Title: Not Another Damn Password! Its the 21st Century After All

– Modernising Infrastructure with Certificate Authentication

Author: Daniel Black (daniel@cacert.org) - September 2009

Abstract:

A combination of username and password are still the building blocks of many computer authentication mechanisms. The rising number of username/password combinations for each user causes increased enterprise costs, frustration and in a global web 2.0+ community, decreased participation. Client certificate authentication can be used, developed and retrofitted to various web applications and Transport Layer Security (TLS) services as a common authentication mechanism without some of the username and password costs.

Introduction:

Username and password has been the authentication mechanism for computers since their general inception[1]. They served as a simplistic, self-contained but reasonably rugged knowledge based authentication mechanism. 40 years later it is still the most common authentication mechanism around. In many ways the ruggedness has only been slightly deteriorated with the advent of password brute-forcing, and offline cracking. Its self contained nature has extended slightly with LDAP, RADIUS and a proliferation of other standards and not so standard authentication mechanisms. However these are still largely contained to intranets and undoubtedly an application will come along that does not support the intranet standard which, over time, is followed by ten more. The result is that the intranet users now have eleven passwords and the usability of passwords authentication has got weaker due to the coping behaviours of users[2].

The second way passwords fail is that there is an Internet of communities for a person that does overlap in unique ways for every user. Authentication is introduced on a web site for reasons including spam prevention and identification of individuals is different from the next. With a web 2.0+ websites it is desirable for the community to contribute to increase the value for all users. User registration and password authentications schemes are a well understood paradigm for users/administrators however these are a barrier to entry that some contributors may not be willing to overcome when each site is different. Take for instance you had a problem with your mail client where it displayed the error 'cannot connect to host'. You searched the web for this problem and found 5 unanswered threads on different forums and one solution somewhere else. Are you willing to create five accounts on all the forums each requiring an email verification to help out the other users with problem? This one barrier has inhibited your contribution to five communities on the Internet.

1 http://en.wikipedia.org/wiki/Password#cite_note-15, Reference to CTSS Programmers Guide, 2nd Ed., MIT Press, 1965

2 http://www.schneier.com/essay-246.html, Passwords Are Not Broken, but How We Choose them Sure Is, Bruce Schneier, The Guardian, November 13, 2008

Cheaper Authentication:

Alternatives require simple paradigms that need less time to interact than password authentication should lower this barrier for at least some users.

Where the authentication aims to reduce spam other mechanism that require interactions that are hard to automate, like CAPTCHA, may be more appropriate.

For web sites that require a centralised network based authentication mechanism to people can register and have an identity on a site OpenID[3], OAuth[4], Shibboleth[5], CAS[6] are a ways of providing this.

For sites requiring a strong validation of user identity, a not networked authentication service, or where the community has certificate usage experience, client side certificate authentication may be appropriate.

Client-side Certificate Authentication:

Client side certificate authentication means that users (or to be more specific their computing devices) are in possession of a private cryptographic token (file (PKCS#12), USB token or smart-card (both PKCS#11)) that has been validated with their identity with a certification process.

The common uses of client certificates, commonly accepted to be X509 based[7], are in Secure Socket Layer (SSL)/Transport Layer Security (TLS[8]) connections and Secure MIME (S/MIME[9]) email encryption and signatures. As SSL and TLS are the foundation for HTTPS, it is not surprising that HTTPS is a common application of client certificates. Other TLS extended services like POP3, IMAP, SMTP can use certificate information to validate identity however implementations to do so are not common.

The way client side certificate authentication works is intermingled with common server authentication. Server side certificate authentication occurs when web browsers instigate a SSL or TLS connection to a HTTPS website an make sure the certificate that the server presents is signed by a certificate authority it recognises and a number of other checks. In this SSL/TLS connection negotiation the web server can request that web browser provide a certificate and this is the beginning of client side certificate authentication.

Architecture

The use cases for client certificate architecture presented here have been centred around two use cases, enterprise consistency, and global community. While the technology that supports them is identical there could be slight differences in the architecture.

An advantage in an enterprise architecture is an option to self manage a certificate authority (CA). Keeps the dependency on a external entity to a minimum at the cost of a

3 http://openid.net/add-openid/, Add OpenID to your site
4 http://oauth.net/, OAuth - An open protocol to allow secure API authorization in a simple and standard method from desktop and web applications.
5 http://shibboleth.internet2.edu/, Shibboleth
6 http://www.jasig.org/cas, Central Authentication Service
7 http://en.wikipedia.org/wiki/X509, Wikipedia X.509
8 http://tools.ietf.org/html/rfc5246, The Transport Layer Security Protocol (TLS) Version 1.2.T. Dierks et. al., August 2008
9 http://tools.ietf.org/html/rfc2633, S/MIME Version 3 Message Specification, B. Ramsdell, June 1999

little overhead. There are open source packages that support running a certification authority like OpenCA[10], EJBCA[11], PKI in a Web Page[12].

In a global community architecture to gain benefit from the a registration barrier the time/financial benefit to the user needs to exist. For this to occur, running authentication solely on your own CA is probably going to be less appealing to the end user than a password based mechanism. Using one, or more, public CAs will enable a benefit to the user that their client certificate will be usable in multiple places.

Other factors you may consider include certificates used externally? are your users going to be using certificates for S/MIME emails and is the external validation of these emails important. The importance and availability of revocation services (OCSP) and issuing services for internal and external CAs may be a consideration.

Transport Layer Security

The process of SSL/TLS connection is quickly described by Dr Yang[13] though the more complete version is in RFC 5246.

From a client certificate authentication point of view the important parts of the TLS interaction are after Server Hello in the Certificate Request message, which asks the client for a certificate issued by a list of up to 255 certificate authorities.

The client should provide a certificate that meets the server's certificate authority criteria. If the client does not have one a certificate, or the user declines its use, no certificate is provided and the server may optionally abort the connection.

It is important to note that the client certificate does not need to be signed by the same certification authority as the server.

HTTPS Client Certificate Authentication:

Typical web servers handle the TLS connection including client authentication. A number of web servers support client certificate authentication including:

- Apache
- IIS
- Nginx[14]
- GlassFish[15]
- Lighttpd soon - planned for 1.4.24[16]

The concepts apply pretty consistency across web servers so only Apache will be explained and you can translate to your own web server configuration syntax if required.

10 http://www.openca.org/projects/openca/, OpenCA PKI
11 http://ejbca.sourceforge.net/, EJBCA – The J2EE Certificate Authority
12 https://www.ibm.com/developerworks/mydeveloperworks/blogs/soma/entry/a_pki_in_a_web_page10, PKI in a Web Page
13 http://www.herongyang.com/JDK/SSL-Specification-Overview.html, SSL Specification Overview JDK (Java Development Kit) Tutorials, Dr. Herong Yang, Version 5.00, 2008
14 http://wiki.nginx.org/NginxHttpSslModule, NginxHttpSslModule
15 https://glassfish.dev.java.net/javaee5/security/faq.html#configcert, glassfish: Security FAQ
16 http://redmine.lighttpd.net/issues/1288, Lighttpd Feature #1288 SSL Client Certificate validation

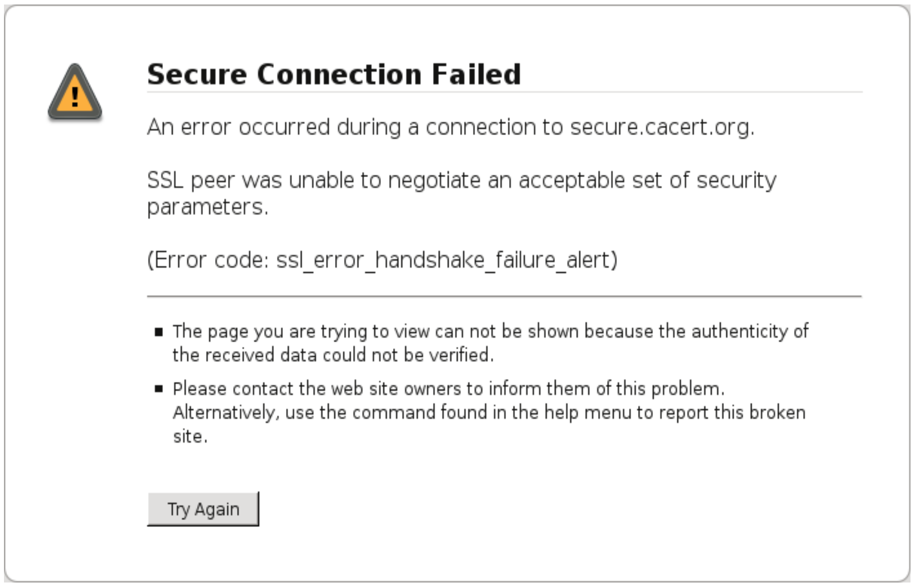
Apache Certificate Authentication Basics

To start with lets look at the basic items of Apache configuration required for client certificate authentication from the mod_ssl documentation[17]:

- SSLVerifyClient {optional/required/none} – the level of client certificate validation required
- `SSLCACertificateFile` / `SSLCACertificatePath` {file/path} the list of acceptable certificate authorities.
- `SSLVerifyDepth {number} if using intermediate CA you will need to set this higher than its default of 1.`

Those of you who have used other forms of Apache authentication before will notice that the username shows up in the Apache access logs in the third column. For auditing and debugging with client certificates web services changing the CustomLog format to include `%{SSL_CLIENT_S_DN_Email}x (or similar SSL parameter) where the standard %u column is recommended.`

You may be tempted in Apache to set 'SSLVerifyClient' to 'required' initially to mandate that the SSL client criteria must be accepted. What this means for web browser like Firefox does not contain a certificate the error looks like:



Secure Connection Failed

An error occurred during a connection to secure.cacert.org.

SSL peer was unable to negotiate an acceptable set of security parameters.

(Error code: ssl_error_handshake_failure_alert)

- The page you are trying to view can not be shown because the authenticity of the received data could not be verified.
- Please contact the web site owners to inform them of this problem. Alternatively, use the command found in the help menu to report this broken site.

Though this error message describes the SSL/TLS alert message according to the RFC it fails to give the end user on how to self correct the problem. As this is unlikely to change very soon[18], the recommended approach it to handle this error as a HTTP 403 Forbidden error using the mod_rewrite directives show in example below.

Access control by Certificate Authority

In this example we are going to allow all certificates issued by the Certificate Authority specified by the CA.crt access to this portion of the website.

```
SSLVerifyClient optional
```

17 http://httpd.apache.org/docs/trunk/mod/mod_ssl.html, Apache Module mod_ssl

18 https://bugzilla.mozilla.org/show_bug.cgi?id=419069, Bug 419069 - SSL client authentication with no installed certificate gives a bad error message

```
SSLCACertificateFile  /etc/ssl/certs/CA.crt
RewriteEngine       on
RewriteCond     %{SSL:SSL_CLIENT_VERIFY} !=SUCCESS
RewriteCond     %{HTTPS} =on
RewriteRule     .? - [F]
ErrorDocument 403 "You need a client side certificate to access this site"
```

The Rewrite condition here says that if the SSL_CLIENT did not verify to give the user a 403 (Forbidden) error with the described text (or HTML file alternately).

Static Allowed List

In this example a number of certificates are allowed to access the site. This could be useful for putting on an administration web interface where there only a few users.

The following should be within a Directory or Location directive:

```
SSLVerifyClient   require
SSLCACertificateFile  /etc/ssl/certs/CA.crt
SSLOptions        +FakeBasicAuth
AuthName          “Admin Only Area"
AuthType          Basic
AuthUserFile      /var/www/.htpasswd
require           valid-user
```

And in the file/var/www/.htpasswd contains:

```
/CN=Daniel Black/emailAddress=daniel@cacert.org:xxj31ZMTZzkVA
```

“SSLVerifyClient require” has been used here as without it failure to provide a certificate will mean the user has a username/password prompt. This is protected against using a username of '/CN=Daniel Black/emailAddress=daniel@cacert.org' and a password of 'password' (the DES decryption of 'xxj31ZMTZzkVA'[19]) however this is a barrier to usability and intuitive behaviour.

Certificates with Specific attributes

This is a small variation on the Access Control by Certification Authority example above. Here we add another requirement that the certificate is issued to an email address @example.com.

So its the same as Access Control by Certification Authority with the following added:

```
SSLRequire %{SSL_CLIENT_S_DN_Email} =~ m/^[^@]*@example\.com$/
          or %{SSL_CLIENT_S_DN_Email_0} =~ m/^[^@]*@example\.com$/
          or %{SSL_CLIENT_S_DN_Email_1} =~ m/^[^@]*@example\.com$/
          or %{SSL_CLIENT_S_DN_Email_2} =~ m/^[^@]*@example\.com$/
          or %{SSL_CLIENT_S_DN_Email_3} =~ m/^[^@]*@example\.com$/
```

We add the multiple expressions around different email address parameters as a certificate can be issued to more than one email address at a time.

19 http://httpd.apache.org/docs/trunk/ssl/ssl_howto.html, SSL/TLS Strong Encryption: How-To – Apache HTTP Server

Handling Revocation

If revocation is needed, OCSP validation of client certificates is recommended. This is built into Apache version 2.3+ (unreleased version) so handling it at the application layer seems to be the only current option (shelling to openssl command line in applications is common).

The other option is SSLCARevocationFile / SSLCARevocationPath directives and scripting a regular CRL download[20].

Safari Certificate Handling

After a few bug reports and preliminary it seems that Safari does not always pass on the request for a certificate to the user for KeyChain. Observations have shown on Safari (identified by user agent "Intel Mac OS X 10_5_8; AppleWebKit/531.9; Version/4.0.3 Safari/531.9") will terminate a TLSv1 connection on receiving a certificate request from the server and reconnect using a SSLv3 connection.

At the time of writing the full behaviour of Safari has not been determined however hopefully by the time you read this the behaviour and workaround will be described on the CAcert wiki http://wiki.cacert.org/wiki/CAcert?action=fullsearch&context=180&value=Safari&titlesearch=Titles.

Programming with Certificate Authentication

Note that so far all authentication has been controlled exclusively by Apache. The underlying application has no idea as to what authentication took place.

To let web applications that recognise the REMOTE_USER environment variable to describe their user, you can utilize this using the directive "SSLUserName SSL_CLIENT_S_DN_CN" (or other variable) in the configuration to pass this along to the application.

The generic form of certificate authentication below uses optional authentication and leaves it up to the application to work out if the authentication took place.

```
SSLVerifyClient optional
SSLCACertificateFile  /etc/ssl/certs/CA.crt
SSLVerifyDepth 5
SSLOptions +StdEnvVars +ExportCertData
```

Mandatory authentication where Apache handles the errors is possible using the mod_rewrite directives from the Access Control by Certification Authority example previously described.

With these now the application can access the SSL variables[21] that Apache has requested and parsed if provided.

You can access these variables in the languages depended on the web server/language module interface. For example in Perl you would use:

```
if (($ENV{'SSL_CLIENT_VERIFY'} eq 'SUCCESS')
```

20 https://lists.cacert.org/wws/arc/cacert-sysadm/2009-08/msg00066.html, Cacert-sysadm: tip(s) of the day: crl fetching with wget, Daniel Black, August 2009

21 http://httpd.apache.org/docs/trunk/mod/mod_ssl.html#envvars, Mod_ssl Environment Variables

and for PHP either of:

```
$email = apache_getenv('SSL_CLIENT_S_DN_EMAIL');
$email = $_SERVER['SSL_CLIENT_S_DN_EMAIL'];
```

Guidance for Web Programming with Certificate Authentication

Now that you have the basics of a certificate issuing policy, a web server requesting the certificate you want, how to you go about providing an authentication framework on a web application?

One of the hardest choices to make when doing certificate authentication is working out what is the primary identifier for the user. This is because certificates expire or get revoked and usually you want the user's registration to be seamless across this change.

To take into account certificate changes there are a number of approached here:

- ignore requirement – use a serial number (and a certificate authority identifier if more than one CA);
- use email address(es) as distinguishing identifiers;
- use the common name as a distinguishing feature;
- use an API (if exists) of a certificate authority to determine a unique ID; or
- look for a unique attribute of the certificate that will occur next in the next release of the certificate (and assume the user will use the same CA).

As email address may be the one that you choose, some code that extracts emails from certificates can be seen in this roundcube plugin in the get_username function[22]. Looking at this plugin there is a lack of standardisation as to where to place email information in a certificate until all certificate authorities implement RFC 5280[23] when the SubjectAltName will contain email addresses clearly and in a standard form.

Guidance for Non-Web Programming with Certificate Authentication

Looking back at the principles of SSL, TLS, and the Apache configuration above we can see that the majority of authentication is server side. What a non-webserver should provide is at a minimum what Apache provides; a list of certificates to request (SSLCADNRequestFile/Path), a list of certificates to accept(SSLCACertificateFile/Path) and a rich set of directives to map certificate information to users.

Going further supporting OCSP validation of provided certificates will make the revocation process simple. OCSP Stapling[24] is a nice addition to assist the client in validating the server. If you run multiple SSL/TLS based services on a single IP, it is recommended that

22 http://svn.cacert.cl/certificate_authentication/certificate_authentication.php, RoundCube certificate authentication plugin, Daniel Black, 2009

23 https://svn.cacert.cl/certificate_authentication/certificate_authentication.php, Request for Comments 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, D. Cooper et. al., May 2008

24 http://en.wikipedia.org/wiki/OCSP_Stapling, OCSP Stapling (RFC 4366 – 3.6)

you take advantage of Server Name Indication[25] TLS extension to provide a site by site configuration.

On the client side a selection of certificates needs to be provided to the application. If the privacy of “what certificates are provided to which servers?” is important then some interface needs to manage this. Other things you should do are the standard things associated with TLS connections to servers. Please consider your server administrator and provide a Server Name Indication in the TLS client hello (its a very small code addition).

“That's all very well and good by how to I code it?”

Answer: OpenSSL and GNU TLS both provide C libraries for SSL and TLS. Both have sample command line utilities that provide demonstrated ways to use their libraries. Openssl code reside in apps/s_client.c and apps/s_server.c. GNU TLS has src/serv.c and src/cli.c and examples in the manual[26].

Stuck With Password Authentication?

For some reason or another you may be stuck with password authentication of services like email where the clients or servers do not support certificate authentication. What you can do with these, and your PKI infrastructure, is to write a simple application that lets user's set/reset their own password using a certificate. While you are writing this you can create a special interface so the administrator can set a password for anyone (email author for PHP/Mysql example).

If you have a propriety web application that you would like to make certificate friendly you may need to make a small authenticating front end that maps the authentication and then acts as a simple HTTP(S) proxy for the rest of the application. Another option may be to manipulate some authentication store and use a HTTP redirect to redirect the user's web browser to the real application.

Future

Apache's HTTP server release 2.2.12 included in July 2009 for the first time a mod_ssl that supports Server Name Indication. As later versions of this propagate into distributions, web hosting will be able support multiple HTTPS sites on a single IP address. The only downside on the browser side is that IE (XP) and Konqueror are missing the client side support. It is unknown if this will drive down the price of HTTPS hosting. With cheaper HTTP hosting means more possibilities for client certificate authentication on reasonably low end sites.

The existence of free certificate authorities like StartSSL and CAcert may just make the barriers to entry lower for individuals.

C libraries supporting Server Name Indication, OCSP Stapling and other newer standards are setting up the base functionality that can be wrapped and put into higher libraries of like PHP, Python, Perl, Ruby in way that makes it accessible to more developers.

The future of client certificate stores like KeyChain, Kleopatra, gnome-keyring, Windows Certificate store play an important part in enhancing the usability to end users. An abstraction layer for these is really needed to support cross platform client applications.

25 http://en.wikipedia.org/wiki/Server_Name_Indication, Server Name Indication (RFC 4366 – 3.1)

26 http://www.gnu.org/software/gnutls/manual/html_node/How-to-use-GnuTLS-in-applications.html, How to use GnuTLS in applications

OCSP will become more important as the value of what certificates protect increases. Along with the value of revocation to the user, hardware tokens storing private keys becomes valuable in protecting keys from theft and misuse. Hardware token designs need to put control with the user rather than the increasingly vulnerable applications and operating systems.

Conclusion:

The default username-password model of authentication is becoming a real liability to users, systems administrators, and opensource community leaders. The need for innovation that re-examines the business requirements and examine alternate authentication mechanisms. PKI has caused much pain in the past for many systems engineers that it is potentially still being overlooked because of some pains that do not exist currently. Web servers, like Apache HTTP Server, have innovated to deliver you, the developers, administrators and architects, features related certificate authentication that you can use as an alternate authentication mechanism.

Web applications generally are being developed with flexible authentication mechanisms that allow developers to consider certificate authentication as a shared sign-on mechanism across an infrastructure. The majority of the work with certificate authenticating is handled by the web browser parsing data into information that can easily be used in web programming languages.

Non-web applications are a little behind in being able to deliver the same results however there is C level library support to enhance these applications for the future, where hopefully, the next user is not given another damn password.