WebALPS: A Survey of E-Commerce Privacy and Security Applications

S.W. Smith
Department of Computer Science/ Institute for Security Technology Studies
Dartmouth College
Hanover, New Hampshire 03755 USA
sws@cs.dartmouth.edu

Web-based commerce is rife with scenarios where a party needs to trust properties of computation and data storage occurring at a remote machine, operated by a different party with different interests. In our WebALPS project, we have used off-the-shelf hardware and open source software to build *trusted co-servers* co-resident with Web servers, and bring the secure SSL channel all the way into these trusted co-servers. In this paper, we survey how this tool can be used to systematically address privacy and security issues in e-commerce.

1. INTRODUCTION

Moving commercial activity into distributed electronic environments creates a fundamental trust problem: how does a client know what happens at remote sites? The current "secure" Web infrastructure addresses some issues of server authentication and channel protection, but does not address this core trust problem. This paper surveys how our WebALPS project may help address these issues. (We refer the reader to other reports [Jiang 2001; Jiang et al. 2001; Knop 2001] for architectural and implementation details.)

1.1 Trust Issues

By freeing merchants from the overhead of bricks-and-mortar and of the barriers of local neighborhoods and legacy media, the Web transforms the commerce playing field. A start-up bookstore can have the same on-line presence and reach the same potential customers as an established on-line merchant such as Amazon—and can

This work was supported in part by IBM Research, by the U.S. Department of Justice, contract 2000-DT-CX-K001, and by Internet2/AT&T. However, the views and conclusions do not necessarily represent those of the sponsors.
Part of this work was performed while the author was a Research Staff Member at IBM Watson.

reach more than a bricks-and-mortar merchant at the local mall.

Unfortunately, the start-up merchant has a disadvantage when it comes to trust. Commerce typically requires that customers trust the merchant with sensitive data—credit card numbers, home addresses, preferences, health information. Customers are more likely to trust merchants with pre-established reputations. The risk of fraud at least partially negates the wider range of this new marketplace.

The trust relations in commerce go beyond this initial characterization, with more parties and more complex relationships. Each trust issue raises more security and privacy risks.

1.2 The Current Infrastructure

The currently deployed infrastructure for "secure" Web-based commerce consists of *Secure Sockets Layer (SSL)* [Freier et al. 1996] with server-side authentication. One of the standard browser trust roots certifies that a merchant's server is the unique possessor of a private key matching a specific public key. When a customer Alice wishes to engage in a sensitive interaction with a merchant Bob, she opens an SSL session: Alice's browser and Bob's server use Bob's key pair to establish symmetric keys that protect the channel between them. If Alice trusts her browser to accurately indicate that an SSL session exists with Bob, then she can trust that her information is protected from adversaries along the channel.

This infrastructure is insufficient for security. To start with, it is not clear that Alice should trust her browser to indicate an SSL session exists with Bob—in related work [Yuan et al. 2001], we have demonstrated how, for common client platforms, a malicious server can spoof all of the SSL clues. But furthermore, this infrastructure does not protect what happens at Bob's machine, and gives no grounds for *non-repudiation* of any transaction activity.

1.3 This Paper

In this paper, Section 2 quickly reviews our WebALPS project to remedy these infrastructure problems. Section 3 surveys applications of this tool for privacy and security issues in e-commerce. Section 4 concludes by discussing the path to making these potential applications real.

2. WEBALPS

Addressing server-side flaws in the current infrastructure requires providing a server-side sanctuary for computation and data storage safe from attack by the server operator. Making a solution real, not just hypothetical, also requires a way for real server operators to set up such sanctuaries, and for real clients (as well as all other stakeholders) to interact with and trust this sanctuary.

Our WebALPS project [Jiang 2001; Jiang et al. 2001] does this. We started with a commercial off-the-shelf secure coprocessor [IBM 2000] that permits users to install third-party applications in the hostile field [Smith and Weingart 1999], that provides cryptographic mechanisms for remote parties to verify they are interacting with these applications [Smith 2001], and that is validated against the highest level of assurance for commercial devices [NIST 1994; Smith et al. 1999]. We then integrated this coprocessor with Apache and modSSL, so that a server's SSL key pair resides in the coprocessor, and the client's SSL session goes all the way into

coprocessor-protected code.

In the current infrastructure, clients trust the the CA to certify the identity of a server. Our approach extends this by trusting a CA to also certify the identity of the co-server application—a judgment the underlying coprocessor technology enables. The client can then trust that this application will carry out its advertised activity without risk of subversion even by a malicious server operator.

A functioning prototype [Jiang 2001] and demonstration application [Jiang et al. 2001] exist; we are currently preparing this code for open-source distribution.

3. PROBLEMS AND SOLUTIONS

We now survey some areas where WebALPS may provide privacy and security for e-commerce. (Validating this potential is an area of ongoing work.)

3.1 Credit Card Transaction Security

Problem. The current Web infrastructure provides secure transmission of a client's information to the server—but what happens there is anyone's guess.

For example, consider the credit-card information and transaction amount a client sends when he wishes to purchase something. An adversary who compromises the server (or a malicious server operator) can use this data to carry out lots of mischief:

—He can increase the amount of the transaction.

—He can retain the amount but repeat the transaction many times.

—He can use the credit card information to forge additional transactions.

This situation may significantly reduce the potential market for new e-merchants without a pre-established reputation. ("Ribo's Books has a cheaper price than well-known Amazon, but how do I know that unknown Ribo won't steal or accidentally divulge my credit card info?")

Solution. To solve this problem the WebALPS guardian at the server can trap the credit card and transaction information, and then inject it directly into the acquirer's system. The CCN data never appears in plaintext at the server site; the server operator or a penetrator has no opportunity to inflate the transaction amount; and (unlike SET) the client need not change the way she operates.

We are currently implementing this approach in a scenario that does not require changing how acquirers accept charge information. Many entities (such as theater groups and athletics) at Dartmouth wish to sell tickets online; our computing services group is happy to set up a server, but would rather not have the liability of exposure to customer data. Consequently, a WebALPS co-server can capture the client information over an SSL channel, then re-encrypt it with the public key of the actual merchant party.

3.2 Nonrepudiation of Client Authentication

Problem. The current Web infrastructure prevents a server from being able to prove anything to a third party about the *identity* of an alleged client.

Without a public-key infrastructure for citizens, clients are forced to use human-usable authenticators, such as userids and passwords. However, in the current

infrastructure, these authenticators are exposed to the server of unknown integrity. As a consequence of this exposure, an adversary who compromises the server (or a malicious server operator) can impersonate this user at that site, and at any other site where the user has used these authenticators. This exposure also prevents legitimate server operators from being able to argue that it really was a particular client who opened a particular a session.

Solution. To solve this trust problem, the WebALPS guardian at the server can trap the password, authenticate the client, then issue a signed receipt for the server that client properly authenticated for that session. (Our demonstration application [Jiang et al. 2001] already does this first step of this process, using the legacy username-password system in place on campus.)

The client can trust that the password safely remains inside the coprocessor, even if the server operator might be motivated to retrieve it. The server operator, additionally, has a signed statement from a mutually trusted third party (the WebALPS guardian) that the alleged client really did authenticate correctly.

3.3 Nonrepudiation of Client Activity

Problem. The current Web infrastructure prevents a server from being able to prove anything to a third party about the *activity* of an alleged client.

For example, how can an insurance company taking an application from Alice over the Web turn around and prove that Alice really answered that question that way?

Solution. To solve this trust problem, the WebALPS guardian of Section 3.2 above can also issue a signed receipt for the entire transaction. "Alice not only authenticated correctly, but she issued a request of type X with parameters Y."

3.4 Nonrepudiation of Server Activity

Problem. The current Web infrastructure prevents a server from being able to prove anything to a third party about the activity of that server in an interaction.

For example, consider trying to prove something about the *questions* that generated the answers a client provided.

—Case law already exists that permits, in paper interactions, a client to alter a waiver before signing it—and if the service provider accepts the form without noticing the alteration, he is bound by it.

—Colleagues of the author have reported difficulty in U.S. Government security clearance processes, because an answer to a question five years ago was not consistent with the current revision of that question.

Do the responses to Web forms typically include the questions (or a hash of them)? If the clients are not even signing responses, how can we expect them to sign questions too?

Solution. To solve this trust problem, the WebALPS guardian of Section 3.3 above can include, in its signed receipt for the transaction, the prompts and responses the server provided.

3.5 Taxes on E-Commerce Activity

Problem. The current Web infrastructure provides no acceptable means to balance

the legitimate interests of a third party to accurately learn certain information about individual or collective Web interactions, with the privacy interests of the other participants.

For example, consider the problem of a government tax collection service trying to learn how much sales tax an e-merchant owes them for last month.

—Reporting *all* transactions to the government would be unacceptable to the merchant and customer for privacy concerns.

—Reporting only a total amount owed would be unacceptable to the government, since the figure would be unverifiable, and the merchant reporting this unverifiable figure would be motivated to understate it.

Solution. To solve this trust problem, the WebALPS guardian at the server can monitor the total tax owed by that merchant for the transactions that went through it (e.g., because of some other co-server application there), and report that authenticated total back to the government revenue agency. The agency can trust that the reported amount is correct; the merchant and customers can trust that the agency learns only what it is supposed to—and, in particular, is *not* learning details of transactions or identities of customers.

This problem is fairly complex; to be effective, this solution requires that something else exist to motivate customers to purchase through this channel.

3.6 Re-Selling of Intellectual Property

Problem. The current Web infrastructure provides no acceptable means for a third party who participates in an interaction indirectly, by licensing proprietary information to the server, to protect their legitimate interests.

For example, a publisher who owns a large copyrighted image database might wish to make this available to a university library—but might worry that compromise of the university server will compromise the database.

Solution. To solve this trust problem, the WebALPS guardian at the server would be set up to receive a session key and licensing rules from the owner of the intellectual property. The owner would provide the intellectual property in ciphertext to the server; the guardian would decrypt the particular items being used, and ensure that whatever licensing/royalty/watermarking requirements were being enforced.

3.7 Privacy of Sensitive Web Activity

Problem. The current Web infrastructure provides no means for a server operator to plausibly deny that they (or an adversary who has compromised their machine) is not monitoring all client interactions for nefarious purposes.

We present some examples from several domains:

—*Commerce.* Suppose a corporation provides a patent server as a public service. How can its competitors know for sure that the server operator is not data-mining their queries, in order to learn details of their proprietary research and development efforts?

—*Defense.* A unit preparing a special operation may request many maps for the particular region of interest. What prevents the operator of the map server from

observing this sequence of requests—all from the same unit and focused on the same region—and drawing a conclusion about the existence and target of the planned operation?

—*Social.* Consider people who wish to obtain sensitive literature—about health topics, for example, or about currently unfashionable politics. What prevents the server operator from learning of their activity, and acting in a manner (such as informing employers or health insurers) in way that would unjustly compromise their privacy?

Solution. To solve this trust problem, the WebALPS guardian would be set up with the appropriate *private information retrieval* software (e.g., [Smith and Safford 2001]) that makes use of encrypted storage in the server's file system. The client, through the SSL-protected channel, makes her request to the WebALPS guardian, which then retrieves the record, re-encrypts it, and returns it to the client. If we assume that cryptography works, the server operator learns nothing except the fact that a query has been made.

3.8 Correctness of Web Activity

Problem. The current Web infrastructure provides no means for a server operator to establish that they (or an adversary who has compromised their machine) has not otherwise altered or corrupted important correctness properties of the service.

For example, suppose an auction server provides a bulletin board service where customers can post "timestamped, anonymous, confidential" comments about participants and interactions. How can customers know that the anonymous posts came from *bona fide* customers, and that the timestamps are correct?

Solution. To solve this trust problem, we move the computation critical to the appropriate correctness properties from the server into the WebALPS guardian—whose application program would need to advertise that it was performing these computations.

Of course, this solution only establishes that the trusted guardian witnesses that the alleged *bona fide* customers authenticated properly. Subverting the authentication system subverts this approach (however, Section 3.2 may help).

3.9 Enforcement of Logo Licenses

Problem. The current Web infrastructure provides no effective means for a party to ensure that logos or endorsements appear only on the appropriate server pages.

For example, Dartmouth ISTS could establish an "ISTS-inspected" logo to endorse servers who have withstood penetration testing by ISTS specialists. However, any client who visits these pages can capture the logo, and put it on any page, whether or not that site has withstood the testing.

Solution. To solve this trust problem, the WebALPS guardian would be set up to provide the logo information, when appropriate. Logos that do not appear in the portion of the browser window from an authenticated guardian-to-client channel are not legitimate.

3.10 Safety of Downloadable Content

Problem. The current Web infrastructure provides no means for the client to ensure

that executable content downloaded from a server is indeed safe.

With the current state of consumer platforms, safety depends on the client themselves actually running the latest anti-virus software. Since most consumers do not do this, this leaves them at risk. Moving the virus-checking computation (and the concomitant problem of maintaining the latest updates of virus signatures) to the server is more efficient—but how can clients know the server really carried this out?

Solution. To solve this trust problem, the WebALPS guardian could run the anti-virus software with the latest signatures:

—either dynamically, as the guardian was feeding data back to the client;

—or offline (but then, when the guardian was feeding data back to the client, it would verify that it had indeed scanned this data earlier).

Clients could then trust that content downloaded via this SSL-authenticated channel from the trusted co-server has been scanned.

3.11 Authenticity of Downloadable Content

Problem. The current web infrastructure provides no easy means for the client to authenticate the origin of downloadable content.

Posters of content can provide digital signatures, but then the client needs to explicitly obtain and verify the trust chain on each item. Moving this verification computation (and the concomitant problem of maintaining the latest certificate revocation lists) to the server is more efficient—but how can clients know the server really carried this out?

Solution. To solve this trust problem, the WebALPS guardian could itself verify the signatures of the posted content (using all the latest certificate revocation lists, etc.), and then include in the SSL-encrypted channel an assertion that this content had been verified, and the identity of its poster. The client Alice could then trust that content she downloaded via this SSL-encrypted channel from the trusted co-server did indeed originate with the alleged poster. (We can even save bandwidth here, since the client only need download the identity of the poster, not his public key, signature, and appropriate certificates.)

3.12 Integrity of Server Machine

Problem. The current Web infrastructure provides no means for the client to verify the integrity and site security of server machines.

For example, some servers may run on machines whose administrators who run hardened operating systems and/or engage in other good security practices, such as regular runs of a network security analyzer, or enhanced OS boot via secure hardware (e.g., [Arbaugh et al. 1997; Yee 1994]).

But any site can *claim* to do this. How can a client know?

Solution. To solve this trust problem, the WebALPS guardian can witness that the appropriate computational security tool (such as a network security analyzer or a particular hardware-directed secure boot technique) was applied to the host—perhaps because this tool was applied from the co-server itself, or from a companion trusted machine. The client Alice can trust such assertions it receives through the SSL-authenticated communication channel from the co-server to the client.

4. CONCLUSIONS

By providing a practical way to insert a trusted third party into Web interactions, WebALPS provides a way to systematically address privacy and security issues in Web-based e-commerce. The next step is using our prototype to build practical implementations of these solutions. This work requires designing coprocessor-protected code that implements the required functionality, modifying the client-merchant session to invoke this code in a non-spoofable way. Widespread deployment requires setting up an SSL CA to certify such enhanced servers, educating clients to recognize such certificates. and releasing the applications as officially signed codeloads for the coprocessor platform.

Our group's current research includes these activities; however, we would be happy to work with additional colleagues.

ACKNOWLEDGMENTS

The author wishes to thank the helpful discussions and suggestions of many colleagues including R. Brentrup, D. Chess, J. Dyer, J. Hill, N. Itoi, S. Jiang, E. Knop, J. Kravitz, K. Minami, E. Palmer, and R. Perez.

REFERENCES

Arbaugh, W. A., Farber, D., and Smith, J. M. 1997. A secure and reliable bootstrap architecture. In *Security and Privacy* (1997). IEEE Computer Society.

Freier, A. O., Karlton, P., and Kocher, P. C. 1996. The ssl protocol version 3.0. http://home.netscape.com/eng/ssl3/draft302.txt.

IBM. 2000. Ibm 4758 models 2 and 23 pci cryptographic coprocessor. Product Brochure G221-9091-02. http://www.ibm.com/security/cryptocards/.

Jiang, S. 2001. Webalps implementation and performance analysis: Using trusted co-servers to enhance privacy and security of web interactions. Master's thesis, Department of Computer Science, Dartmouth College. (Available as Technical Report TR2001-399).

Jiang, S., Smith, S., and Minami, K. 2001. Securing web servers against insider attack. In *Annual Computer Security Applications Conference* (December 2001). ACSA/ACM. (A preliminary version is availabe as Technical Report TR2001-410, Dartmouth College.).

Knop, E. 2001. Secure public-key services for web-based mail. Senior thesis, Department of Computer Science, Dartmouth College.

NIST. 1994. Security requirements for cryptographic modules. Federal Information Processing Standards Publication 140-1.

Smith, S. 2001. Outbound authentication for programmable secure coprocessors. Technical Report TR2001-401 (March), Department of Computer Science, Dartmouth College. http://www.cs.dartmouth.edu/~pkilab/oatr.pdf.

Smith, S., Perez, R., Weingart, S., and Austel, V. 1999. Validating a high-performance, programmable secure coprocessor. In *22nd National Information Systems Security Conference* (October 1999). National Institute of Standards and Technology.

Smith, S. and Safford, D. 2001. Practical server privacy using secure coprocessors. *IBM Systems Journal 40*, 3. (Special Issue on End-to-End Security).

Smith, S. and Weingart, S. 1999. Building a high-performance, programmable secure coprocessor. *Computer Networks 31*, 831–860. Special Issue on Computer Network Security.

Yee, B. 1994. *Using Secure Coprocessors*. Ph. D. thesis, School of Computer Science, Carnegie Mellon University. (Available as Technical Report CMU-CS-94-149.).

Yuan, Y., Ye, E., and Smith, S. 2001. Web spoofing 2001. Technical Report TR2001-409 (July), Department of Computer Science, Dartmouth College. http://www.cs.dartmouth.edu/~pkilab/demos/spoofing/.