Global Information Assurance Certification Paper

David Thompson
Version 1.2f

Securing Web Applications with PKI

There would be few who would argue that the Internet is a non-essential tool. Our dependence on the Internet, or even intranets, is unquestionable. So how then do you secure a web application on an architecture that was designed to be open and allow the free flow of information? PKI is one answer to this question. Public Key Infrastructure (or PKI) is a security system designed to validate and authenticate the identity of a user. PKI allows the administrator to determine who get access to which web application based upon their identity.

PKI – Public Key Infrastructure

The Internet has opened communications between individuals to levels never thought of before. In many cases, information has become the most valued asset that an individual or company can possess. The Internet has provided us the pathway to pass information, but has not guaranteed its security. Information that is transported from one destination to another is at risk. Public Key Infrastructure (PKI) addresses the risks associated with securing information. The concepts of PKI revolve around the use of keys to ensure trust and privacy between a sender and a receiver.

Essentially, PKI employs the use of public and private keys in conjunction with a trusted third party to mitigate. PKI hinges upon four major concepts:

- Authentication – the communicating parties are who they say they are.
- Confidentiality – the communication between parties is secure.
- Non-Repudiation – neither party can deny that the content sent between did not originate from them.
- Availability – communications are reliable (BT Ignite, 5).

Until recently, the most common method for two parties to securely send each other information was to use symmetric encryption to create a trusted channel. Symmetric encryption requires that both parties encrypt data with a common encryption algorithm. Both parties can then communicate with each other through this encrypted channel. Using this method, both parties would meet and establish a common secret key that they would both use in the future to encrypt/decrypt messages sent to each other. In order for a message to be cracked, the exact number used to encrypt the key would have to be guessed to decrypt the key. Key strength can be measured by how large in bits the key number is, so a key that is 128 bits in length is significantly more secure than an 8 or 16 bit key. Below is an example of a key:

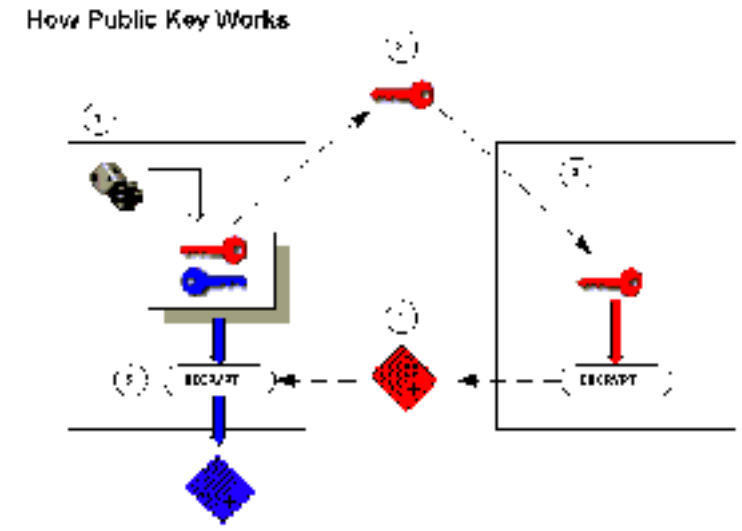
```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 7.0.3 for non-commercial use <http://www.pgp.com>

mQGi8OrTRWwR8AOkcn4OPzPvZRAI57n58ZPWY86g2tlvOzisntMQZ/ONTwwvNMO8
RAElp9U7iKj1dmOqijf1bqUTooksjHwY2kf3fPdp5vaOR75cFcmNGkUJpxpjyx1t
j84z2wwmhPzOXQmmWuhXy/tyyKJQXTvgOv/pnbAo9qR8IIsPN8R1NCx5IQCg/8S3
9+7lwZEOvNQ3ObcZ4H7n+WEEANWgNHT/L9jPxY8tdyrlPyRR2fTEhzxMOhMO64Cc
hbKk6J8McpFkqReiuXjvrJPq86ubRjfIaijCMYCuFg4bSkLllSzgRC2o1MQfN8PI
4KXuAVbecGT3rrbxaNnMvs9MV56bvktdn20toQ3L56pREYquiGzNgXHRmV8FzgR1
quM2A/9X3RCK8gqmACOPsx8fWJ8yQIAdPu68d82TAvw/eAVHPIGHTz54fW+36h8
fhOvPsC4m8s8G/S8/U9je4mqbrZsJV7cJztSUcPxA5fS3NvjMCwy1Nd8uY1mJC8t
lsGymktqgr0P3WMOx2OcJ6tMgOePIli8Mlm7jOSwIo6lxwm8bQWRGF2aWQgPGFl
d2fhQGJsYWguY29tPokAWAQQEQIAGAUCOtNFbAgLAwkI8wI8CgIZAQUbAwAAAAAK
CRAK/r92MxgnQ6TeAJ411jKVxFgyCymR7djCK8J2mGtPvgCdGC4U6/cz48LK7buv
HzQrEiC2uxC5Ag0EOtNFb8AIAPZCV7cIfwgXcqK61qlC8WXo+VMROU+28W65Szgg
2gGnVqMU6Y9AvlfPQ88bLQ6mUrfdMZIZJ+AyOvWxpF9Sh01O49Vlf3HZSTz09jdvO
meFXklnN/biudE/F/Ha8g8VHMGHOfMlm/xX5u/2RxSc8qtNbno2gpXI618rwv0YA
WCvl9Ij9WE5J280gtJ3kkQc2azNsOA1FHQ98iLMcFFstjvbzySPAQ/ClWxiNjrtV
jLhdONMO/XwXvOOjHRhs3jMhLLUq/zzhssSlAG8GNfISnCnLWhsQ0GcgHKXrKlQzZ
lp+rOApQmwJG0wg9ZqRdQ2+cfL2JSyIZJrqrol7OVekyCzsAAgIIAIW8ZFo818TZ
875/AFKyb/AeOvf4ISUKCgz63sT3aa7s2vr4Hst7peNxvAE+vuSFg6aOO7rjE/Ts
ZsjkVeOvLjOSmgeudb1bUevOkjMgMcv8UzPoaFOahMb/L8dNvIa/e0wEAaaQibMh
4hPgM28AgJqP1TX66uyrjZZqN/rw4hXlzTa+MjlFb/QEbEYUqo/160NG/cC/hhv7
8xUOPJs/iuvwI/aOz9PQTI/nISisdLy8S/YGjx88GOwryNUKeKGqjAPpbLcYxONr
wJcQc3TVNLWOfPOMXQ0okLuv8h7IoKkofnQdUT045dpebgx09wRRLYbrZ93uOxoh
AtQl4dqSH/uJAEwEG8ECAAwFAjrTRWwFGwwAAAAACgkQCV6/djMYJOMu9QCgj+G+
kfOY8QySQ6GrmixpQrtTnRkAoI8hO6TQbkXwZk7f7XtF9N1ixwtI
=8L2w
-----END PGP PUBLIC KEY BLOCK-----
```

While this method addresses the issue of confidentiality, it fails to address certain issues. The established secret key cannot be used more than once to ensure security. This can be a disadvantage if the parties require secure communication across multiple sessions. This method also has the disadvantage of making it difficult for two parties that had no pre-established relationship with each other to guarantee secure transmissions (Dickerson 2). How then do two parties securely encrypt communications? The answer lies with asymmetric key encryption.

In this model, each user has a public and private key pair that is created at the same time to be used for encryption. The two keys are inseparably linked, like a mold and a cast, when one is defunct, the other will be too. The public key is available to anyone, while the private key is kept secret and known only to the owner. The use of asymmetric keys hinges upon one major concept: *Only the public key can decrypt a message encrypted with the private key and only the private key can decrypt a message encrypted with the public, within the key pair.* Lets consider the following example (see figure 1):

Figure 1



(Cearley and Winsor)

1. A user, Alvin, wants to send the user Matt, his request for a promotion in an encrypted format. Matt first generates a key pair on his computer, which key he chooses to designate as his public and private key is arbitrary.
2. Matt places his public key in a publicly accessible place. The private key he keeps protected on his computer.
3. Alvin retrieves Matt's public key from this public directory, and encrypts his response to Matt with Matt's public key.
4. Alvin then sends the encrypted document to Matt in any appropriate format.
5. Matt retrieves this document and decrypts it with his personal private key. Using this, Alvin never has to know the secret private key for Alvin.

This model solves the problem of individuals without a previous relationship sending files securely, but it raises another question; how do you verify that the sender is who they may say that they are? What is to prevent Jack from spoofing a request to Matt to attain his public key, after he has intercepted Matt's email? (This issue of authentication is one of the faults with the popular PGP tool). This problem is solved through the use of certificates.

Certificates

Certificates are digital signatures. A certificate is nothing more than a file that binds a user to a user's identity. Certificates are digital "fingerprints". These are usually stored within a directory that the web browser uses. These fingerprints must come from a third party, or they are not very trustworthy. It would be easy for someone to sign a file with a spoofed certificate if there were no way to validate his or her information. This trusted third party is addressed through the use of a Certificate Authority (CA).

A certificate authority is a third party that issues certificates and validates an individual's identity. A certificate authority is an entity that has its own public/private key pair. A certificate is a digital ID card of an individual; it binds a person's identity to a file that is encrypted with the CA's private key. Now instead of having to obtain a public key for every individual that someone may want to securely communicate with, one would simply need to get the CA's public key through a trusted means. This prevents people from having to keep a database of public keys on hand in order to decrypt messages. The process can be simplified more if many certificate authorities grant reciprocity of trust to each other, creating a ring of trust. This is common with many commercial certificate authorities.

So how do we know we can trust a certificate issued from a CA? In the case of commercial certificate authorities, a user's information that is provided when they apply for a certificate is compared to information obtained from a large financial institution. When discussing a private CA, the issue of trust should be addressed in that you trust yourself or your organizations' data center. It is possible to further expand the web of trust through the use of a Registration Authority (RA). Registration authorities simply take a user's information and submit it to the certificate authority that implicitly trusts the registration authority. Now that we have established trust in a CA, we have addressed the

concept of non-repudiation. A user that signs a file digitally and uses a CA cannot refute the authenticity of that file.

Web applications can use certificates to validate user's identity and grant them access. A web application can use the data presented to the server through the user's web browser to authenticate that user. This leads us to the next question; how can we securely transfer certificates between client and server? Additionally, how do we know that the machine on the other end is the server or workstation that we want to talk to? This can be accomplished through the Secure Sockets Layer protocol (SSL).

SSL – Secure Sockets Layer

Secure sockets layer is a transmission protocol that runs at the Network layer of the OSI model. This means that it runs on top of TCP/IP but below other protocols such as HTTP, SNMP, and IMAP. SSL was developed by Netscape to establish secure communications as well as provide authentication. Essentially, SSL uses asymmetric key encryption between two communicants to establish a symmetric key to be used to encrypt the flow of data. Below is a general overview of an SSL session:

1. A client workstation sends the server some preliminary information such as SSL version, cipher settings and some randomly generated data.
2. The server replies with its own SSL version, cipher settings and randomly generated data. The server also sends its own certificate and requests the client's certificate.
3. The client then checks the server's certificate. If the server cannot be authenticated, then the connection is rejected and an error message is returned.
4. If the server authenticates, then the client creates the premaster secret key in conjunction with the server (depending on the cipher used). The client then encrypts this key with the server's public key and sends it to the server.
5. If the server requires the client to authenticate (as we have been discussing), then the client also sends additional data that uniquely identifies the session.
6. The server then attempts to authenticate the client. If the client does not authenticate, then the session is rejected. If the client successfully authenticates, then the server decrypts the package with its private key. The server and the client then both agree on a master secret key for the session.
7. The client sends the server a message informing the serve that transmission will now begin using the agreed upon session key.
8. The server responds in kind and the session begins (Netscape).

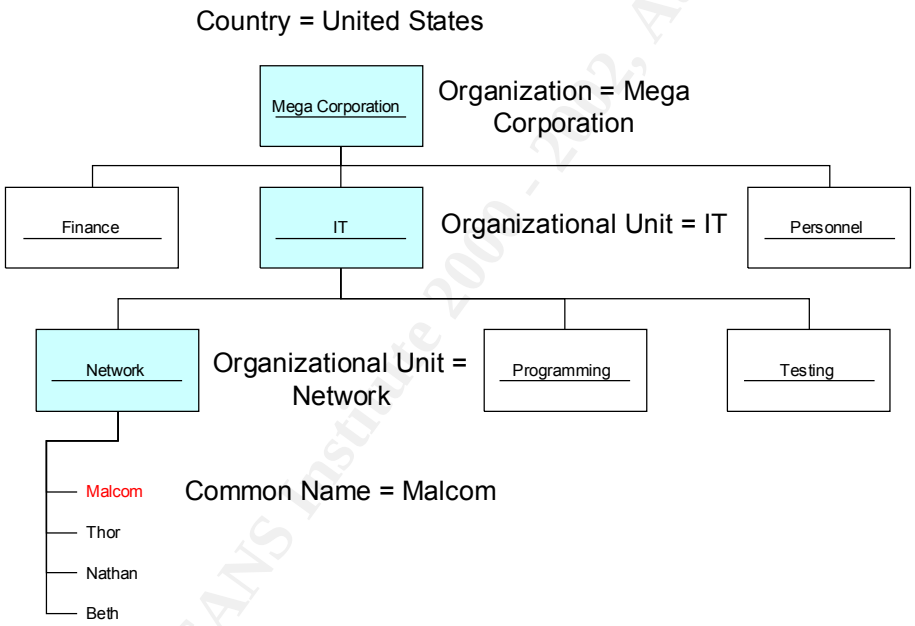
SSL is composed of two separate protocols, the SSL record and the SSL handshake protocols. The SSL record protocol is responsible for establishing an agreeable transmission between two platforms. The SSL handshake then uses the SSL record protocol to complete transactions between the two platforms. The SSL handshake then completes a list of transactions that eventually validate the server to the client. Ideally, the client will also validate with the server (Netscape). SSL uses cryptographic ciphers to guarantee security between the server and the client. As part of the SSL record protocol, an agreed cipher is used to open a channel between the server and the client. Some

examples of commonly used ciphers include RSA, DES, DSA, MD5 and SHA-1. These ciphers are used to transmit certificates, and keys between the server and client. Now we have secured authenticity and validity from the user to the server, but how does a web application access and validate all of these certificates that will be presented to it? This is where Lightweight Directory Access Protocol comes into play.

Lightweight Directory Access Protocol (LDAP) and X.500

The University of Michigan originally developed Lightweight Directory Access Protocol. LDAP is a protocol that runs over TCP/IP that is used usually to access an X.500 hierarchical directory tree structure. X.500 is a directory structure that is based upon OSI specifications. LDAP is a protocol used to query a resource directory structure that allows multiple clients to access information that is static. X.500 directories are hierarchical in that accounts are stored in containers that start general and become more specific. For example, let's consider Malcom, a network guru at Megacompany. Malcom is a member of the networking staff, which is a member of the IT division (see figure 3).

Figure 3



In this example, Malcom's distinguished name would be cn=malcom.ou=network.ou=it.o=megacorporation.c=us. Using this model, it is possible to store Malcolm's certificate in a central repository, this will allow a system administrator to apply specific rights, permissions etc. to Malcom's account. A web application can then use LDAP to communicate with an X.500 directory to access stored certificates and authenticate users against those entries. Permissions can be set on an account that will allow data to be kept private. A web application can also only use the most restrictive permissions on the directory, and then security can be maintained while authentication can still take place. This method is most effective when used on an

intranet. The only trick that a web application needs to concern itself with is extracting the user's certificate when the web site is accessed. The end result leads to a secure "single sign-on", from start to finish, a secure method for allowing users to access a web based application with discretion.

Summary

The result of putting all of these components together provides a pathway to allow users with valid certificates to present their certificate to a web application and authenticate the certificate against a LDAP directory to grant or deny access. This model allows validation users who have valid certificates on their computer, those who have an invalid certificate on their computer and users that do not yet have a certificate. Users without a certificate installed on their workstation will not have automatic access, but can still have access to the application using a simple authentication through the web page. Users, who have valid certificates, will be checked against the LDAP directory, and automatically logged in to the application. Those who have a certificate, but do not have permission to access the application will be denied access when the certificate is validated. Combining all of these elements provides a tight security model and allows users access to only the resources they are authorized to access.

References

Cearley, Kent and Windsor, Lindsay. "Securing IT Resources wit Digital Certificates and LDAP." University of Colorodo System Office. Boulder, Colorodo. 1999. http://www.educause.edu/ir/library/html/cnc9707/cnc9707.html.

Dickerson, Michael. "Implementing PKI". SC Magazine. January 2001. www.scmagazine.com/scmagazine/2001_01/survey/survey.html.

"Introduction to SSL". Netscape Communications Corporation. October 10, 1998. http://developer.netscape.com/docs/manuals/security/sslin/contents.htm#1041986

Kelm, Stefan. "The PKI Page". http://www.pki-page.org.

Markey, Bruce. "A System Administrators View of LDAP". 1998. http://www.gracion.com/server/whatldap.html.

Netscape Communications Corporation. "Introduction to SSL". October 9, 1998. http://developer.netscape.com/docs/manuals/security/sslin/contents.htm.

"Public Key Infrastructure (PKI)". BT Ignite. 2001. https://www.trustwise.com/campaigns/SecurityGuide.html

RSA Security. "Understanding Public Key Infrastructure (PKI)". 1999. www.rsasecurity.com

Whittle, Robin. "Cryptography for encryption, digital signatures and authentication". First Principles Consulting. December 19, 1996. www.ozemail.com.au/~firstpr/crypto/index.html.

Last Updated: July 26th, 2017

Upcoming Training

SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event