Appeared in *IEEE Computer Networks*, (July, 2000)

Risks of the Passport Single Signon Protocol

David P. Kormann and Aviel D. Rubin
AT&T Labs - Research
180 Park Avenue
Florham Park, NJ 07932
{davek,rubin}@research.att.com

Abstract

Passport is a protocol that enables users to sign onto many different merchants' web pages by authenticating themselves only once to a common server. This is important because users tend to pick poor (guessable) user names and passwords and to repeat them at different sites. Passport is notable as it is being very widely deployed by Microsoft. At the time of this writing, Passport boasts 40 million consumers and more than 400 authentications per second on average. We examine the Passport single signon protocol, and identify several risks and attacks. We discuss a flaw that we discovered in the interaction of Passport and Netscape browsers that leaves a user logged in while informing him that he has successfully logged out. Finally, we suggest several areas of improvement.

Keywords: Web Security, Single Signon, Authentication, E-commerce

1. Introduction

It has become common practice for retailers, banks, service providers, and just about everyone else to provide customers with a way of shopping on the Web. To ensure the security of these customers' financial data, the online vendors often require a username and a password to access an account. Users are faced with a dilemma when creating multiple accounts. Do they use the same name and password for all the accounts? If so, that means that, for example, their online grocery store will have access to their stock trading account. The alternative is to maintain a list of usernames and passwords. This list must be written down, as so many names and passwords are cumbersome to remember. As such, compromise of this list constitutes potential for a serious loss.

Single signon is the term used to represent a system whereby users need only remember one username and password, and authenticated a can be provided for multiple services. Kerberos [1] is an example of a system where users provide a password and receive a ticket in exchange. The ticket can be used to authenticate users to different network services. Kerberos single signon is possible because all of the services are under the same administrative control. There is a centralized database containing keys that are shared with each service, and tickets can be issued, encrypted under the keys of the target services.

Single signon on the web is much more difficult. Different web sites are under completely different administrative control. Thus, it is not natural to imagine signing in once, and gaining authenticated access to multiple, independent web services. Passport is Microsoft's ambitious attempt to provide this service. While the overall architecture makes sense given the constraints of the protocol, (namely, to use only existing web technologies that are present in most browsers and servers), there are some risks associated with using this protocol that are not pointed out in the paper. We refer to the online description of Passport that can be found at **http://www.passport.com/business/content_whitepaper2.html**. The draft we refer to is the one at the time of this writing. We were unable to locate a paper copy to reference.

As just mentioned, one of the constraints of Passport is that it was designed to use existing web technologies, so that clients and servers need not be modified. The protocol leverages HTTP redirects, Javascript, cookies, and SSL. While Javascript is not absolutely required, it is highly recommended. Some of the attacks described below result from some fundamental problems with security on the web, and in particular, the public key infrastructure that is built into browsers. As such, they are not specific to Passport, but nonetheless represent risks of using that system (and any system subject to these constraints).

2. The problem with SSL

SSL is a wonderful protocol. It is well designed, has withstood much analysis and scrutiny [2], and its deployment is probably the single most positive step towards anything resembling security on the web. While we find no fault in the SSL protocol or its implementations in browsers and servers, we believe that the certification model and user interface can lead to problems.

Browsers come with many default "root" public keys. For example, Netscape Navigator 4.5 comes with 58 root public keys. Anyone who controls the corresponding private keys can issue certificates that are automatically trusted by all major browsers. All it takes is for one of the certifying authorities with a weak policy, security breach, or intentional compromise (e.g. bribe) for the certification process to be meaningless. If an entity can obtain a certificate from a trusted authority, then the only recourse of the user when presented with a "secure" web site is to check the security information and determine that the root CA that signed the certificate is one it trusts, and that the name in the certificate corresponds to the actual entity with which it wants to have a secure session. Most users are not qualified to determine either of these things, and are probably not even aware of SSL or certificates anyhow.

As it stands, the SSL model does not lend itself naturally to the problem of delegation. This is exactly the feature that Passport requires. So, Passport uses the existing web technologies to the best of its abilities. Unfortunately, the resulting protocol poses several risks to the user, and these are the focus of this paper.

3. How Passport works

In this section, we describe the Passport single signon and wallet protocols. In the Passport model, there are three entities: the client at a web browser (usually a consumer

who has previously registered with the Passport service), the merchant (a store or collection of stores wishing to market to the consumer), and the Passport login server. The login server maintains authentication and customer profile information for the client and gives the merchant access to this information when permitted by the customer. Passport divides client data into profile information (such as addresses, shoe size, and so on) and the *wallet*, which contains credit card information. Passport's protocols are designed to enable the secure transfer of this profile and wallet information between the Passport server and the merchants.

3.1 Single signon protocol

Passport's interaction with a user begins when a client, visiting a merchant site, needs to authenticate (to provide some personal information or make a purchase). The merchant web server redirects the customer's browser to a well-known Passport server. The Passport server presents the user with a login page over an SSL connection. The user logs into the Passport server and the Passport server redirects the user back to the end server. Authentication information is included in the redirect message in the query string. This information is encrypted using triple DES with a key previously established between Passport and the merchant server. The end server then sets an encrypted cookie in the client's browser. This is illustrated in Figure 1.

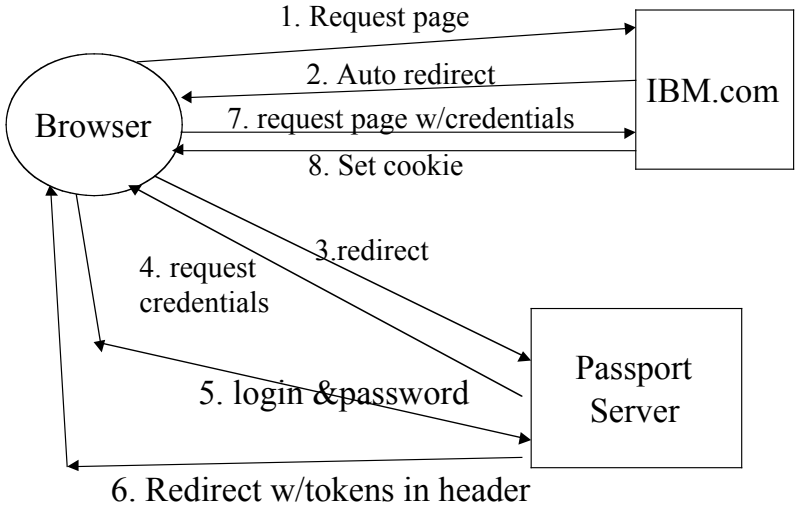

Figure 1. The Passport architecture.

The idea is that when a user returns to the IBM site, for example, the encrypted cookie is returned as well. The site can decrypt the cookie and verify the user is already authenticated. The Passport server also sets a cookie. Thus, if a user visits another site, say dell.com, when the browser is redirected to the Passport server, the user is no longer presented with a login screen because the previous Passport cookie is used. If this cookie contains valid credentials, the client is redirected back to the merchant server without user intervention.

3.2 Wallet protocol

The wallet protocol is very similar in nature to the single signon protocol. Instead of just authenticating, however, the user can insert all sorts of personal and credit card information. Then, when the user is shopping on an end server site, the user can select which information to include for that merchant. The user never needs to enter the information again for participating end servers.

4. Risks of Passport

In the following two sections, we look at some practical risks of the Passport protocol and some specific attacks.

4.1 Practical matters

In this section, we look at some security issues related to the chosen architecture and protocols. These are meant to highlight some of the risks of using Passport, without discussing specific attacks. Those are covered in the following section.

4.1.1 User interface

In security systems, the user interface is one of the most crucial and frequently inadequante parts. For example, any merchant site that uses Passport displays a Passport signout icon which is supposed to remove Passport cookies. One of the most popular Passport services currently deployed is the Hotmail e-mail service from Microsoft. We set up some Hotmail accounts and did some experimentation. One thing we discovered is that in addition to the Passport signout icon, there is also a Hotmail logout option on the page. So, what does this mean to a user? Presumably, the Hotmail logout button is used to remove the Hotmail credentials, while the Passport signout button is used to remove the Passport credentials to all services. While this may be clear to computer security experts, it is unlikely that the average non-expert computer user will understand the distinction. A user making the mistaken, but reasonable, presumption that the Hotmail Logout button will remove Passport credentials could easily walk away from a browser still able to authenticate on behalf of the user.

A curious interaction between Netscape's browsers and the Passport signout process demonstrates how difficult it is to design a user interface that makes sense and works correctly. After signing in to Hotmail with our Passport credentials, we clicked on the Passport signout icon. The redirect occurred, and our screen said that the Hotmail credentials were being removed. The point was made with a page that said Passport credentials were being removed and which displayed a check mark next to the Hotmail service. We were then redirected to a generic page with links, news stories and shopping opportunities. One of the links on that page was to Hotmail. When we clicked on that link, we found ourselves back in our Hotmail account, without reauthenticating. We tried this on several different machines. It turns out that regardless of whether we clicked on the Passport signout button or the Hotmail logout button, the message that we were logged out appeared, but in reality, we were not logged out. We tried this in Internet Explorer, and the logout was successful. There is something about the interaction of Passport and Netscape that invalidates the Passport logout procedure (at the time of this writing).

The risk of this flaw in Passport on the Netscape platform is clear. Imagine a user reading Hotmail e-mail from a public browser. When finished, the user clicks the signout button and is told that the credentials have been removed and that he is signed out from the account. The user confidently walks away from the machine. Then, when another user steps up to that machine and goes to Hotmail, he is automatically logged into that account.

It turns out that the flaw that we discovered was a problem with the Microsoft Passport server when running Netscape with the option to only return cookies to the server from which they came. We pointed out this flaw to Microsoft. Microsoft indicated they were already aware of the flaw, and the it was fixed that same day. Nonetheless, the problem makes an important point: the Passport user interface indicated that the signout process had succeeded without ensuring that it had. The value of the data maintained by Passport (its users' identities) turns this otherwise relatively minor and obscure user interface gaffe into a potentially dangerous flaw.

4.1.2 Key management

In this section, we identify a areas of potential concern with regard to the usage of keys in Passport. The Passport protocol requires that the Passport server share triple DES keys with each merchant. The keys are used to encrypt information transferred from Passport to the merchants in redirect messages. These keys must be generated securely, i. e. randomly, and assigned out of band. Many systems have been broken because poor randomness was used to generate keys (e.g. [3]). It is a difficult problem that requires careful attention.

Assigning the secret keys out of band is a nontrivial task. The mechanism for transferring keys is not stated in the current version of the specification. Ideally, these keys are transferred by physical mail or over the phone. The intuitive solution, to transfer the keys over an SSL connection, requires authentication of the merchant in some way, and is likely to lead to potential breaches.

Passport encrypts information for itself and stores the information in Passport Cookies on client machines. A single key is used to encrypt all of the cookies. This represents an unnecessary risk of exposure of that key. A better solution is to use a master key to generate a unique key per client. This is accomplished as follows:using the master key that is currently used to encrypt cookies, generate a unique key per client by encrypting the client address with the master key, and using the resulting ciphertext as the encryption key for that client. Thus, the master key is used only as a key encryption key. If an individual key is compromised, Passport cookies on other clients are not directly vulnerable.

To illustrate, take three clients, CLIENT_1, CLIENT_2, and CLIENT_3, and say that the passport master key for storing keys in the browsers is MK. Assume that CLIENT_n represents the IP address of client n. To store a cookie on client CLIENT_1, Passport computes K_1 = 3DES(MK, CLIENT_1) and uses K_1 to encrypt the cookie that is stored on CLIENT_1. Likewise, Passport computes K_2 = 3DES(MK, CLIENT_2) and K_3 = 3DES(MK, CLIENT_3). Now, if Passport later receives a cookie from CLIENT_n, it first uses MK to compute the key for that host and then decrypts the

cookie. Of course, this solution would not work for clients whose IP address changes frequently, so perhaps using the domain name is a better idea. The main point is that a compromise of one of the keys does not compromise cookies stored on other hosts.

Since triple DES is used, the three keys needed can be obtained by encrypting the client address with each of the three keys in the master key in turn and using the resulting three ciphertexts as the triple DES keys.

4.1.3 Central point of attack

As with all single-sign-on systems, Passport establishes a service trusted by all others to make authoritative decisions about the authenticity of a user. Whereas in traditional web authentication each merchant is responsible for safeguarding the authentication information of customers, all data is centralized in Passport. Compromise of this central service would be particularly disastrous. Besides authentication data, the Passport login service maintains consumer profile information on all registered users. Storing this information in a central location, while convenient, makes the server an extremely attractive target for attack, both for denial of service and unauthorized access. The centralized service model is antithetical to the distributed nature of the Internet that has made it so robust and so popular.

The effects of a denial of service attack on the login server are particularly acute. Obviously, the usefulness of a system like Passport increases in direct proportion to the number of merchants who subscribe. But as the number of merchants supporting the service grows, the effects of an outage (deliberate or accidental) increase. An operator of a large online shopping site not affiliated with Passport might see a significant increase in traffic (and hence income) by making it impossible (or even difficult) for Passport users to access their Wallet and Passport. The unscrupulous comepetitor might accomplish this by simply flooding the Passport site with bogus profile registrations or logins.

The usual response to such problems of service availability is to replicate the service sufficiently to make catastrophic failure unlikely. No information is provided on how the system could handle the fundamental problems of key distribution and database replication in scale. Furthermore, replicating the service would require multiple copies of the secret keys shared with the merchant and the master secret key of Passport, thus increasing the exposure of those keys to risks of compromise.

One specific denial of service attack that exists is due to the fact that single signon tokens are stored as cookies in the browsers. An active attacker can impersonate the Passport server and delete cookies at will on the clients. Furthermore, attacks such as the Cookie Monster bug (See **http://help.netscape.com/kb/consumer/981231-1.html** and **http://homepages.paradise.net.nz/~glineham/cookiemonster.html**.) for domain names outside of .com .org .gov .edu .net and .mil could easily overwrite merchant cookies on any client.

4.1.4 Cookies and Javascript

The Passport white paper describes two web technologies used in support of Passport. Cookies are used to store encrypted credential information in the browsers, and Javascript

is used to "make certain transactions more efficient (fewer redirects) and also to enable co-branding for participants on most centralized web pages." According to the white paper, Javascript may be disabled without significantly impairing the function of Passport (the system is said to "gracefully degrade"), but the system will not function without cookies.

Usually, the danger with cookies is limited to the exposure of sensitive cookie payloads to unintended recipients. Because the Passport cookie contains sensitive data, the system encrypts these cookies using triple DES (as described above). Passport cookies, though, are also proofs of authentication whose lifetimes are determined only by the lifetime of the web browser and the (encrypted) time window in the cookie. On a public machine, a user who forgets to log out of a Passport account could leave valid authentication tokens behind on the machine for any user to recover.

The most important problems, however, with cookies and Javascript are more social than technological. Regardless of their actual value or security, these technologies have been shown to compromise user privacy. Dictating (or even strongly recommending) the use of technologies which are not felt to be trustworthy in a system whose purpose is to establish trust can undermine significantly the perceived value of that system.

4.1.5 Persistent cookies

Passport leaves authenticators, in the form of browser cookies on the client machine. As the white paper states, "This option keeps a consumer signed in to Passport at all times on that computer even if the consumer disconnects from the Internet, closes the browser, or turns off the computer." The idea is to have a persistent authenticator so that users are not required to retype in their passwords. The Passport server does not have to reissue credentials if the cookie has not expired yet. As mentioned in the introduction, this is reminiscent of single signon in Kerberos.

Kerberos uses tickets, which are encrypted credentials, to establish continuous authentication within a specified amount of time, without requiring a return trip to the authentication server. However, Passport is lacking one of the fundamental properties of single signon with tickets. Namely, there is no concept of an *authenticator*. In Kerberos, the client must send an authenticator that proves knowledge of the key inside the ticket. To accomplish this, the client simply encrypts a timestamp. If the timestamp can be decrypted, the client must have used the correct key. This prevents theft and misuse of a ticket found lying on a machine. In Passport, where cookies stand in for tickets, possession of the cookie is all that is necessary to impersonate the valid user of that cookie. No further proof is required. Furthermore, the breach is undetected, and the attacker gets unlimited use of the victim's authentication information and Wallet. This is especially dangerous if a user uses Passport on a public machine, or if the user's machine is broken into. Given the recent surge in e-mail viruses that compromise integrity and privacy, it is not unreasonable to assume that attackers may get access to a user's cookie file.

4.1.6 Automatic credential assignment

To demonstrate the ability of Passport to scale, all of Microsoft's Hotmail accounts were automatically moved on top of Passport. In a sense, every Hotmail user id and password became Passport credentials, and when users log into Hotmail, they actually run the Passport protocol, with the Hotmail server acting as the merchant. Unfortunately, Hotmail has been fraught with security problems. One compromise allowed an attacker to log into any Hotmail account without knowing the password (See **http://www.zdnet.com/zdnn/stories/news/0,4586,2323960,00.html**). This presents a problem if users use their Hotmail credentials, which are already automatically usable as Passport credentials, to shop online with other merchants. Any compromised account, and for that matter any future compromise of Hotmail, could result in abuse of their account at these other merchants.

4.2 Attacks

In this section, we look at some specific attacks on the Passport protocol that can result in compromise of user credentials and wallet information.

4.2.1 Bogus merchant

The bogus merchant threat is probably the weakest aspect of Passport. Imagine that users get accustomed to using Passport. They enjoy the convenience of single signon and wallet services, while trusting that the service is secure. Perhaps the first time that they use the service and authenticate to the Passport server they actually bother to check the certificate in the SSL connection. It is unlikely that they will do this. It is even more unlikely that they will continue to check this certificate every time they return.

Now, to attack the system, a merchant sets up a phony web store, selling something attractive. In addition, the attacker obtains a certificate for a domain he has set up, called pasport.com. Notice that pasport.com is an incorrect spelling of passport.com. The attacker must convince some legitimate certificate authority to certify his use of the domain name. Given the aforementioned quantity of root certification authorities, the existence of one vunlerable to deception seems likely.The attacker sets up his web site with all of the Passport images that would appear on a legitimate Passport customer site. When the user visits the phony web site, the server simulates a redirect to pasport.com, and the user is prompted for his credentials on a page that looks exactly like the legitimate Passport server. The user is in the habit of filling this out every once in a while, so he does not notice the misspelled URL, or check the certificate. Even if he did check the certificate, he might not notice the misspelling. In practice, any URL, even one that does not resemble the word "passport" would probably work as well.

After the user fills in all of his information and submits it, the bogus web site can proceed to process the request any way the attacker wants. The important thing to note is that the attacker has obtained the user's valid authentication information, and he can now authenticate to Passport on behalf of the user, use wallet services, sell the credentials, or exploit them any way he pleases.

The fundamental problem is that users tend to inherently trust the web, and services such as Passport serve to increase that confidence. However, by simply simulating a valid merchant, an attacker can abuse this confidence to bypass the entire system.

4.2.2 Active attack

An attacker with access to the network between the client's web browser and the merchant server (and able, therefore, to rewrite packets passing between the two hosts) can take advantage of this access to achieve the same result described above while permitting the client to interact normally with the merchant site. Such access is not prohibitively difficult to obtain. Large ISPs concentrate the traffic of thousands of users through a fairly small set of routers and servers. Obtaining unauthorized access to one of these hosts interposes the attacker between these users and all services they wish to access. Any traffic passing through such a compromised host could potentially then be read and rewritten.

As before, this attack makes use of the fact that users are unlikely to check the contents of URLs or certificates except under extraordinary circumstances (such as when the web browser complains of a mismatch between the certificate and server URL). The attack also relies on the attacker's ability to identify when the Passport authentication process begins. This is fairly simple. Imagine a client communicating with a merchant server, using a login service at **www.passport.com**. The attacker, waiting between the client and merchant site, watches for an HTTP redirect to www.passport.com. The merchant site is required to perform this redirection at the beginning of a Passport session, and the redirection is not protected by SSL. Seeing this redirection, the attacker intercepts the packet and rewrites the URL in the redirection to a previously established bogus Passport server, perhaps again making use of creative domain names and legitimate certificates to make the service appear genuine. This server then acts as a proxy between the client and www.passport.com, and between the client and merchant site, impersonating the Passport service to the client and vice versa while rewriting all URLs and HTTP redirects to force traffic through the proxy. Passport's use of SSL cannot at this point prevent the proxy from reading and possibly rewriting each packet, as all SSL connections are terminated on the proxy, and the user is unlikely to notice the proxy acting on her behalf.

While an intruder who accomplishes this attack cannot read the contents of the encrypted cookies (and so cannot directly extract credit card or personal information), it would be quite simple to store the user's password and use it to retrieve the information from the stored customer profile.

Figure 2 shows the intruder's rewriting process and proxy service as a component of the Passport architecture shown in Figure 1. The attacker has established a bogus authentication service at www.psport.com, and has compromised a host in the path between the client browser and the merchant site, at www.ibm.com.

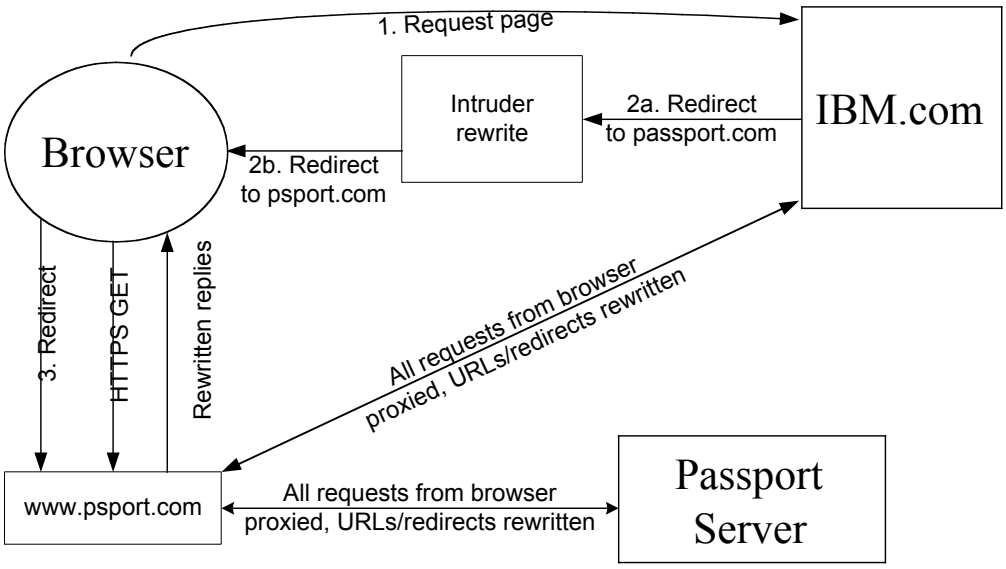

Figure 2. Rewriting and proxying requests.

A common (if weak) protection against this sort of attack, which involves having the server receiving the redirection inspect the HTTP Referer: header to ensure the referral comes from a legitimate site, will not help here, as this can be rewritten as well.

4.2.3 DNS attacks

Passport's security model depends heavily on the Domain Name System. In addition to the well-known problem of SSL's dependence on the DNS, HTTP redirects generally specify a host to receive the redirection in the form of a DNS name. An intruder who controls a client's DNS service could transparently perform the rewriting process described in the previous section by simply aliasing www.passport.com to the IP address of a server controlled by the intruder. Specifically:

- The attacker inserts the bogus record in the local DNS server
- A client is redirected by the merchant service to www.passport.com
- Resolving this hostname, the client receives and connects to the IP address of the attacker's fake Passport service
- The attacker's Passport service proceeds as described above, acting as a proxy between the client and the Passport and merchant services.

Another form of attacking the DNS is to append bogus DNS information to valid DNS replies from the legitimate DNS server. This attack was identified in [4]. Until there is widespread adoption of DNSSEC, which provides digitally signed DNS information, such attacks will be possible.

5. Conclusions

As e-commerce proliferates, the need for a tool to help users manage authentication and personal information across a variety of sites becomes increasingly critical. Passport

is an ambitious attempt to meet this need while requiring no changes to existing browsers and servers. However, the system carries significant risks to users that are not made adequately clear in the technical documentation available.

The bulk of Passport's flaws arise directly from its reliance on systems that are either not trustworthy (such as HTTP referrals and the DNS) or assume too much about user awareness (such as SSL). Another flaw arises out of interactions with a particular browser (Netscape). Passport's attempt to retrofit the complex process of single sign-on to fit the limitations of existing browser technology leads to compromises that create real risks.

Some improvement is possible in Passport without violating the system's goals of supporting unmodified browsers. Rotating the keys used to encrypt cookies would significantly increase the difficulty of retrieving cookie contents, as would using the master key to generate encryption keys instead of encrypting all cookies with the same key. Requiring SSL for all transactions would eliminate the possibility of forged redirects (at the cost of significantly increased load on merchant servers). Replacing password-based authentication with a challenge-response scheme (such as HTTP digest authentication) would make it impossible for an attacker to reuse passwords to impersonate a user.

In the end, Passport's risks may be inevitable for a system with its requirements. We believe that until fundamental changes are made to underlying protocols (through standards such as DNSSEC and IPSec), efforts such as Passport must be viewed with suspicion.

6. References

[1] J.G. Steiner and B.C. Neuman and J.I. Schiller", "Kerberos: An Authentication Service for Open Network Systems", *Usenix Conference Proceedings*, 1988, (pages 191-202).

[2] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 Protocol", *The Second USENIX Workshop on Electronic Commerce Proceedings*, 1996, (pages 29-40).

[3] I. Goldberg and E. Wagner, "Randomness and the Netscape Browser", *Dr. Dobb's Journal*, 1996, (pages 66-70).

[4] Drew Dean and Edward W. Felten and Dan S. Wallach, "Java Security: From HotJava to Netscape and Beyond", *1996 IEEE Symposium on Security and Privacy* 1996, (pages 190-200).

7. Vitae

Dave Kormann is a senior technical staff member at AT&T Labs -Research in the Online Platforms Research department, where his work
includes systems administration and Internet services development. He received the MS in Computer Science from Northeastern University in

1996.

Aviel D. Rubin is a Principal Technical Staff Member at AT&T Labs, Research in the secure systems research department, and an Adjunct Professor of Computer Science at New York University, where he teaches cryptography and computer security. He is the co-author of
the Web Security Sourcebook. Rubin holds a B.S., M.S.E., and Ph.D. from the University of Michigan in Ann Arbor (89,91,94) in Computer
Science and Engineering. He has served on several program committees for major security conferences and as the program chair USENIX
Security '98, USENIX Technical '99, and ISOC NDSS 2000. Rubin is a frequent invited speaker at computer security conferences, industry groups, and on wall street.