Security and Privacy Concerns of Internet Single Sign-On

Risks and Issues as They Pertain to Liberty Alliance Version 1.0 Specifications

Authors: Gary Ellison, Jeff Hodges, Susan Landau

Date: 6 Sep 2002

Introduction

As noted in ‘‘A Brief Introduction to Liberty,’’ for the man on the street, the lawyer in her office, the shopper or investor at home, identity on the Internet is a burdensome and off-putting process. There is one sign-on and password for the online book merchant, another for the corporate network and for each of its pieces --- the client’s private pages, the outsourced human resources office, the online travel agency --- yet another pair for accessing Lands End, still another for Bank of America. The same holds for business-to-business interactions. The result is a cumbersome user experience. The first challenge of Web services is a simple and secure identity mechanism, the second, and equally important, concern is privacy protection.

Single sign-on and federated network identity (a system for binding multiple accounts for a given user) are key to solving these issues. A federated system allows businesses to manage their own resources including customer data. Federated network identity enables customers to retain some control over which companies have access to their own information [INTRO].

The Liberty Alliance was formed to deliver and support a federated network identity solution for the Internet, enabling single sign-on for consumers as well as business users in an open, federated way. The Liberty Alliance version 1.0 specifications (referred to as the Liberty Specifications or Liberty protocols throughout this document) clear the way for a user to have a seamless Internet experience and simplify the business of conducting business with multiple Web sites. Through "Circles of Trust" --- groups of businesses that have built on-line relationships using Liberty technologies --- Liberty enables the user, whether a consumer at home or a business-person accessing different pieces of the corporate network, to sign-on once and conduct her business at various Liberty-enabled sites. The Liberty specifications are intended for deployment with existing Internet technology and are designed to be compatible with the majority of the installed base of browser implementations (web user-agents) without the need for additional software.

The intent of Liberty version 1 is to make single sign-on to multiple sites substantially as secure as giving a name and password at each site. We believe that the Liberty Alliance Project has succeeded in this goal. However, because this version is built on top of present-day Internet technology numerous security and privacy issues remain. The purpose of this document is to detail these issues. The paper doesn't present a security analysis of the Liberty version 1 protocols per se. [LIBB] does, however, in its section 4 "security considerations". Our only mention of [LIBB] is in terms of implementations, not the protocols per se.

Generally speaking, present-day Internet technology has a number of systemic security and privacy vulnerabilities, and Liberty specifications can only go so far in defining mechanisms to mitigate them. What follows are descriptions of known vulnerabilities and risks. We suggest prescriptive measures where these exist.

Security Vulnerabilities and Risks

When we speak of security vulnerabilities and risks, we are primarily concerned with system defects that compromise the system's integrity, authenticity, or availability. The presence of a vulnerability suggests that there is a risk that the vulnerability may be exploited in an attack. Attacks have different properties depending on which vulnerability is being exploited. Below are descriptions [RFC2828] of the types of attacks which are mentioned in the following sections:

- denial-of-service.
  The prevention of authorized access to a system resource or the delaying of system operations and functions.

- dictionary.
  An attack that uses a brute-force technique of successively trying all the words in some large, exhaustive list.

- replay.
  An attack in which a valid data transmission is maliciously or fraudulently repeated, either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a spoofing attack.

- spoofing.
  A type of attack in which one system entity illegitimately poses as (assumes the identity of) another entity.

The above classifies types of Internet attacks. Clearly, there are multiple vulnerabilities in the Internet infrastructure that can be exploited, and attacks on Liberty may be combinations of the above types (see, for example, the section on Web site spoofing that appears later in this document).

Our purpose in presenting potential routes of attack is not to provide a cookbook. Rather, it is to explain the security vulnerabilities that exist --- largely because of the underlying infrastructure --- so that Liberty implementors (and, to a lesser extent, users) are aware of the various risks and threats and thus exercise due care.

Risks of Single Sign-On

As stated earlier, "The intent of Liberty version 1 is to make single sign-on to multiple sites substantially as secure as giving a name and password at each site. The following are **not** insecurities in Liberty version 1 per se; rather, they are well-known Internet insecurities that can be exacerbated by the use of such Liberty-enabled technologies as single sign-on. This is a strong argument for the user and the implementor to exercise care when using and implementing web technologies.

Risks of Weak Passwords

So-called "resuable passwords" are a typical means of authenticating users. By "reusable" we mean that the password is constant and used multiple times to gate access to an account. Often, people choose weak, "guessable" passwords which render their account susceptible to relatively simple guessing attacks -- also known as "dictionary attacks". The risks of weak passwords are significantly compounded when users choose the same password to access different accounts. This is especially true in a single sign-on environment.

Using arbitrary, unchecked, reusable passwords in conjunction with a single sign-on environment means that multiple accounts may be compromised at once by guessing one password. **If** the user were to choose **different, unrelated** passwords for each site, then the multiple sites would be better protected. But if the user does not, then the vulnerability of the multiple sites is essentially the same with or without the single sign-on environment.

This vulnerability may be mitigated by performing so-called "strength checking" on users’ password selections. Strength checking essentially tests whether the chosen password meets or exceeds a randomness threshold.

Since the Liberty specifications do not specify use of any particular authentication service, both the guessable reusable password vulnerability and the strength-checking countermeasure are not specific to the Liberty specifications. However, Liberty specifications have provisions that enable identity service providers to deploy stronger forms of authentication and to convey the authentication context to service providers [LIBC].

Risks of Embedded Login Forms

The Liberty Architecture Overview [LIBA] describes a deployment scenario where an Identity Providers login form is embedded within a page presented by a Service Provider. Although users may like the seamlessness of this embedded form mechanism, which submits the users’ credentials back to the identity service provider, and deployers will like that the user does not visibly leave their web site, embedded forms carry the peril of training the user to do the wrong thing.

In this mechanism, the user is potentially revealing his identity provider credentials to the service provider in clear-text. Thus privacy surrounding the user’s identity provider account may be compromised. Additionally, a rogue service provider can now wield those credentials and impersonate the user. Thus, when using authentication via embedded form, deployers may want to consider appropriate contract terms between identity providers and service providers to address this risk.

Risks of Internet Deployment

If a user accesses a Liberty-enabled site at a public browser (such as an airport kiosk) which then hangs, there is no simple way for the user to kill her currently alive session (for example, before she leaves to catch her plane). If the browser later springs back to life, an unauthorized party which has physical access to the public browser may take over the users' browser session. However, this is not a problem specific to Liberty, but a general issue. The risks are greater with single sign-on systems because the rogue user can access all the resources available to the legitimate user.

We recommend that Liberty-based identity services have a mechanism that enable the user to later return using a different browser session and kill off the old session. We also recommend that a change password feature be available which challenges the user for their old password before accepting the new one.

To prevent session hijacking at a public browser, short session times are recommended for Identity Providers. This, of course, creates a problem for users who go off and browse, leaving their sessions inactive. But this is a classic tradeoff problem, and our recommendation is in favor of security. We recommend that in addition to short session times for Identity Providers, if, during a slightly shorter interval, the account shows activity by the user at a Service Provider, the Service Provider with whom the activity is occurring sends a refresh message to the Identity Provider.

Such a refresh message is unspecified by Liberty v1.0. Until a refresh mechanism is defined in future Liberty specifications, implementors will have to bear the burden of developing an interoperable mechanism. In lieu of a specified mechanism one plausible way to perform the "refresh" is for the Service Provider to send an Authentication Request (AuthnRequest) message to the Identity Provider over the preferred channel containing some defined combination of the AuthnRequest parameters, thus signaling that "the user is still active over here".

Risks of Weak Cryptography

One of the most vexing issues in securing web services is that the currently-installed browser base includes many browsers that only have weak 40-bit cryptography enabled. It is well known that 40-bit ciphers are considered weak and can be compromised with minimal computing effort. This poses a risk since users' encrypted communication may be easily recovered to the unencrypted form. In such cases, improving the user's privacy and security is out of the control of the Liberty-enabled sites. However, Liberty-enabled sites are committed to protecting user's security and privacy and therefore the Liberty specifications recommend cipher suites that minimally have effective key sizes of 112-bits.

Social Vulnerabilities

Social vulnerabilities abound when deploying Internet technology. Many factors contribute. A non-exhaustive list includes; human nature, rogues and ruffians, buggy software, insecure systems, bad designs, poor human interfaces, and a large installed computing base of untrusted systems. These factors present a spectrum of opportunities to exploit one or a combination of the weaknesses discussed below to the advantage of a rogue. Web site spoofing is one of these opportunities.

Web Site Spoofing

There is no way to prevent an attacker from erecting a false facade that mimics the appearance and behavior of a legitimate web site. If an attacker can lure the user into visiting such a site then the user may be fooled into believing she is visiting the authentic web site. This is sometimes referred to as the "fake ATM" attack (in fact when the Liberty Alliance Project web site first went live a rogue registered a DNS domain name that was extremely similar to www.projectliberty.org). If the fake site happens to imitate a site at which the user typically logs in, then the rogue web site may be able to collect the users credentials, for example, an account name and password. The attacker could then use this information to impersonate the user at a legitimate web site. There is nothing about this attack, or the weaknesses exploited to carry it out, that are specific to Liberty specifications. It is a general vulnerability. Detailed web site spoofing scenarios can be found in [KORRUB].

Weaknesses in Internet Protocols and Browsers

The Liberty specifications were built on existing Internet technologies, meaning both protocols and browsers. The core Internet protocols we encounter whenever online include the Transmission Control Protocol (TCP) [RFC793], the Internet Protocol version 4 (IP) [RFC791], the User Datagram Protocol (UDP) [RFC768] and the Domain Name System (DNS) [RFC1035]. When we browse the Web we add another layer of protocol with the Hypertext Transfer Protocol (HTTP) [RFC2616]. These protocols are insecure. They do not support fundamental security properties of integrity, confidentiality or authenticity. In the event that a web site needs to securely communicate with the browser the Transport Layer Security version 1.0 (TLS) [RFC2246] or Secure Socket Layer version 3.0 (SSL) protocol is inserted in a layer between TCP/IP and HTTP. This combination yields the protocol scheme known as HTTPS.

The most common tool used to access Internet resources -- and thus make use of the protocols mentioned above -- is the web browser. Web browsers also have insecure aspects.

The remainder of this section describes vulnerabilities of these protocols and browsers and the risks and threats they pose. Please note that these issues discussed here are present whenever anyone browses the Internet regardless of whether Liberty protocols are being used.

DNS Weaknesses

A DNS server resolves the host names found in Uniform Resource Locators (URL) [RFC2396] into a numeric Internet address. There are two well-known ways that

DNS spoofing can occur, and both can result in a user connecting to a bogus site and mistakenly believing it is real.

First, there is no assurance in the protocol that replies to queries are genuine and have not been tampered with. It has been demonstrated that bogus DNS Address records can contaminate the cache of an otherwise trusted resolver [BEL]. The obvious threat this spoofing attack presents is that the peer host may end up connecting to the rogue site which is not what was intended.

Another threat which surfaces due to our reliance on DNS is the possibility that of a compromised DNS server. If a DNS server is hijacked, then what seems to be legitimate address resolution may also result in the user connecting to a rogue web site.

In both scenarios the user has no way of knowing that she is not communicating with the correct host. This contributes to the "spoofing" social vulnerability discussed elsewhere.

To be more resilient to these sorts of attacks deployers can utilize SSL server authentication via HTTPS. Generally the subject name in the public key certificate bears the domain name of the server, which should match the host name in the URL used for contacting the server. Thus, along with proper certificate path validation of the server domain name from the certificate, one can verify that both that name and the host name in the URL indeed match, and are bona fide. Procedures for performing such name matching, also known as a "server identity check", are specified in [RFC2595] and [RFC2830].

Structural remedies for the DNS vulnerabilities are available but not widely deployed. The Domain Name System Security Extensions (DNSSEC) [RFC2065]] define extensions which integrity protect the records returned through the use of digital signatures. Also the security extensions provide for the optional authentication of DNS protocol interactions.

HTTP Weaknesses

HTTP, the prevalent web protocol, makes extensive use of URLs. URLs have a syntax enabling the embedding of adjunct information. Liberty makes extensive use of this capability. At times, such embedded information is sensitive. HTTP implementations must convey and consume URLs thus the information embedded in a URL must be visible to the endpoints. Thus there are no provisions in HTTP for protecting such sensitive, URL-embedded information. Therefore, the onus is upon HTTP implementations -- browsers and web servers -- to avoid inadvertently disclosing this information. URL leaks are discussed in more detail below.

Browser Weaknesses

Web browsers implement a number of features that augment HTTP’s basic functionality. These features also carry security and privacy risks.

URL Leaks

Some Liberty protocols convey sensitive information between parties in URLs. There are many ways that referenced URLs leak.

- Most browsers maintain a history of visited web addresses.
- Browsers may report to the visited web site the referring URL.
- Most web sites maintain logs of activity by capturing the URLs being requested as well as the referring URL.
- Typically Web proxy servers are deployed within intranets to facilitate passing web traffic through the corporate firewall and proxy servers also maintain logs of the URLs requested.
- Finally, firewalls typically log traffic passing through them for auditing purposes.

All of these retention points pose a risk if the data in the URL is sensitive and exposed to an unauthorized party.

In designing the Liberty specifications common-sense efforts were made to minimize the chance of disclosing the information conveyed in the URL to an unauthorized party. Two specific recommendations are prescribed by the specification:

(*i*) The specifications recommend that entry points and subsequent protocol exchanges are initiated over a secure communication transport, TLS or SSL, which implies the URLs should specify the HTTPS scheme. By following this guidance, sensitive information contained in URLs is available only at the points where it is produced, relayed (via the browser) and consumed. More simply stated, unauthorized observers can’t see the exchanged URLs.

(*ii*) The specifications recommend that state information passed in the URL be integrity and confidentiality protected. This is a privacy enhancing measure that limits the exposure of the users service provider activities. It also protects the service provider from initiating actions on behalf of the user if the state information were fabricated or tampered with.

Cookie Exposure
Web browsers implement a technology known as HTTP Cookies [RFC2965]. The intended function of cookies is to supplement web protocols with state management (or session) capabilities. Cookies can be transient (used just for the lifetime of the browser session) or persistent. A persistent cookie is saved to permanent storage so that it is available the next time the user starts a web browser. The various manners in which cookies are used may sometimes violate users’ privacy [RFC2964].

In addition, web browsers and other Internet software have been shown to be susceptible to inadvertent disclosure of cookies to unauthorized parties. Since a cookie may contain private information or possibly even could be used to impersonate a user, this represents an additional security and privacy risk.

The Liberty specifications do not mandate the use of cookies. However, the specifications describe an optional cookie based mechanism which is used to simplify single sign-on. The information in this cookie is not a serious privacy risk since the only information revealed are the locations or web sites at which the user authenticates herself. This particular cookie is referred to as the "common domain cookie" in the Liberty specifications.

However, in addition to the common domain cookie, it is recognized that many web sites, in order to provide a ‘‘seamless’’ user experience, will rely on the state management properties of cookies. Note that this is not an issue specific to the Liberty specifications. Any web site that uses cookies for state management --- with or without Liberty implementation --- is subject to the risks regarding the exposure of cookie contents. The actual risk depends on how the web site constructs their cookies, the lifetime of the cookie, and the cookie contents. Even if the contents of the cookie supplied by a web site were to present the above-mentioned risks, an attacker would still need to discover a software defect or have access to the user’s computer in order to exploit the vulnerability. A more thorough description can be found in [WEBCOOK] and [WEBDO].

Cross Site Scripting
Cross Site Scripting (CSS) was originally published as a CERT advisory [CSS] in February 2000. To date this is still a very common threat and has been used to trick browsers to make incorrect trust decisions such as erroneously trusting malicious code.

A CSS vulnerability could potentially be used to collect HTTP Cookies or the URL history and disseminate the data to an unauthorized party. Note that this is not an issue specific to the Liberty specifications. Combining a CSS vulnerability with a social vulnerability could potentially fully compromise a user’s accounts in a single sign-on environment.

Related Documents/RDF
The "Related Documents Feature" (RDF) implemented in both Netscape and Internet Explorer, "Smart Browsing/What’s Related" and "Show Related Documents" respectively, is known to be a very leaky channel [WR]. Essentially when the feature is enabled the browser reports to the RDF service the referring URL and the URL to which the browser is being navigated. Note that, again, this is not an issue specific to the Liberty specifications.

Network Time Protocol (NTP) Weaknesses
Both the Liberty specifications and the Security Assertion Markup Language version 1.0 specifications employ time-based mechanisms to qualify the validity of a message and the assertions they contain. This suggests that the clocks of participating systems are synchronized so that the validity periods can be accurately verified and honored. The time-based qualifiers may also be used as

countermeasures against replay attack described above. By enforcing the validity periods we minimize the attackers window of opportunity. The smaller the time window between the generation of an assertion and its consumption, the better the security of the protocol. Therefore, if such a countermeasure is deployed, then it will be necessary to keep the clocks of the participating sites synchronized.

NTP is designed to keep the clocks of distributed systems synchronized. NTP is not a secure protocol and measures must be taken to prevent an attacker from disrupting the service. To defend against a rogue system influencing the synchronization process by broadcasting invalid time information, authentication and access controls should be used to limit potential synchronization sources. A more thorough coverage of this topic can be found in the Sun Blueprint Series [NTP].

The Liberty Alliance made the decision to develop a set of standards for single sign-on and federated network identity building on currently-deployed browsers. The vulnerabilities described above, from the serious ones of Cross-Site Scripting to the more arcane problems of Network Time Protocol, are problems of the underlying infrastructure. The Liberty Alliance has made every effort to make the version 1 standards secure, but because the standards are built on top of insecure protocols, unavoidably there are potential vulnerabilities. This should not be misconstrued to suggest Liberty version 1 specifications are insecure, but that implementations are dependent on all the underlying protocols, and thus care must be taken in implementation.

Potential Vulnerabilities in Liberty Implementations

The security considerations section of the Liberty Bindings and Profiles Specification [LIBB] describes potential vulnerabilities that are present as a consequence of implementation decisions. Implementors are also given guidance in constructing name identifiers, which are a privacy-enhancing mechanism.

This section describes the vulnerabilities which arise if the guidance is not followed or deployers do not adequately design the web site to counter attacks. Also, two additional denial-of-service attacks are present when the implementation or deployment requires signatures on authentication requests.

Privacy Threats

Privacy threats generally are the result of poor data gathering and handling practices. Centralized data, excessive data collection, leaky channels, and linkability are the chief culprits.

As discussed above, Liberty protocols pass information in URLs. If this is done improperly, the user’s privacy may suffer. For this reason, the Liberty specifications recommend implementors protect the sensitive data carried in URLs. The methods recommended to protect this information in transit are for the service provider to use confidentiality and integrity protections for the sensitive relay-state information. Since the service provider is both the producer and consumer of the relay state information the Liberty specifications do not mandate what specific cryptographic algorithms and primitives to use.

Liberty protocols specifically enable the federating of identities. This is accomplished by binding accounts together. If done improperly, the binding could release personally identifiable information. For example, if the actual account name or password were used, then the user's privacy would be compromised. Liberty specifications avoid this problem by recommending implementations generate opaque pseudonyms that map into the account. The Liberty protocol specifies how to exchange these pseudonyms to federate accounts. Additionally, Identity Providers are required to create unique pseudonyms for each of the users federated accounts. This diminishes the threat of collusion and tracking.

However, the above mentioned use of pseudonyms does not completely alleviate the threat of rogue service providers colluding to compromise the privacy of a user. For example, rogue service providers could exchange pseudonyms during sign-on at one service provider by passing the pseudonym to the other rogue service provider. It would be very difficult for the user to detect this aberrant behavior. More generally, this threat exists for any common data (e.g. phone number) shared by rogue service providers. The only protection is for users to be cautious when they choose service providers and understand their privacy policies.

Forced Signatures on Authentication Request

Digitally signing authentication requests is optional in Liberty version 1.0. The signature serves two purposes: integrity protection and author (e.g. web site) authentication of the request. However, signing these requests potentially creates an opportunity for a denial-of-service attack against both the service provider and the identity provider.

The denial-of-service attack against the service provider simply floods the service provider with fetches to the URL(s) which triggers the generation of an authentication request. Since the intent is to make the sign-on seamless the service provider cannot distinguish legitimate requests from bogus ones.

A denial-of-service attack against the identity provider is also possible if the identity provider consumes the signature before verifying the identity of the user.

Fabricated Signatures in an Authentication Request

An identity provider which requires signed authentication requests is potentially vulnerable to a similar denial-of-service attack. An attacker can fabricate what looks like a signature and flood the identity provider with the requests. The effectiveness of this attack depends on the processing model the identity provider implements. If the identity provider consumes the signature before verifying the identity of the user then this attack could significantly impair computational resources.

Conclusion

Liberty Version 1 was designed with the intent that single sign-on to multiple sites be substantially as secure as giving a name and password at each site. We believe that the Liberty specifications meet that requirement. However, as this document makes clear, in the Internet environment, security is fragile, and implementors must carefully implement specifications. Users, too, should exercise due diligence in vetting web sites ("Did I type in www.whitestrips.com or www.whitestripes.com?") and in choosing passwords, not picking ones susceptible to a dictionary attack.

The first set of specifications are enabling technologies to pave the path for identity services but until the core Internet protocols are evolved to be secure, Liberty can do little to abate the types of risks described in this document. However, it is not likely that future Liberty protocols will be reliant upon the browser. More specifically, browsers may begin to natively implement Liberty protocols and identity services will be implemented so that the browser is no longer the conduit used to carry requests and responses. This alone could greatly improve security.

References

[BEL] Steven M. Bellovin, "Using the Domain Name System for System Break-Ins", in Proceedings of the Fifth Usenix UNIX Security Symposium, June 1995, http://www.research.att.com/~smb/papers/dnshack.ps

[CSS] CERT, "CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests", February 2000, http://www.cert.org/advisories/CA-2000-02.html

[INTRO] Susan Landau and Jeff Hodges, "A Brief Introduction to Liberty", July 2002, http://research.sun.com/Liberty/

[KORRUB] David P. Kormann and Aviel D. Rubin, "Risks of the Passport Single Signon Protocol", Computer Networks, Elsevier Science Press, volume 33, pages 51-58, 2000. http://avirubin.com/passport.html

[LIBA] Hodges, J., "Liberty Architecture Overview version 1.0", July 2002, http://www.projectliberty.org/specs/liberty-architecture-overview-v1.0.pdf

[LIBB] Rouault, J., "Liberty Bindings and Profiles Specification version 1.0", July 2002, http://www.projectliberty.org/specs/liberty-architecture-bindings-and-profiles-v1.0.pdf

[LIBC] Madsen, P., " Liberty Authentication Context Specification version 1.0", July 2002,
http://www.projectliberty.org/specs/liberty-architecture-authentication-context-v1.0.pdf

[LIBP] Beatty, J., "Liberty Protocols and Schemas Specification version 1.0", July 2002, http://www.projectliberty.org/specs/liberty-architecture-protocols-schemas-v1.0.pdf

[NTP] David Deeths and Glenn Brunette, "Using NTP to Control and Synchronize System Clocks - Part II: Basic NTP Administration and Architecture", August 2001, http://www.sun.com/blueprints/0801/NTPpt2.pdf

[RFC768] J. Postel, "User Datagram Protocol", RFC 768 , August 1980, http://www.isi.edu/in-notes/rfc768.txt

[RFC791] J. Postel, "Internet Protocol", RFC 791, September 1981, http://www.isi.edu/in-notes/rfc791.txt

[RFC793] J. Postel, "Transmission Control Protocol", RFC 793, September 1981, http://www.isi.edu/in-notes/rfc793b.txt

[RFC1034] P.V. Mockapetris, "Domain names - concepts and facilities", RFC 1034, November 1987, http://www.isi.edu/in-notes/rfc1034.txt

[RFC1035] P.V. Mockapetris, "Domain names - implementation and specification", RFC 1035 , November 1987, http://www.isi.edu/in-notes/rfc1035.txt

[RFC2065] D. Eastlake and C. Kaufman, "Domain Name System Security Extensions", RFC 2065, January 1997, http://www.isi.edu/in-notes/rfc2065.txt

[RFC2246] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999, http://www.isi.edu/in-notes/rfc2246.txt

[RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998, http://www.isi.edu/in-notes/rfc2396.txt

[RFC2595] C. Newman, "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999, http://www.isi.edu/in-notes/rfc2595.txt

[RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.isi.edu/in-notes/rfc2616.txt

[RFC2828] R. Shirey, "Internet Security Glossary", RFC 2828, May 2000, http://www.isi.edu/in-notes/rfc2828.txt

[RFC2830] J. Hodges, R. Morgan, M. Wahl, "Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security", May 2000, http://www.isi.edu/in-notes/rfc2830.txt

[RFC2964] K. Moore, N. Freed, "Use of HTTP State Management", RFC 2964, October 2000, http://www.isi.edu/in-notes/rfc2964.txt

[RFC2965] D. Kristol, L. Montulli, "HTTP State Management Mechanism", RFC 2965, October 2000, http://www.isi.edu/in-notes/rfc2965.txt

[WEBCOOK] Kevin Fu and Emil Sit, "Web Cookies: Not Just a Privacy Risk", September 2001, http://www.csl.sri.com/users/neumann/insiderisks.html#135

[WEBDO] Kevin Fu and Emil Sit, "Dos and Don’ts of Client Authentication on the Web", MIT Laboratory for Computer Science technical report MIT-LCS-TR-818, May 2001, http://www.pdos.lcs.mit.edu/papers/webauth:tr.pdf

[WR] M. Curtin, G. Ellison, D. Monroe, "‘‘What’s Related?’’ Everything But Your Privacy", October 1998, http://www.interhack.net/pubs/whatsrelated/