DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033—Computer System Engineering Handout 35—May 18, 1995

Notes on Design Project #2

These notes record various observations about the second design project, and were made by Prof. Bill Dally in the course of reading the project papers of his two recitation sections.

Solutions to design project two should address the following issues:

1. INTEGRATION WITH EXISTING SOFTWARE

1.1 The best solutions use standard Web browsers and servers with no modification by having a separate piece of software at either end that deals with the secure part of the protocol (and maintaining the shopping cart at one end).

A. At the browser end this could involve either (1) using a "proxy server" that sits between the browser and the server or (2) integrating the additional functionality into an "external viewer."

B. At the server end this is best accomplished by having the secure software at the client end carry out the purchase with a "payment switch." A Web server doesn't have the functionality required.

1.2 Other proposals involve modifying existing browsers and servers to integrate the functionality required for privacy, authentication, shopping carts, and executing purchases. This is clearly less desirable than (1.1) as it requires all users to use one particular browser, and requires that the modifications be re-integrated into each release of the browser.

2. SHOPPING CARTS

The shopping cart functionality can be placed on either the client side or server side.

2.1 Client Side: This does not tie up resources on the server and hence scales better.

Since the client machine is not trusted, a mechanism is required to guarantee that the shopping cart contents are valid. For example, one could require that item descriptors have an expiration date and be digitally signed by the merchant to guarantee the authenticity of pricing and availability. Alternatively the payment switch can validate items by querying the merchant for confirmation before processing the order. However, this method would allow merchants to repudiate an offer of goods -- e.g., bait and switch.

Another advantage of the client side is that if either 1.1A or 1.1B is used, the external client software can manage the shopping cart allowing merchants to use standard Web servers.

One approach proposed for integrating the shopping cart into an existing browser was to modify the "hotlist" feature of Mosaic.

2.2 Server Side: This requires that the server maintain the state of the shopping cart and have some means of associating users with their carts.

The state can either be maintained in a file or carried back and forth in the URLs and replies between the browser and server. If the state is carried along in the URLs and replies, there is no need for the name association.

To associate users with their carts the server first needs to identify the user making each request. Since requests might be coming through a proxy server or from behind a firewall, the IP address is *not* sufficient. The easiest way to do this is to rewrite the URLs so each user has a different URL and the user ID is passed back and forth in the URLs and responses.

3. AUTHENTICATION

Most solutions used Kerberos for authentication while a few used public key schemes. In either case the report needed to include enough details to show that authentication was provided. Simply saying, "Kerberos is used for authentication" isn't sufficient. In particular the scheme used must include:

3.1. A nonce to truly establish two-way authentication. Many of the proposals simply had the client get a ticket for the payment switch and present that ticket. Without a nonce (e.g, the timestamp of the authenticator plus one) that must be included in the encrypted reply, this scheme is vulnerable to a replay attack -- even with encrypted communication.

3.2. Challenging the user for credentials. Depending on the value of the purchase, the user should be prompted to reenter her credentials - this avoids the problem of someone using the user's workstation while she's away but before her Kerberos tickets time out.

3.3. The method of integrating the authentication protocol into the existing browser/server -- See (1).

3.4. Offers for goods and services on the net should be authenticated so bogus offers cannot be posted and so merchants cannot renege on valid offers. The easiest way to do this is to use digital signatures based on public keys (see 2.1). An offer is signed and is guaranteed by the merchant to be valid until an expiration time.

4. PRIVACY

Most solutions use encrypted communication for the purchase. The real issue here is how this is integrated with existing software (see (1)) and how the keys are managed. For key management, proposals either used Kerberos session keys or certified public keys (e.g., as in Netscape's Secure Sockets Layer protocol [http://www.netscape.com/newsref/std/SSL.html]).

Some solutions also avoided storing credit card information on-line by having the user reenter the data for each purchase. There is a trade-off here between ease of use and security. It would be sufficient to store the card data in encrypted form on-line and have the user enter a 4 digit PIN. In either case the data should be entered in a "protected" field -- like a password -- so an eavesdropper cannot read the secret off the screen.

5. PAYMENT SWITCH

A piece of server software is required to mediate the purchase. This software queries merchant computers to reserve goods, queries financial computers to reserve money, and then commits the transaction and sends confirmation messages to the merchant and financial computers.

In general, it's simpler to have the entire back-end settlement process be separate from the merchant's Web servers.

6. PROTOCOL

Protocols varied but most that worked followed the following scenario.

1. An unmodified Web browser sends GET requests to an unmodified Web server to browse the store.

2. From time to time the user clicks on a hot-link and the returned page causes an item to be placed in the shopping cart. This is done either by a proxy server intercepting the reply as it goes by or by spawning an external viewer in response to the reply. Each item in the cart consists of a product identifier, an offering price, and an expiration time signed by the merchant.

3. Eventually the user clicks on a purchase button. The reply causes client software (either the proxy server or the external viewer) to initiate a secure authenticated connection with the purchase switch (not the Web server). Any additional information (PIN, credit card number, shipping address) is entered in a dialog box brought up by the client software.

4. The purchase switch completes the purchase using a two-phase commit protocol to make sure that the entire purchase is atomic with respect to other purchases and crashes of the purchase switch, merchant computers, and financial computers. A reliable log is kept by the purchase switch.

5. The purchase switch returns a receipt (digitally signed) to the customer and adds the customer to ACLs for any on-line goods purchased.

7. OTHER ISSUES

Some of the better proposals discussed other issues like:

1. How disputes are resolved for unauthorized purchases or purchases that “never arrive.” Where are the liabilities? Where is the burden of proof?

2. How are on-line goods protected from copying?

3. What are the threats to safe operation of the system?