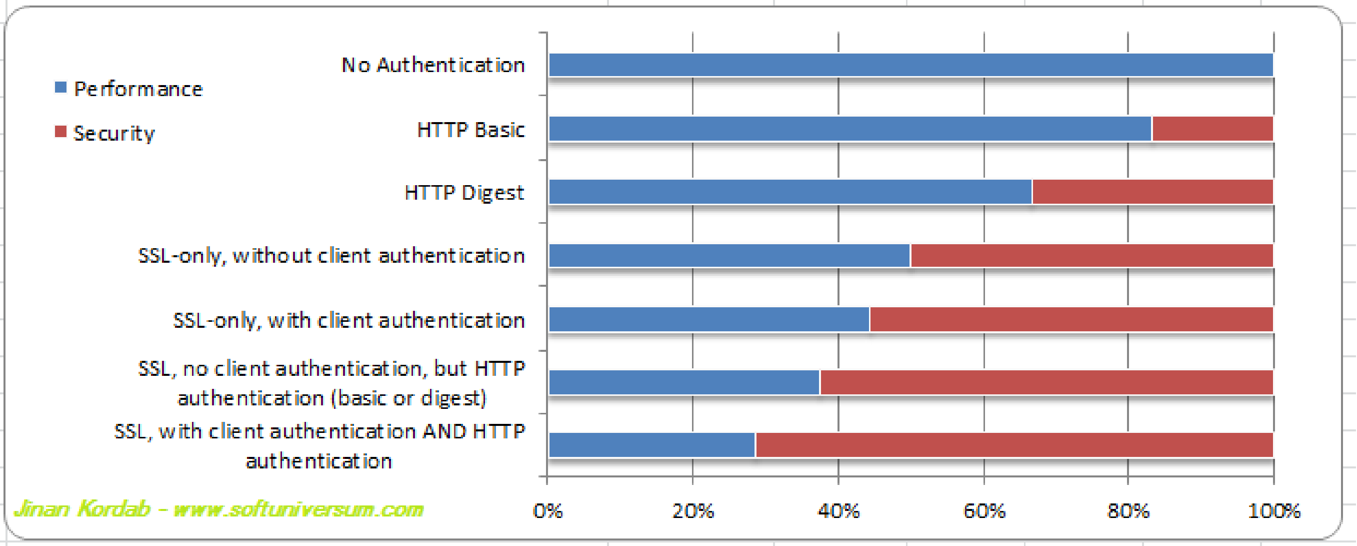
Methods of Authentication for Secure Web

Written by Jinan Kordab – Programmer – www.softuniversum.com

Whether you are designing your own implementation of OAuth, or simply want to authenticate a client to your solution, website, or online store that you have recently built, how do you do this? How do you authenticate a client? How do you know that this is Peter, Sally, John, Abed, or Margaret that is logging in? The answer is you need to use authentication methods. There are many authentication methods, each with its own security and performance characteristics. The graph below summarizes most important authentication methods you might want to use, comparing performance of each and security. I will then describe each method in more detail.



No authentication

No authentication is provided on the server side, and no access control mechanisms are applied. This means that everyone can access server's resources. Performance and server response time are maximum. Attack against server's resources is also maximized. Every bit of data that is flowing between client and server is in plain text form.

HTTP Basic

User Name and Password are required to log in to server's resources. Although the password is stored on the server in encrypted format, it is passed from the client to the server in plain text across the network. Anyone listening with any kind of packet sniffer or network analyzer will be able to read the username and password in the clear as it goes across. Also, the username and password are passed with every request, not just when the user first types them in. So the packet sniffer need not be listening at a particularly strategic time, but just for long enough to see any single request come across the wire. The content itself is also going across the network in the clear, and so if the web site contains sensitive information, the same packet sniffer would have access to that information as it went past, even if the username and password were not used to gain direct access to the web site.

HHTP Digest

In digest authentication, the password is never sent across the network in the clear, but is always transmitted as an MD5 digest of the user's password. In this way, the password cannot be determined by sniffing network traffic.

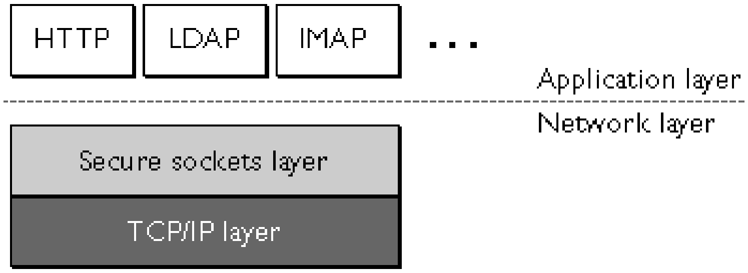
Although digest authentication has this great advantage that you don't send your password across the network in the clear, it is not supported by all major browsers in use today, and so it is best NOT to use it on a web site on which you cannot control the browsers that people will be using, such as on your intranet site. In particular, Opera 4.0 or later, Microsoft Internet Explorer 5.0 or later, Mozilla 1.0.1 and Netscape 7 or later as well as Amaya support digest authentication, while various other browsers do not.

Although password is not passed in the clear, all of the other data is, and so this is a rather small measure of security. And, although your password is not really sent at all, but a digest form of it, someone very familiar with the workings of HTTP could use that information - just your digested password - and use that to gain access to the content, since that digested password is really all the information required to access the web site.

In regards to performance, if you have only two users, it is very fast. But if you have hundreds of users, it is very slow, because basic authentication techniques store user names and passwords in text file (in encrypted format), and there is no indexing service for searching each user, so each time a request is made, each time a server must visit that text file and go one by one until it finds the match. That's why databases are more efficient in storing information.

SSL only – without client authentication

SSL was created by Netscape, and one can obtain it from Certification Authority (CA). In its default state it comes with Server Authentication only, where server authenticates itself to a particular browser, with additional option: Client Authentication. So one can use both authentication mechanisms and just the default one (only server).



Performance wise using the default SSL (without client authentication), and with Client Authentication does not change much because three way handshake remains the same, but overhead changes slightly when client authentication is also involved since client will also send his credentials to the server rather than just accepting servers credentials and replying with HTTP OK. Security wise, it is still the highest level of security available, even without Client Authentication because many strong and known ciphers are used: DES, Triple DES, MD5, RC2 and RC4, SKIPJACK and millions of web servers and users are using it.

SSL-only, with client authentication

As mentioned in previous paragraph, using additional option in SSL (Client Authentication) that comes with it, increases security and minimizes or even eliminates attacks such as brute force attacks that are very much applicable to HTTP basic and even digest authentication methods.

Performance wise, especially with Moore’s Law moving forward, speed of the transactions is almost not affected. The only difference is that a client also needs to authenticate itself to the server by sending it’s encrypted ***Digital Certificate***, that was encrypted using servers public key that was sent previously by the server.

Security wise, it is the most preferable solution to large e-commerce websites, or even intranets of a large company, where sensitive information is exchanged on daily basis.

Full SSL (with client authentication) is the advisable option for VPN (Virtual Private Networks). It is used heavily with Security Companies that have many branches around the country, and where private WANs are the basis of their daily operations.

SSL, no client authentication, but HTTP authentication (basic or digest)

Basic or Digest authentication across an SSL connection is secure, since everything is going to be encrypted, including the username and password. However, performance may suffer since the server will need to check the user first, and if the data is stored in a file (which is the most probable case scenario for basic authentication methods), and with many users, it will take time, and some servers will simply return HTTP 500 Page Not Found. But, with correct implementation, it is secure.

SSL, with client authentication AND HTTP authentication

In this case scenario, client is authenticated two times, which causes a lot of overhead. This type of authentication is advisable for Government Agencies that have sensitive data.

Considering all those secure authentication methods above, below are the threats to **Web Security** and ways to counter them using a particular feature of SSL.

1. ***Brute-Force Cryptanalytic Attack***: An exhaustive search of the key space for a conventional encryption algorithm.
    a. ***Solution***: SSL uses different ciphers for security, and administrators of SSL can enable or disable any cipher that they want. Though simple, brute force attack is only practical for cryptosystems with key size of maximum 56 bits (over 256 =

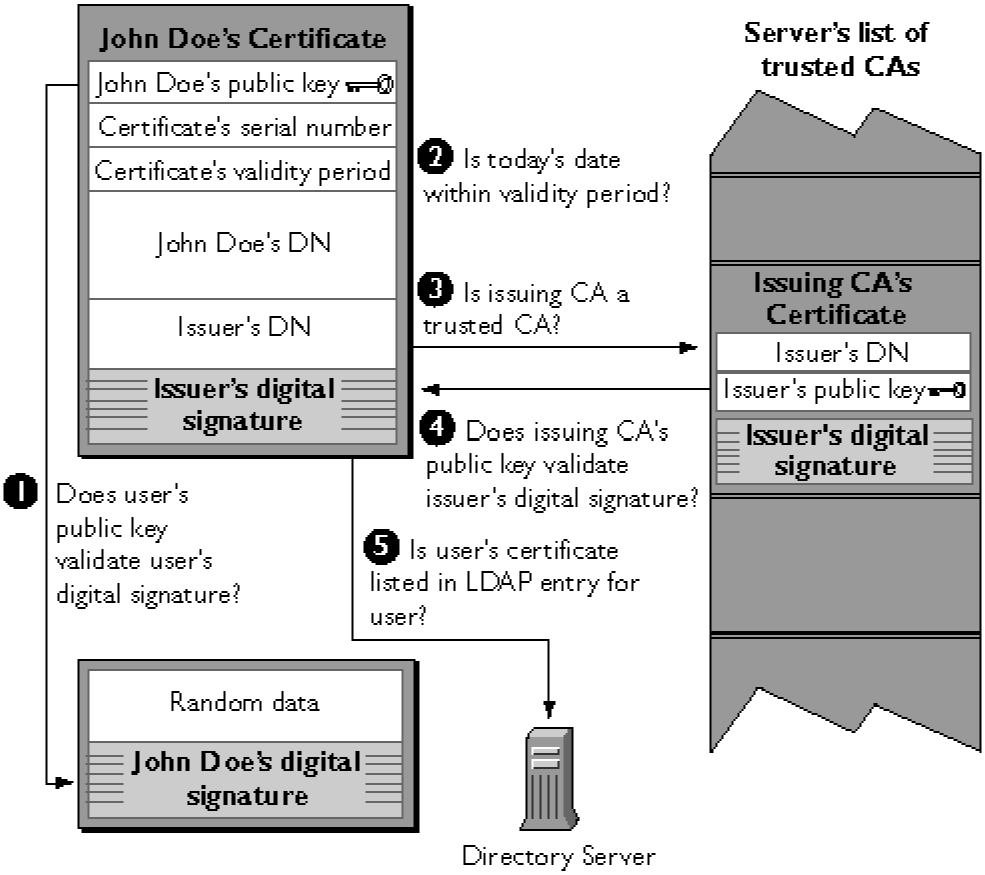
72 quadrillion tries). The current AES with a 128 –bit block size is almost impossible to crack by brute force. The number of combinations to try is more than the grains of sands on earth, many times more than a billion for every square meter on earth. So SSL can use AES to counter brute force attack.

2. **Known-Plaintext Dictionary Attack:** Many messages will contain predictable plaintext, such as the HTTP GET command. An attacker constructs a dictionary containing every possible encryption of the known-plaintext message. When an encrypted message is intercepted, the attacker takes the portion containing the encrypted known plaintext and looks up the cipher text in the dictionary. The cipher text should match against an entry that was encrypted with the same secret key. If there are several matches, each of these can be tried against the full cipher text to determine the right one. This attack is especially effective against small key sizes (i.e. 40-bit keys).
    a. ***Solution:*** Use <u>passphrases</u> instead of <u>passwords</u> to generate the private and public keys in SSL. Use Asymmetric cipher such as RSA (Rivest, Shamir, and Adelman) which uses key sizes of 1024, 2048, and 4096 bits.

3. **Replay Attack:** Earlier SSL handshake messages are replayed.
    a. **Solution:** Replay attack can be prevented using SSL in the following manner: SSL has default option where server authenticates itself to the client. The second option is when a server sends a challenge to the client. The client authenticates itself to the server by returning the client's ***digital signature*** on the challenge, as well as its public-key certificate. Digital signature can be obtained only from ***CA***.
4. **Man-in-the-Middle Attack**: An attacker interposes during key exchange, acting as the client to the server and as the server to the client.
    a. **Solution:** Even if the attacker manages to out himself in the middle between client and server, and re-rout all packets to /from client/server, he will not be able to read the contents of the data due to SSL's asymmetric encryption standards that are used (public key encryption).
5. **Password Sniffing**: Passwords in HTTP or other application traffic are eavesdropped.
    a. **Solution**: The use of private/public key in SSL with AES cipher for encryption with 128 –bit block size makes it impossible for password sniffer to crack the original password even if he has it, since the password is sent encrypted through secure SSL.
6. **IP Spoofing**: Uses forged IP addresses to fool a host into accepting bogus data.
    a. **Solution:** SSL counters IP Spoofing by using its optional option of validating and authenticating the client. Even if an attacker pretends hat his address is 10.10.100 even though his actual address is 123.10.123, this means nothing, because the server that uses SSL will send a challenge to him, where he has to provide his digital signature, and once that fails, the connection fails, and 3 way handshakes fails, and our attacker is out of business.
7. **IP Hijacking:** An active, authenticated connection between two hosts is disrupted and the attacker takes the place of one of the hosts.

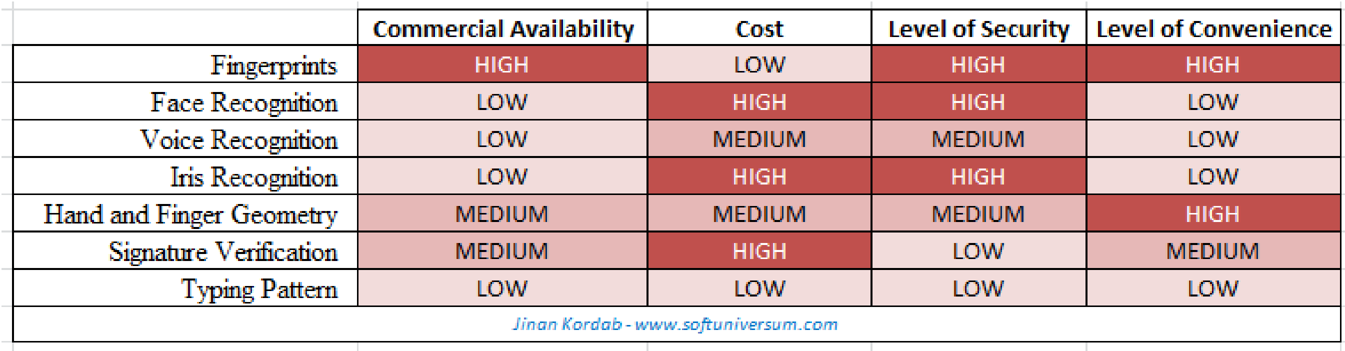
   a. Once the active authenticated between secure SSL server and client is broken, and an attacker takes place of one of the host, and tries to contact either of the hosts, the 3 way handshake will need to be established again. Here, the attacker will fail to authenticate himself, because he does not have digital signature from CA, and once that fails, the connection fails. Installing Hardware Switches also helps.

8. **SYN Flooding:** An attacker sends TCP SYN messages to request a connection but does not respond to the final message to establish the connection fully. The attacked TCP module typically leaves the "half-open connection" around for a few minutes, so repeated SYN messages can clog the TCP module.
   a.


John Doe's Certificate
John Doe's public key
Certificate's serial number
Certificate's validity period
John Doe's DN
Issuer's DN
Issuer's digital signature
Server's list of trusted CAs
2 Is today's date within validity period?
3 Is issuing CA a trusted CA?
Issuing CA's Certificate
Issuer's DN
Issuer's public key
Issuer's digital signature
4 Does issuing CA's public key validate issuer's digital signature?
1 Does user's public key validate user's digital signature?
5 Is user's certificate listed in LDAP entry for user?
Random data
John Doe's digital signature
Directory Server

Solution: If the server doesn't get to the last step for any reason, the user identified by the certificate cannot be authenticated, and the user is not allowed to access any server resources that require authentication. Therefore SYN Flooding is not possible because Half-Connection is not possible.

Biometrics is another method of authentication. Below is a comparison chart of different types of biometrics and their commercial availability, cost, their level of security, and their level of convenience.



	Commercial Availability	Cost	Level of Security	Level of Convenience
Fingerprints	HIGH	LOW	HIGH	HIGH
Face Recognition	LOW	HIGH	HIGH	LOW
Voice Recognition	LOW	MEDIUM	MEDIUM	LOW
Iris Recognition	LOW	HIGH	HIGH	LOW
Hand and Finger Geometry	MEDIUM	MEDIUM	MEDIUM	HIGH
Signature Verification	MEDIUM	HIGH	LOW	MEDIUM
Typing Pattern	LOW	LOW	LOW	LOW

Jinan Kordab - www.softuniversum.com

It is true that a security application is only as safe as its weakest link. Biometrics is no exception. Biometrics methods of authentication are only first level of security of any application, because after the authentication comes encryption, a second and most important level of security in any application. So no matter how complex and productive biometrics methods become, they will still be at first and only first level of security.

Biometrics methods can be, and will be so probably for hundreds of years to come, classified into two parts: Embedded and Global (my own classification).

Embedded biometrics are the best and most efficient, they do their work locally, which means that after authentication some action is made, either opening the door, a file, or logging in to local PC. With local biometrics, no wires or networks, no matter how big(internet, intranet, WAN) or small(wires that transmit data couple of meters long, LAN), are involved.

Global biometric methods of authentication involve some kind of a network (big or small). After the user is authenticated, the bits of data that are received, whether encrypted or not, are sent to some other place. Here is the danger, since many attacks can be made at this point: sniffing, brute force …etc. So for global biometric methods of authentication, the problem is encryption algorithms and ciphers, and not the complexity of authentication that is done,   and much research and study must be made towards improving the encryption ciphers.

Some encouraging uses of local fingerprint **biometrics**:

- Starting a car with your fingerprint. Company: National Instruments Italy , ni.com/Italy
- Logging to your PC : UCLA Students

References:

- http://www.verisign.com
- Foundations of Security [Neil Daswani, Christoph Kern, Anita Kesavan]
- Database : Lexis Nexis Academic http://www.lexisnexis.com
- Apache HTTP Authentication with PHP By Kevin Yank November 1st 2000 http://www.sitepoint.com
- CVS-Nserver Administrator's Guide Copyright (C) 2001 Alexey Mahotkin http://cvs-nserver.sourceforge.net/doc/unstable/admin-guide/html_chapter/cvs-admin-guide_2.html
- Dos and Don’ts of Client Authentication on the Web, Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster {fubob, sit, kendras, feamster } @mit.edu MIT Laboratory for Computer Science http://cookies.lcs.mit.edu/
- Software Security, an article by Gary McGraw, gem@cigital.com
- RSA Laboratories, What Is SSL, http://www.rsa.com
- http://www.instantssl.com
- http://searchsecurity.techtarget.com
- Biometric Security , an article by Jonathan Coupal
- Authentication in an Internet Banking Environment , an article by Federal Financial Institutions Examination Council 3501 Fairfax Drive • Room 3086 • Arlington, VA 22226-3550 • (703) 516-5588 • FAX (703) 516-5487 • http://www.ffiec.gov
- Biometrics in Access Control Gerik Alexander von Graevenitz, Bergdata Biometrics GmbH, Bonn, Germany, graevenitz@bergdata.com
- Specification Guide for PalmSecure, Biometric Authentication System - Fujitsu
- Enhancing security and privacy in biometrics-based authentication systems by N. K. Ratha, J. H. Connell, R. M. Bolle
- Secure SmartcardBased
- Fingerprint Authentication [full version] T. Charles Clancyy Computer Science University of Maryland, College Park tcc@umd.edu Negar Kiyavash, Dennis J. Lin Electrical and Computer Engineering University of Illinois, UrbanaChampaign fkiyavash, djling@uiuc.edu