Web Authentication Comparisons

DRAFT 0.9
Jeffrey I. Schiller
September 4th 2005

I am writing this document as background for our meeting on Tuesday. However I am also keeping in mind that it may want to become an ITAG document and published on the Wiki. Consider this a Draft 0.9 quality document. Given that I wanted to get it to you prior to Tuesday's meeting, I haven't had time to go over it with a fine tooth comb. However I believe I have things 90+% correct.

Table of Contents

Introduction

In 1996 we decided that web authentication at MIT would be based on X.509 Client certificates, which were supported in the then popular browsers. We made this decision for several reasons, based on where we believed the technology was evolving as well as what we believed would be good security practices. Some of what we expected to come about (in terms of web authentication standards) did and some didn't. If I have time I can go into more of the history later or in person.

Be that as it may, today using X.509 certificates is but one technology in use for purposes of authentication web users, and it is far from the majority solution! The goal of this paper is to outline several alternative technologies and explain their advantages and disadvantages as well as explaining those of X.509 certificates as used on the web.

The list of technologies I will consider:

- X.509 Client Certificates
- Username/Password prompting (stand alone on each server)
- LDAP and similar schemes
- kx509 (aka “Junk Certificates”)
- Using Kerberos directly
- “Pub Cookie” based schemes (like .NET passport)

Before comparing the various technologies, it is worth considering the criteria to be used to compare them. Although I would not call all of these “requirements”, as many of them do not lend themselves well to binary comparisons. However they are all important in determining the appropriateness of a given technology both in general and in our application at MIT.

Criteria:

- Ease of use from the point of view of the end-user (client)
- Ease of implementation from the point of view of a web site author.
- Cost
- Security

X.509 Certificates

X.509 in an international standard originally defined as part of the CCITT series of documents (X.400, X.500, etc.). The Internet Community began adoption of

and use of X.509 certificates around 1992 in the original privacy enhanced mail standards. They were chosen as the way to do web authentication originally by Netscape Communications in their “SSL” protocol and then adopted by Microsoft for Internet explorer. Eventually the IETF become involved through the formation of the “X.509 Public Key Infrastructure (PKIX) working group” (that I chartered around 1994-1995).

End-User Experience:

Each MIT user must obtain a certificate on an annual basis. This certificate is stored on their own computer, optionally protected by a user chosen password. If a person has multiple computers, they may obtain a different certificate for each. The different certificates will contain the same ID, so will be equivalent as far as web sites are concerned. Client may also obtain a shorter lived certificate and renew it as they please. Obtaining a certificate requires an MIT Kerberos account and the use of a website where their Kerberos name and password, along with their MIT ID is required.

Website implementation:

Implementation depends on the webserver in use. Apache is the easiest to use, once one masters how to setup the Apache configuration files to make use of certificate authentication. Apache does all the “heavy lifting” of validating the certificate. The attributes in the certificate are exposed to programs on the website via “environment variables.” For access to static webpages a simple “.htaccess” file may be setup. This is particularly easy to do for pages served by web.mit.edu because IS&T has already added the necessary code to Apache.

The easiest way to use certificate with Apache also turns out to be the most secure as it results in all pages being served via SSL, which means that all of them are encrypted. A website author concerned about the performance of a webserver continually encrypting pages can configure the site to only encrypt the initial pages used to enter the site and not encrypt others. However this requires work to do and is often not worth it. 99% of websites at MIT never see a load sufficient to cause worry about encryption performance issues, at least not on modern hardware.

Advantages:

Very secure, as deployed at MIT, and can be made extremely secure with some additional cost. Relatively easy to implement from the point of view of a web site author. Is particularly easy to use with Apache. Web site operators need not be trusted, as nothing sensitive (such as a password) is disclosed in the authentication protocol.

Because it is a feature of SSL, by default all pages used and viewed on a website

built on certificates encrypts all information between client and server. So we need not be concerned about “sniffing” of the actual application data. This permits us to use certificate based websites for distributing student information (aka grades etc.) without too much concern.

Disadvantages:

Because it is not mainstream technology there is a learning curve both for web site authors and end-users. End-users need to obtain a client certificate and keep it on their personal computers (exception: Athena users, where it is stored in their directory which is “attached” to whatever Athena workstation they login to).

Doesn't handle public “Kiosk” machines very well. Because we expect the client certificate to be stored on the machine, a kiosk doesn't work well. This is a problem both when we want to deploy kiosks on campus (particularly at the libraries) and when MIT community members need to access MIT services from public Internet Cafe's and other public machines such as those found at airports [We can have a separate conversation about the wisdom of people accessing sensitive information from such locations, but I won't go into it here].

Revocation is often cited as a problem with X.509 certificates. Certificates have a finite lifetime, typically measured in months to years. If the private key corresponding to the public key inside a certificate is compromised, there is no good way to declare that the certificate is invalid. Technologies and techniques do exist, but they are rarely implemented. Unfortunately these techniques must be implemented in each webserver accepting certificates. On the other hand, in the nine years that we have been using client certificates we have only rarely been contacted by someone who was concerned about their certificate being compromised. We are aware of no case where a compromised certificate was used to gain unauthorized access to MIT information.

Finally, many people do not understand certificates and how they work. This results in some people obtaining a new certificate fairly frequently, when in fact they do not need to. I believe the record is someone obtaining some 80 certificates in a one month time frame.

Username/Password

This is the “standard” approach to most web authentication. It is both the easiest and riskiest approach. When viewed from the point of view of a single website, it is certainly the easiest for all concerned. However if someone has to interact with a dozen websites at MIT, do we wish them to have a separate username/password for each website?

End-User Experience:

Client is prompted to enter a username/password upon first making use of a website. Some websites can be configured to deposit a long lived cookie in the user's browser so that on subsequent visits to the website they are automatically logged in.

Website Implementation:

One of the easier ways to setup restricted access to a website. Using passwords is the “mainstream” approach to many web applications and is therefore one of the easiest to setup and use. However the “trivial” implementation has security problems and also provides no automated way for people to change their password if they forget it. Passwords may be used securely and automated password changing may be done, but this requires significant implementation work and is not done by the webservers (both IIS and Apache) “out of the box.”

Advantages

Easy to use and implement. No user training or learning curve.

Disadvantages

Usually very poor security. Out of the box, the easiest way to setup password protected access has the end-user's password going over the network unencrypted on every web transaction. [Note: “Basic” authentication obscures the password by base64 encoding it. Some vendors call this “encryption” and claim that their systems which do this are secure. We have one such situation going on now between a vendor and Audio Visual. The device uses unencrypted “Basic” authentication and the vendor claims it is secure]

Different websites have different passwords, end-users have to remember many usernames and passwords. This is somewhat mitigated by modern browsers which cache usernames and passwords. However this is a feature that must be turned on and in many browsers is implemented in a very insecure fashion. If the cache is flushed, the end-user may have a bit of a hassle remembers and/or reseting all of the remembered passwords.

LDAP and similar schemes

LDAP is a general “Directory” and attribute store. It is often touted as an authentication system because you can provide an LDAP servers with a username and password and ask if it is valid. In the case of web authentication, the webserver doesn't have a database of usernames and passwords but instead submits a username and password to an LDAP server and asks for verification.

This technique permits several websites to share a common username and password database, though it doesn't directly result in single sign on, you still have to login to each website.

This technique isn't limited to LDAP. Kerberos can be similarly used. Replace “LDAP” with “Kerberos” in the paragraphs above, and you have it. For example “Blue Socket” uses this approach for authentication to a blue socket wireless network (if configured). This is in use today by the Media Laboratory.

End-User Experience:

Very similar to username/password approach above. Rarely will the end-user even know that LDAP (or Kerberos) is evening involved.

Website Implementation:

Non-trivial. Most webservers will not do this “out of the box.” However their exists plenty of add-on software to perform this function. Many packaged web based applications and “appliances” use this approach. During initial configuration the person setting up the application or appliance is asked to configure the address of the LDAP (or Kerberos) server to use for account validation.

Advantages:

Provides for a single username/password database to be used by several (many) websites. Is often implemented in web appliances, which makes administering the appliance easier. End-user interface is as simple as username and password.

Disadvantages:

Because the webserver has access to all usernames and passwords, all servers which talk to the LDAP (or Kerberos) server must be trusted. So for example if a student run website wanted to use this approach for login, then the students administering the website would be in a position to capture the Kerberos (or LDAP) password of any website user. Similarly if the site is compromised (because it isn't run by people without a lot of security knowledge) user passwords are similarly at risk. This is also true of websites that use their own password database. However in that situation only the passwords for that website are at risk [yes, I know people often use the same password, but at least in the simple password approach people can choose not to!].

Nothing prevents the website operator from asking the password over an insecure channel (i.e., non-encrypted web login dialog) potentially putting users at risk.

KX509 (aka “Junk Certificates”)

KX509 (http://x509.org) is a combination of certificate based authentication and password (or Kerberos) based authentication. It was developed by the University of Michigan. The basic idea is that when a user needs to be authenticated a browser plugin is used to verify them via Kerberos. Upon successful authentication they are issued a certificate with a very limited lifetime, measured in hours. This lifetime is very similar to the lifetime of a normal Kerberos ticket. In essence this results in certificates which are issued and discarded in a short period of time, just the pejorative “Junk Certificates.”

End-User Experience:

Very transparent. Provided the user has the necessary plugins, applications and libraries, web authentication just happens. The actual “login” process happens when the user logs into Kerberos via “Leash” or any similar Kerberos login program (provided it has the necessary extensions for kx509).

Webserver implementation:

Very similar to the implementation of regular certificates. The only difference is in the time to live of the certificates themselves. No need to implement X.509 revocation.

Advantages:

End users do not need to be “certificate aware.” Use of certificates is completely hidden in the background. Instead users login with Kerberos as they do for any Kerberized application use. X.509 revocation is not an issue because the valid lifetimes of the certificates used is very short.

Because certificate lifetimes are short, on-campus kiosks are possible (assuming that the kiosks have the necessary kx509 software installed). Kiosks must be trusted because they will be given the end-user's Kerberos password.

Because the actual authentication occurs using certificates, no sensitive information is provided to the web server. So webservers do not need to be trusted to manipulate end-user passwords and other sensitive authentication information..

Disadvantages:

The biggest disadvantage is that kx509 requires a software download and install on every user's computer. It also requires testing (and maybe debugging) which each major release of each major browser (at least those that you wish to support). Similarly the code must be compiled for each platform. Although on-campus Kiosks are possible, off-campus kiosks (aka Internet Cafe's and airport public

terminals) are not supported because they would not have the necessary software installed.

Using Kerberos Directly

Rather then give a complete advantages/disadvantages writeup for this approach I will summarize it as being pretty much similar with the kx509 use case (one can argue that kx509 is all about leveraging the use of Kerberos). The only difference is that webservers have to implement Kerberos credential verification instead of certificate verification. I am not sure what the level of standardization is of Kerberos within http (the protocol used on the web). I know some standards were written, but I am not sure that anything was every implemented (beyond a test implementation).

“Pub Cookie” approaches

Pubcookie (http://pubcookie.org) is a project of the University of Washington. The basic idea is to provide the simplicity of username/password login combined with a single signon system, so only one login is required to access multiple services that use pub cookie. This is an approach very similar to what Microsoft does with .NET Signon.

The first time a user attempts to use a pub cookie authenticated service the user is “redirected” to a login server maintained by a central organization (aka the campus or in our case IS&T). This central server performs the login function (using password, or Kerberos or even certificates). Upon successful login a cookie is placed on the user's machine and the user is redirected back to the application. The redirect back to the application contains information encrypted by a key shared between the application server and the login server so the application knows that the redirect is legitimate. The application then maintains is own session state which includes whether or not (or to whom) the session is logged in.

The next time a user goes to the application and needs to be authenticated, or goes to a different application that requires authentication, the user is redirected to the login server. The login server sees the cookie it deposited early and bypassed any login dialog and instead creates the appropriate redirect and sends the user immediately back to the application with the appropriate encrypted message. Unless the user was carefully looking at the bottom status bar of their browser, they might not even notice the redirection.

End-User Experience:

The user interacts with the login server once per session and after that all authentication is transparent.

Webserver implementation:

Pubcookie requires that pubcookie aware software be installed on the webserver. Modules exist for Apache and IIS which do the “heavy” lifting. However the web-server application designer needs to understand how to use these modules in order to build a pubcookie based service. Furthermore if the website consists of many pages, the webserver author must manage the webserver state including the authentication state.

Advantages:

Easy to use for the end-user. Provides single signon. Secure in as much as user's passwords or other credentials are not shared with the servers. Untrusted services can therefore be provided for just like with certificates and kx509. Login server can use range of techniques to authenticate end-users. Webservers do not need to be aware of these techniques or how to implement them.

Because no special software is required for the web browser, nor are certificates or other data stored, pubcookie authentication is well suited for kiosks, both campus and public.

Disadvantages:

Requires some effort on the part of webserver application developers. Requires each webserver to be registered with the login server (they must share a common encryption key).

Proposal for MIT

We should continue to make use of and promote the user of Certificates for web based authentication. I believe this is the correct direction to follow as I believe that we will eventually see the use of X.509 certificate based hardware tokens become more mainstream (this is what they are getting read to do at Lincoln Labs). I believe that many of the problems that we have faced with certificates are literally because we are ahead of the curve. The world will likely eventually catch up, so we should not endeavor to turn the clock back!

That said, we do have several use cases that do not work well with certificates. These are almost exclusively cases which are either kiosk cases directly or cases that look very much like a kiosk situation (for example new student registration, where we want to authentication new students and computers on campus before we can expect them to have obtained their first certificate). The calendar and webmail servers are other examples.

Today these servers have an option for people to provide their Kerberos name and password. So these services have to be more “trust” then other services. We

run into difficulty when a department lab or center wants to do something similar, and we are concerned that they may not have the in-house security expertise to properly secure and protect services. Yet, we sound judgmental if we tell them that they cannot prompt for a person's Kerberos password, when we do.

Therefore I propose that we operate a pubcookie login server and use this as an additional authentication system (based on Kerberos) so we can support these kiosk like use cases without having each service manipulate a person's Kerberos password. I would further propose that we then convert those IS&T services that requires a Kerberos password to use pubcookie instead. This would include, for example, the webmail service and the calendar service, among others.