support HTTP session state. To allow the server to identify the originator, the session id is sent with each message following the startSession message (message 2s, 3s, 4s, and 5s).

1s) startSession=<clientid>
1r) OK,<sessionId>,<serverNouce>

2s) authClient=<sessionId>,<hmac(sharedKey,serverNounce)>,<clientNounce>
2r) OK,<hmac(sharedKey,clientNounce)>,<enc(sharedKey,sessionKey)>

3s) requestPassphrase=<sessionId>
3r) OK,enc(sessionKey,passPhraseList)

4s) selectedPhrase=<sessionId>,<enc(sessionKey,selectedPassPhrase)>

4r) OK,sessionAuthenticated

5s) killSession=<sessionId>
5r) OK,sessionTerminated

The first 4 request/response pairs are required to complete an user authentication. The killSession request (5s) is used to terminate an existing session.

Server

Security Proxy Server

The Security Proxy Server mediates the interaction between a non-secure terminal, a mobile phone, and a target server and is responsible for the user, session and target service data.

Interaction with the non-secure terminal

The user initiates an authentication session with the Security Proxy Server from a non-secure terminal by providing a username at the Server's web based interface. If the user provides a valid credential, the Server will generate and store a session id and a session word. The session word is returned to the web interface with instructions to complete the authentication with the user's mobile device. The Server periodically checks the status of the authentication and presents the links to the user's pre-configured services [i.e., Yahoo!, Amazon.com, etc.] upon successful completion. If the Server finds the user already has an active session, it will prevent them from starting a new one, however users have the option of terminating the session at this point and at the session phrase screen.

Interaction with mobile client

The user authenticates the session they began at the non-secure terminal by initiating a keyed hash exchange with the secure proxy from their mobile device. Upon receipt of the client identification, the server and mobile client exchange random nounces and return HMAC SHA-1 hashes. Nounces and hashes are compared to authenticate to each other. Upon authentication, the server returns session challenge consisting of a list of words, including the word selected at the non-secure terminal. If the mobile client returns the correct word, their session will be updated and they will be allowed to select a target service from the non-secure terminal. An incorrect selection will cause this session to fail and the user would need to authenticate again.

Interaction with the target service

The Security Proxy Server is responsible for redirecting the user to their target server and supports the form-based interaction required to automatically log in. A user's site information, including url, username and password are pre-configured by the user at a secure computer and are presented as selectable options upon successful authentication.

Data Management

The Secure Proxy Server manages user, session and target service data, utilizing a local mySQL database. Users would manage their site credentials in the user and target service tables, though a simple and secure interface, while the Server manages sessions and validations primarily from the session table.

System design and technology

The Secure Proxy Server is built on a LAMP stack [Linux, Apache, mySQL & PHP] and relies on built-in libraries to support the keyed hash exchanges and the session encryption. These tools were chosen for their appropriateness for building fast web-based applications and their substantial built-in feature set.

Future work

Several areas need to be developed to improve the usability, integrity and security of the Security Proxy Server.

Session timeouts should be applied to specific tasks, i.e., a very small timeout should be used during the hashed key exchange, while a longer one could be implemented for terminal interactions. Likewise, users should have appropriate time penalties when they kill an active session to prevent a rapid succession of authentication attempts.

Presently, the Secure Proxy will automatically log a user into a target server but it needs to be able to proxy all the http traffic between the non-secure terminal and the target to truly be secure. The Secure Proxy needs to be developed to perform http url rewrites and trap and store cookies. The proxy must isolate all potentially sensitive information from the non-secure terminal.

With the lack of a common login API, the site-specific authentication is potentailly the most fragile piece of our system. Each server requires specific code to support the form based authentication and while usernames and passwords have some degree of longevity, the same cannot be said of URLs and the web-based forms that use them. Sites whose users would benefit from using the Secure Proxy Server could be encouraged to standardize their login APIs and improve the integrity of their user's logins.

Web Interface Authentication

The following screen shots document the process of the login process from the web interface.

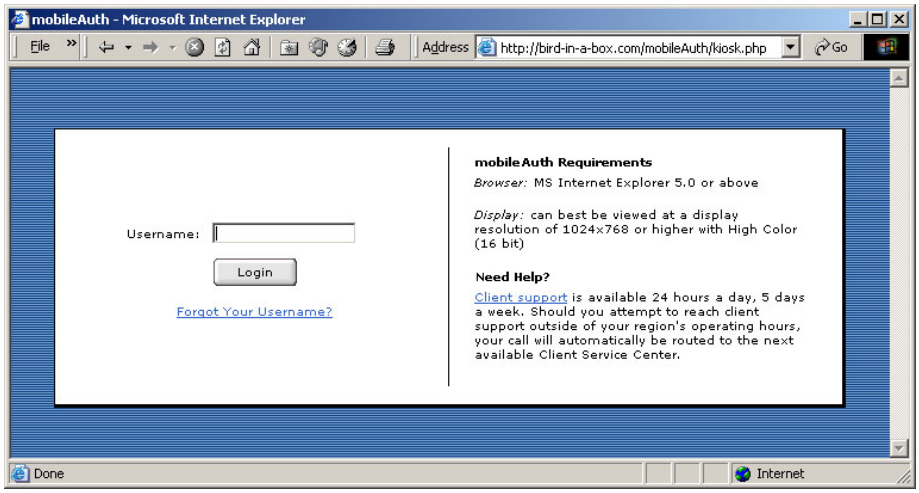


Figure 2 User navigates to Security Proxy Authorization

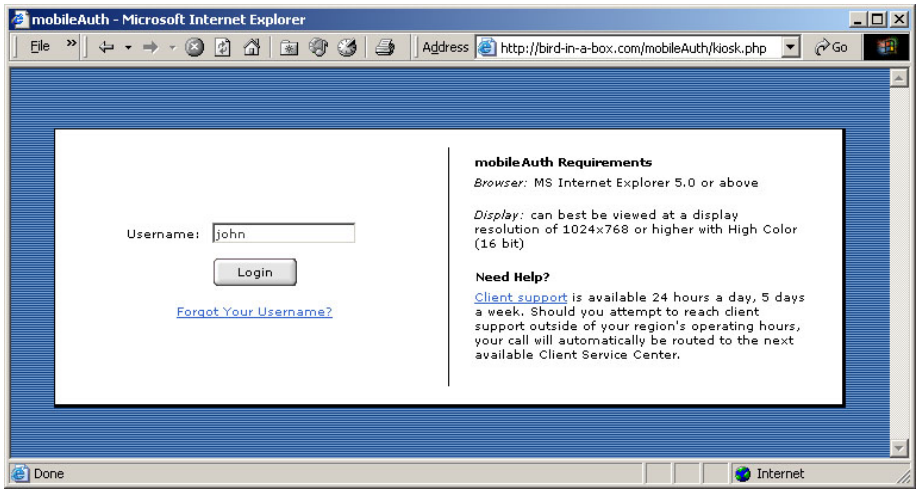


Figure 3 User types user id "John"

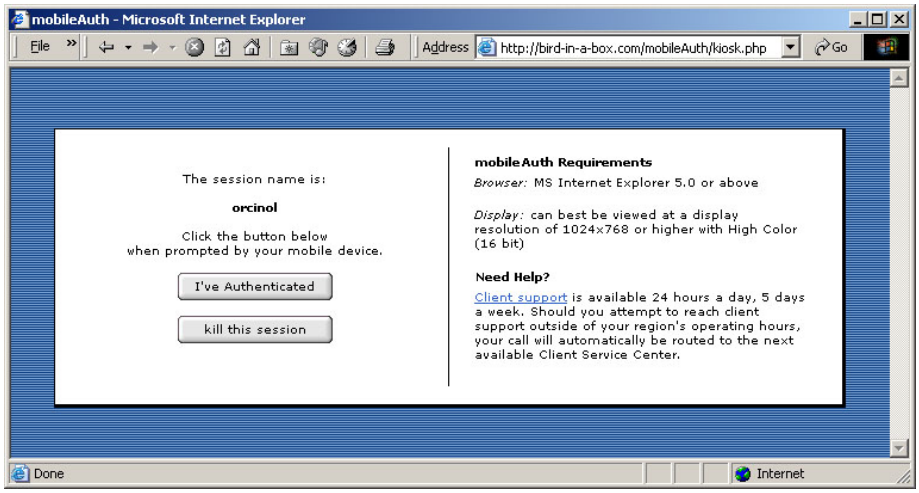


Figure 4 Pass phrase "orcinol" is displayed and user is prompted to authenticate using mobile phone.

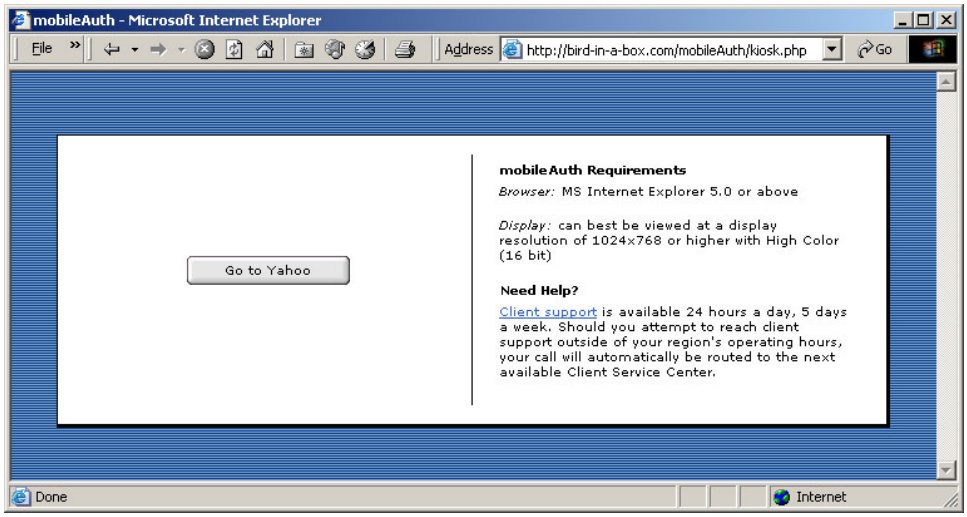


Figure 5 Successful authentication, user is prompted to go to Yahoo by clicking on "Go To Yahoo" button

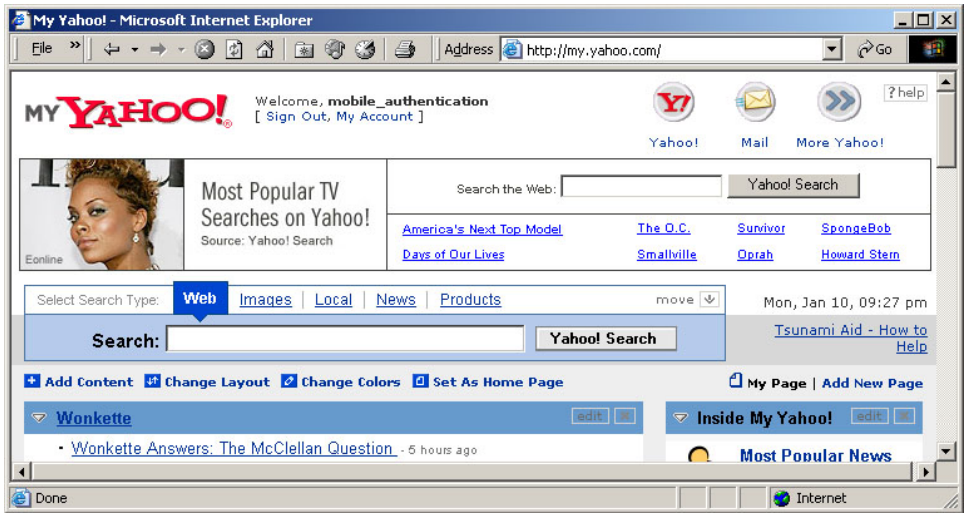
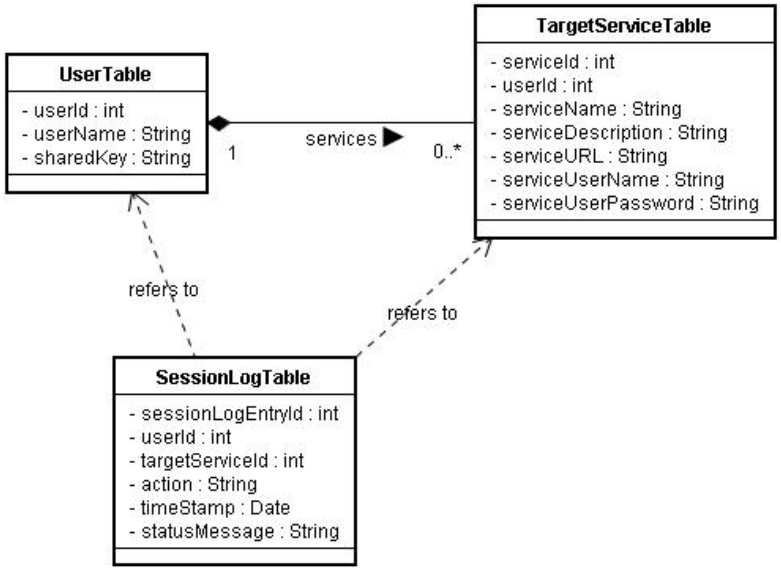


Figure 6 Yahoo page comes up, with user login already done.

Database Schema

The following diagram describes the database schema used to data for the security proxy server.



UserTable

This table stores a list of all users supported by the system.

TargetServiceTable

The TargetServiceTable stores service information. A user may have multiple target servers. Each entry contains the information required to find the service and login.

SessionLogTable

The session log table contains an event log of client activity. Events include browser interactions, and mobile authentication actions. This table journals user activity and may be used to detect suspicious behavior.

Data Security

The database contains highly sensitive data, including user names and passwords. Extreme care must be taken to insure that the data is secure and not compromised. In a production system, the database will require strong encryption to prevent casual browsing of information. The database must be maintained in a facility that provides both electronic and physical intrusion protection.

Summary

We were successful in goals of creating a working mobile authentication client based on a keyed has authentication using the (hMac Sha1). The client was implemented using the MIDP platform running on a cell phone (emulator). The server was implemented using PHP and mySQL. The resulting application had good usability characteristics, being fast and simple to use, without any key typing required by the user. We successfully demonstrated the ability to authenticate and automatically log in to the Yahoo web site. Do to time constraints, we did not support log in to Amazon.

Bibliography

1) Secure Web Authentication with Mobile Phones
Min Wu, Simson Garfinkel, Robert Miller
MIT Computer Science and Artificial Intelligence Lab, 32 Vassar Street, Cambridge MA 02139 {minwu, simsong, rcm}@csail.mit.edu
http://dimacs.rutgers.edu/Workshops/Tools/abstract-wu-garfinkel-miller.pdf

2) Trusted Paths for Browsers
Zishuang (Eileen) Ye, Sean Smith, {eileen,sws}@cs.dartmouth.edu, Department of Computer Science, Dartmouth College, June 12. 2002

3) Authentication for Humans
Audun Josang, Mary Anne Patton and Anthony Ho
Distributed Technology Center, Brisbase, Qld 4001, Autstriala, email: ajosang.patton,anthony@dstc.edu.au

4) SSL Web Proxy, A Secure and Inexpensive Remote Access Implementation
Prepared and Developed by David E. Culp
April 2, 2003

5) User Interaction Design for Secure Systems
Ka-Ping Yee
http://zesty.ca/sid/
ping@zesty.ca

6) Usability of Security: A Case Study
Alma Whitten and J. D. Tygar
December 18, 1998
CMU-CS-98-155

7) Secure Java MIDP Programming Using HTTPS with MIDP
by Qusay H. Mahmoud
June 2002
http://developers.sun.com/techtopics/mobility/midp/articles/https/

8) Database Nation, The Death of Privacy in the 21st Century, copyright 2000, 2001.

by Simson Garfinkel ISBN: 0-596-00105-3

9) IAIK JCE ME 3.02 - July 2004 Users Manual
http://jce.iaik.tugraz.at/

Appendix

Security Proxy Server Log File

The following log was collected by the Apache Web Server, and documents a successful authentication session.

```
24.60.64.65 - - [09/Jan/2005:22:28:22 -0500] "POST /mobileAuth/kiosk.php HTTP/1.1"
200 3291 "http://www.bird-in-a-box.com/mobileAuth/kiosk.php" "Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)"
24.60.64.65 - - [09/Jan/2005:22:28:38 -0500] "GET
/mobileAuth/mobileAuth4.php?startSession=eric HTTP/1.1" 200 96 "-" "-"
24.60.64.65 - - [09/Jan/2005:22:28:39 -0500] "GET
/mobileAuth/mobileAuth4.php?authClient=a141fe8edc2209c9ad69c60c6e88d9bb191bc3
da,d732bf885d32b2821d5ec92a8efdc8438cc4c244,1acf3e8bed355b3872d25d9d466ded
8dd1046860 HTTP/1.1" 200 96 "-" "-"
24.60.64.65 - - [09/Jan/2005:22:28:40 -0500] "GET
/mobileAuth/mobileAuth4.php?requestPassphraseList=a141fe8edc2209c9ad69c60c6e88
d9bb191bc3da HTTP/1.1" 200 47 "-" "-"
24.60.64.65 - - [09/Jan/2005:22:28:49 -0500] "GET
/mobileAuth/mobileAuth4.php?selectedPhrase=a141fe8edc2209c9ad69c60c6e88d9bb19
1bc3da,Keatsian HTTP/1.1" 200 28 "-" "-"
24.60.64.65 - - [09/Jan/2005:22:28:56 -0500] "POST /mobileAuth/kiosk.php HTTP/1.1"
302 5 "http://www.bird-in-a-box.com/mobileAuth/kiosk.php" "Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)"
24.60.64.65 - - [09/Jan/2005:22:28:56 -0500] "GET /mobileAuth/kioskConnect.php
HTTP/1.1" 200 2730 "http://www.bird-in-a-box.com/mobileAuth/kiosk.php" "Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)"
```

Database Session Log Table Entry

sessionID	clientID	startKiosk_ts	sWord	sMobile_ts	Nserver	authClient_ts	keySession	reqList_ts	rWord_ts	kill_ts	status_ts	status_message
a141fe8edc2209c9ad69c60c6e88d9bb191bc3da	eric	2005-01-09 22:28:22	Keatsian	2005-01-09 22:28:38	11a530b28e3f67326eb1523c786cf826b88f9f6a	2005-01-09 22:28:39	a39207f2d58728465ea7e7fb45e1bed15a4a06d4	2005-01-09 22:28:40	2005-01-09 22:28:49	*NULL*	2005-01-09 22:28:49	0

Note that the time stamps show an average of one second response times. This is using an emulator which is slower that the actual phone.