Implementing PKI in a Heterogeneous Environment A Primer on Digital Certificate And Key Formats

Tim Sills

Tim R. Sills
SANS Security Essentials
GSEC Practical Assignment
Version 1.2e

Implementing PKI in a Non-Heterogeneous Environment
A Primer on Digital Certificate And Key Formats

Abstract
This document will discuss the various file formats for both X.509 digital certificates and encryption keys. It will also bring to light potential issues one would face when implementing a public key infrastructure (PKI) in a non-heterogeneous environment. In particular, the focus is specifically on the topic of binary and PEM encoded digital certificates and the Public Key Cryptography Standards (PKCS) file formats. Further, the discussion will also include some hard learned lessons on the nuances of supporting and implementing diverse systems that utilize digital certificates. As we'll see, required digital certificate file formats will vary from application to application. Although a light overview of PKI and digital certificates will be provided, this document assumes the reader has some familiarity with the secure sockets layer (SSL) handshake and how digital certificates are utilized within a public key infrastructure.

Introduction
With a heightened awareness and need for increased security, the usage and implementation of digital certificates has almost become standard. Yet, varying key and digital certificate file formats makes support and implementation far from being standard. All across the Internet and on the local bookstore's shelves one can find numerous references to the importance of digital certificates, how they're used, and their role within a PKI. Yet, I have yet to find a comprehensive document that discusses the various packaging of certificate and key files.

When deploying a public key infrastructure (PKI) to your own organization, you get to set the rules and establish the standard operating procedures. Often times the required digital certificate formats will be dictated by your own system's requirements.

Yet what happens when the environment is for instance an Internet based E-commerce system with non-heterogeneous trading partners? Consider for a moment what you would do if you had to support a multitude of trading partners in a business-to-business environment with each having unique digital certificate requirements? Imagine these clients having invested significant amounts of money and resources into their own infrastructure and business practices. Their systems are in place and business processes formulated. Getting them to rethink their practices to adopt your policies will be akin to lifting the Titanic. It just isn't going to happen without a lot of money and effort.

Hence, here is the problem at hand. Because digital certificates and key pairs can be stored in a variety of formats, the resulting file requirements will vary from application to application. Just short of dictating your processes and procedures to your clients, your next best move will be as prepared as possible in understanding the various file formats and how to manipulate them. This document will introduce and define the PEM and DER file formats for X.509 digital certificates as well as cover RSA's PKCS specifications. Finally, we'll wrap up with what it all means in the real world and how you can manipulate these files to support a diverse client base.

Brief Overview, Digital Certificates and PKI

In its most basic form, X.509 digital certificates are digital documents that bind a public key to an individual or company. The certificates provide the function of authentication and conveyance of information used to establish a secure sockets layer (SSL) connection. In order to trust this relationship between an entity and the public key, it must be vouched for. This is the role of the trusted Certificate Authority (CA), which signs a certificate. The CA includes their digital signature in the certificate thus allowing the certificate holder to confirm they are who they say they are.

The concept of a PKI system is based upon the use of public and private key pairs. The keys are mathematically related and are used for the encryption and decryption of data or even the creation of a unique symmetrical session key. The key pair holder will maintain their private key in strict confidence while the corresponding public key can be freely distributed. Embedded within the X.509 digital certificate is the public key. Figure 1 provides an example of the interaction between a public key and private key during the transmission of data between two end points.

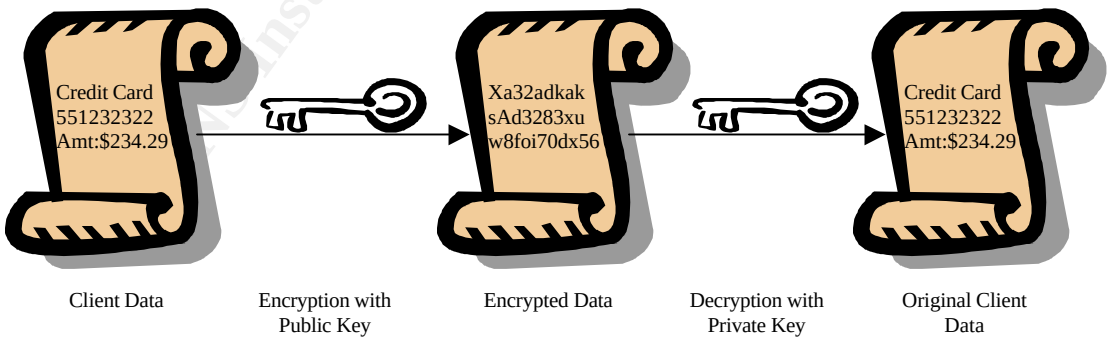

Figure 1. Key Pair Usage

The X.509 digital certificate file sizes can vary from a few hundred bytes to a few kilobytes. Some of the basic elements within a digital certificate consist of the certifying authority's name and digital signature, host name, serial number,

expiration date, and of course the public key. During the establishment of an SSL connection, this information is conveyed to a remote system. Figure 2 is a snap shot of a digital certificate displayed in NT. With the subject field highlighted, you can see some of the information contained within the digital certificate. Which in this case is a digital certificate retrieved from the SANS Institute's course registration server. The following examples will all be based upon this same certificate.

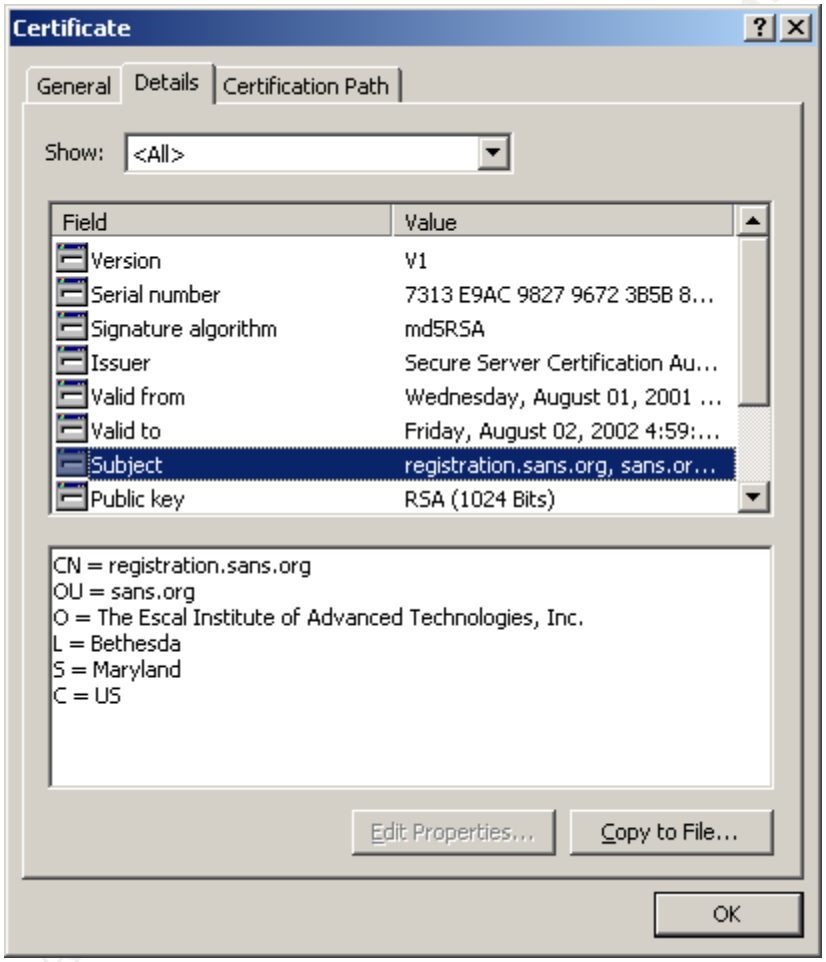

Figure 2. Digital Certificate Example

There are many elements that make up an X.509 digital certificate. And a good starting point would be Request for Comments (RFC) 2459 "Internet X.509 Public Key Infrastructure Certificate and CRL Profile". This document discusses the makeup of a certificate's syntax, defines the various fields, and covers hash algorithms as applied to digital signatures. Also, Netscape also has an excellent overview of cipher suites and the SSL handshake in their document "Introduction to SSL". With all of these references though, it is still hard to decipher the

meaning of the various file formats and understand when one is applicable over another. The next section will cover the specifics of file encoding.

Encoding X.509 Digital Certificates

As previously discussed, the X.509 digital certificate can be used to facilitate authentication and data encryption. And as referenced, there certainly is a lot of literature on the subject of digital certificates and implementing systems that utilize them. Our task though is to understand the nuances of the various file formats.

An X.509 digital certificate is based upon what is known as Abstract Syntax Notation (ASN.1). And according to Paul Tremblett within his paper *X.509 Certificates- Moving Toward Secure Communication*, the ASN.1 format is "a language used to describe data types in such a manner as to eliminate ties to any particular platform". In a sense, making digital certificates universal and independent of applications and operating systems. The X.509 data can be encoded in either a binary or ASCII form. These files are known as DER or PEM based files. DER, distinguished encoding rules, is the binary representation. Using Notepad to view the DER file, Figure 3 displays the binary representation of a digital certificate.

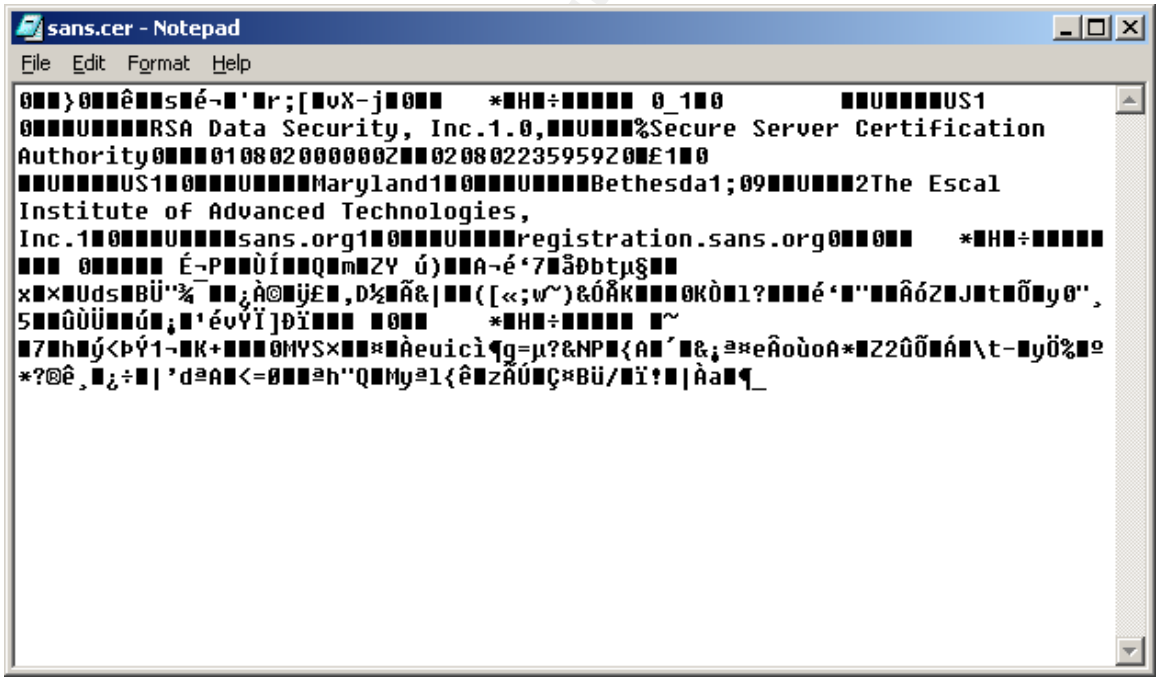

Figure 3. DER Encoded X.509 Digital Certificate

A PEM encoded format is essentially the same X.509 digital certificate but in an ASCII form. PEM, Privacy Enhanced Mail, is a Base64 version of the DER file wrapped with the "----BEGIN CERTIFICATE----" and "----END CERTIFICATE---" lines. This format is used for ease of transport and to facilitate cutting and pasting of the data. With the particular PKI application I currently manage, pasting and copying data is the only way to input and extract the digital certificates. But as we'll see later on, transitioning between PEM and DER is a piece of cake if you have the right tools. Figure 4 is a PEM file opened with Notepad.

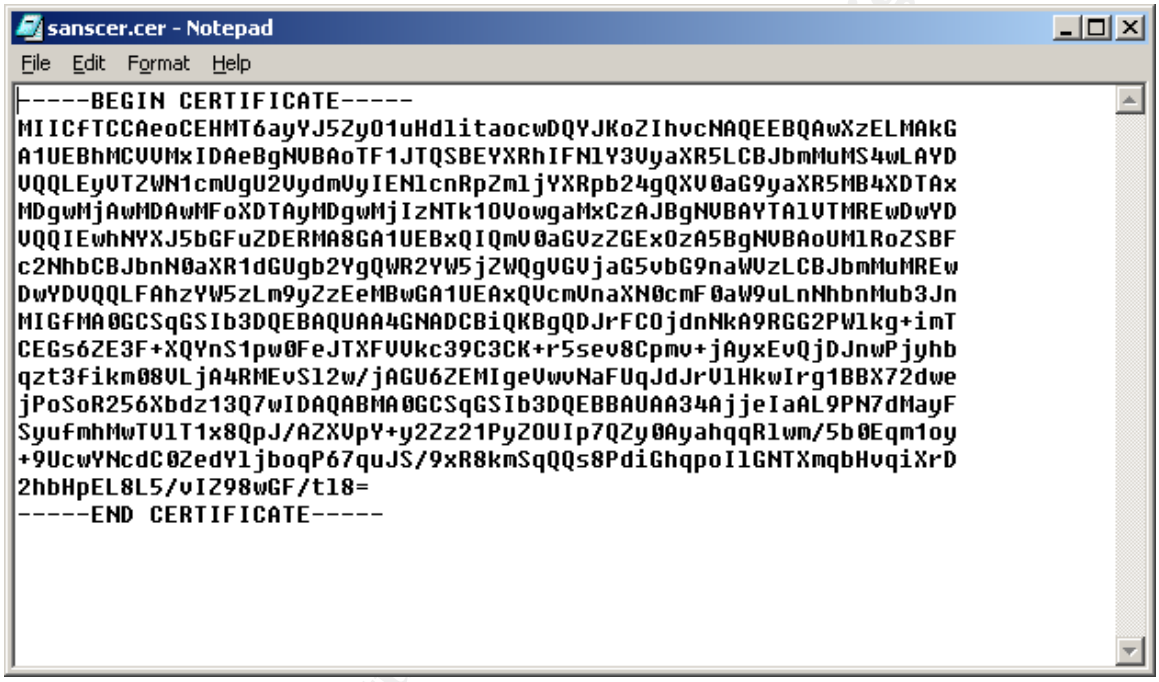

Figure 4. PEM Encoded X.509 Digital Certificate

The creation and management of X.509 digital certificates is a fairly straightforward process. Going back to our non-heterogeneous E-commerce environment, the difficulties begin to arise as you start providing support for various applications that have differing digital certificate requirements. For a lot of applications, particularly web browsers, the format is indifferent as they can transparently load either a PEM or DER file. Yet, my exposure to a wide variety of E-commerce applications has shown that more often than not, these applications will have a preference. The end result is if it isn't in the right format the application will not load it. But so far we have only covered the actual X.509 digital certificate. There is another equally as important aspect, and that is the transport and handling of encryption keys via RSA's Public Key Cryptography Standards.

Public Key Cryptography Standards
Public Key Cryptography Standards (PKCS) are specifications created by RSA Laboratories in conjunction with a consortium of vendors consisting of Apple, DEC, Microsoft, and Sun to name a few. The functions of the PKCS specifications vary and are written to address numerous issues as related to security and cryptography.

For instance, the certificate-signing request standard is known as a PKCS#10. This specification defines the syntax for public key certificates that are submitted to a certificate authority for signing. Another example is the PKCS#8 specification, which deals with managing private keys and protecting them with a password. Table 1 contains a list from http://www.rsalabs.com/pkcs of currently published PKCS specifications and those under development. Selecting the respective PKCS file will take you to RSA's web site where you can download the specification in MS-Word, Acrobat PDF, or PostScript.

PKCS#	RSA Description
PKCS #1	Provides specification for encrypting and signing data using the RSA public-key cryptosystem.
PKCS#1	Specification for Diffie-Hellman key agreement protocol.
PKCS#5	Defines the specifications for encrypting data with a secret key derived from a password.
PKCS#6	Is being replaced by version 3 of X.509.
PKCS#7	Defines a specification for messages that include cryptographic features such as digital signatures and encryption.
PKCS#8	Defines a format for private key information.
PKCS#9	Defines attribute types for use in the other PKCS standards.
PKCS#10	Defines specification for certification requests that are submitted to a CA.
PKCS#11	Defines a programming interface called Cryptoki for cryptographic devices such as smart cards and PCMCIA cards.
PKCS#12	Defines a format for storing or transporting a user's private keys and certificates.
PKCS#13	Is intended to define mechanisms for encrypting and signing data using Elliptic Curve Cryptography.
PKCS#14	Currently under development and discusses pseudo-random number generation.
PKCS#15	A complement to PKCS #11 providing a standard for the format of cryptographic credentials stored on cryptographic tokens.

Table 1. PKCS Specifications (RSA)

Reading the specifications though can be somewhat daunting, and it may not always be clear as to what they are supposed to accomplish. Based upon first hand experience with the implementation of a PKI system and through the interaction with various clients in a non-heterogeneous E-commerce

environment, Table 2 highlights some of the most common file formats I have encountered and how they were used:

File Format	Usage
PKCS#1	Private key. Some key generating tools will either create a PKCS#1or PKCS#8 private key file. For instance, a tool included with WebMethods generates a PKCS#1.
PKCS#7b	Internet Explorer uses this to export and packages digital certificate along with CA signing root certificate. A lot of applications are starting to use this as a means to import a certificate and its trusted root.
PKCS#8	Password protected private key storage. Preferred format of RSA key generation tool.
PKCS#10	Certificate signing request (CSR) used to submit a certificate to the certificate authority for signing. It is in a PEM format.
PKCS#12	Password protected file format that combines both a private key and digital certificate into a single file. Also labeled as .pfx within Internet Explorer.

Table 2. PKCS Usage

An example of a PKCS#7 file as displayed within Windows 2000 is provided in Figure 5. As of lately, I have noticed that more applications are providing support for this file format so that they can easily load in a certificate and it's trusting root certificate. This becomes critical when a PKI is using self-signed certificates and the application needs to add the self-signed root to it list of trusted certificate authorities. As an aside, Internet Explorer and Netscape Web browsers come preloaded with several trusted root certificates. This allows them to recognize certificates signed by the more popular certificate authorities.

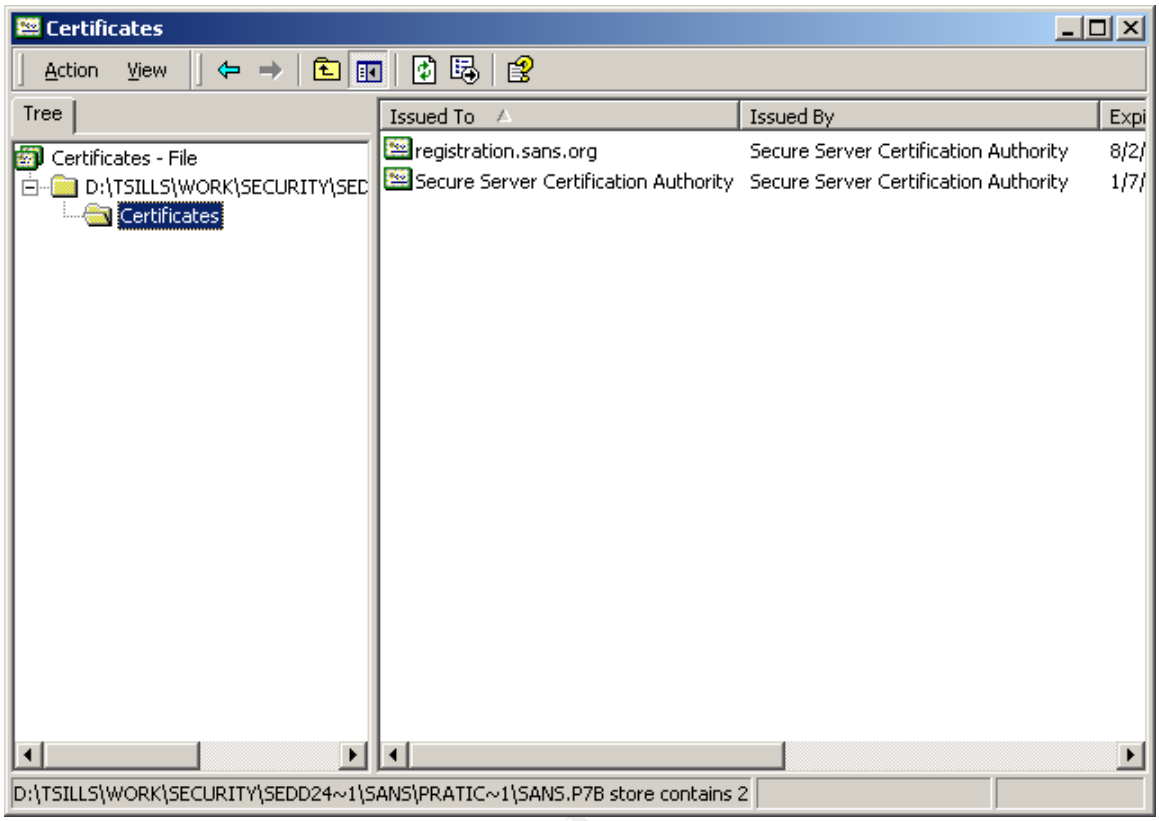

Figure 5. Windows 2000 Display of PKCS#7b

So we've identified the two formats for the X.509 digital certificate as being DER or PEM based file. We have also seen that the encryption keys along with the digital certificates can be manipulated and packaged within various Public Key Cryptography Standards specifications. By understanding these formats and standards, it certainly seems as if it all should be manageable. Well, it would be only if there was some consistency between applications when loading in digital certificates and key pairs. In the next section, the topic of implementation issues and some learned lessons will be discussed.

Implementation Issues

The issue is twofold. For the developer, there are various options and approaches available when loading into their application a digital certificate. The results will often times be inconsistent file type requests from the CA. One client may wish to simply have a binary encoded digital certificate and root certificate supplied separately. Based upon another client's software, it may be required to package the certificates within a PKCS#7b formatted file.

The other facet to this complex equation is the interchangeable usage of extensions and file types. As previously discussed, a PKCS#12 (.p12) file can contain a user's certificate as well as the associated private key. But this same

.p12 file can be internally formatted differently thus making it unreadable by some applications.

In particular, the inability to read certain .p12 files is a known issue for applications that utilize Sun's JSSE 1.0.2 API for the loading of digital certificates. As discussed in Sun's Java Secure Socket Extension (JSSE) 1.0.2 API User's Guide:

> *The 1.0.2 release of JSSE has limitations on its implementation but can read and use PKCS12 keystore files exported by Netscape Navigator. Future releases will also support and be tested with Internet Explorer and other applications.*

This problem was one of the tougher ones to solve as the application attempting the certificate load made it appear there was an integrity problem with the file. In reality it was in the wrong different format.

Other issues may arise during the creation of the certificate-signing request. Some application's keystores do not allow the importation of a private key. This is not an issue as long as the submitted certificate-signing request can be processed. On occasion though when processing a CSR, an error message may arise noting that the file is a malformed PKCS#10. Past experience has shown that this is sometimes due to incorrect data or unsupported characters entered into the fields that are used for the CSR creation such as the host name, locality, state, and country.

Troubleshooting SSL and digital certificate issues is very difficult because the nature of the subject alone is security, which means access to information is limited. The authentication and encryption of data is usually done in a black box and is hidden from the user. To remedy this problem, the next section will provide some ideas as to how you can build a toolbox for the management of X.509 digital certificates and key pairs.

Building a Security Toolbox

The subject of security is far reaching at times and can also be very difficult to manage considering it touches so many facets of an organization. My recommendation is to build a security toolbox of applications and procedures. Table 3 provides some insight into my common practices and tools utilized.

Create a Knowledgebase	Keep track of the various applications you encounter and note their specific file preferences.
Standard Operating Procedures	Develop operating procedures to automate tasks of creating, manipulating, and testing files.

Be Aware	Understand the SSL process and function of certificates and key pairs. This will make troubleshooting much easier.
Common Tools	Utilize Internet Explorer and Netscape Communicator's ability to convert files from PEM to DER and vice versa. Figure 6 is an example of Internet Explorer's capabilities.
Advanced Tools	Acquire applications such as from Baltimore Technologies to create PKCS#12 files or to do file conversations to other PKCS specifications.
Establish a Test Environment	Implement a test server to facilitate troubleshooting. In particular, an Apache web server can provide extensively detailed logs of the SSL handshake.

Table 3. Security Toolbox

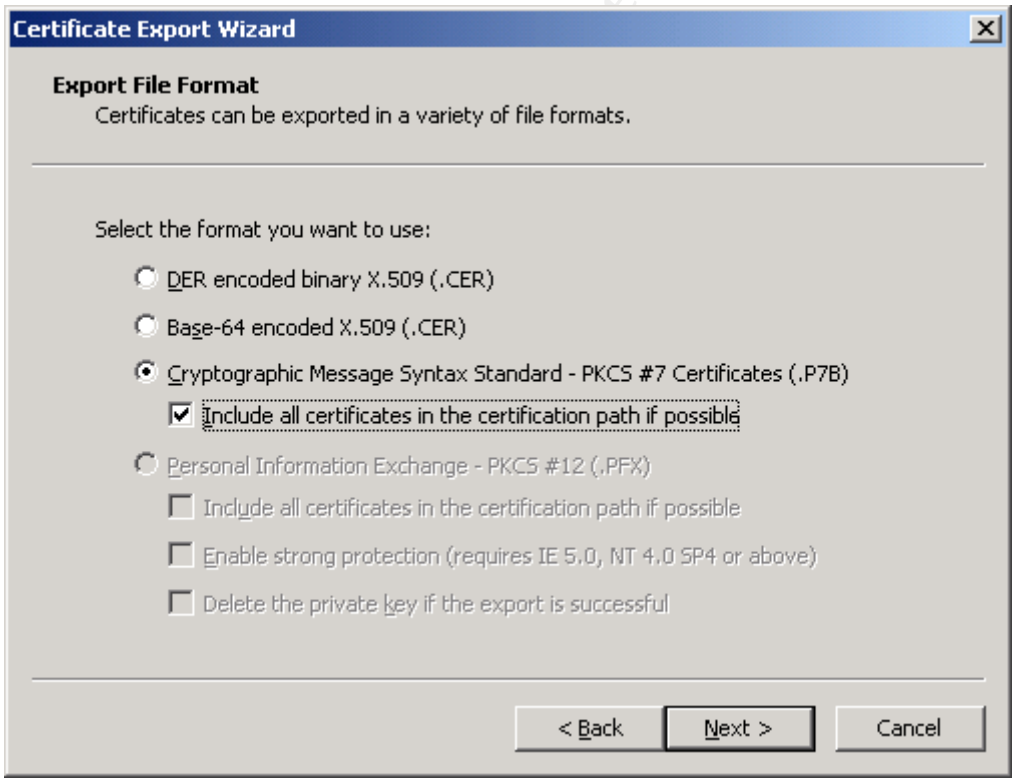

Figure 6. File Conversion in Internet Explorer

One would think that the tools for creating and manipulating files would be the most critical aspect to your toolbox. In reality though, it will be the processes and procedures. This is because it will become imperative that all digital certificates look and act the same as you deploy them. Sure, they will be individually unique,

but collectively you have to rely on the fact that they will all be generated, manipulated, and distributed the same. Otherwise reliability will suffer and you'll spend more time troubleshooting your own system let alone a client's.

Conclusion

The continued demand for increased security will drive the need for some form of establishing authentication and encryption. Right now public key infrastructure is it. Even with the support and implementation issues along with a growing concern of instability with PKI vendors, there still doesn't seem to be an equivalent. Sure there's Pretty Good Privacy (PGP) and Kerberos, but these systems have their own issues and are not as widely accepted just yet.

With this in mind, the best a consultant or administrator can do is arm themselves with a strong understanding of the underlying mechanics of digital certificates and private/public keys. Acquaint themselves with the various file formats. Obtain numerous tools to create and manipulate keys and X.509 digital certificates. And finally strive to understand the client applications and how these applications implement security. Knowing what to expect will ease the troubleshooting and process.

References:

ASN.1 Information Site
URL:http://asn1.elibel.tm.fr/en/index.htm (24 August 2001)

Bobbitt, Mike. "PKI Policy Pitfalls". Information Security. July 2001: 68-79

Green, Roedy. "Digital Certificates"
URL:http://mindprod.com/gloss.html (20 August 2001)

Hirsch, Frederick. "Introducing SSL and Certificates using SSLeay"
URL:http://www.ultranet.com/~fhirsch/Papers/wwwj/article.html (24 August 2001)

Housley, R. Ford, W. Polk. Solo, D. "RFC: 2459-Internet X.509 Public Key Infrastructure Certificate and CRL Profile"
URL:http://www.ietf.org (20 August 2001)

Housley, Russ. Polk, Tim. Planning for PKI. New York: John Wiley & Sons, 2001.

IPlanet. "Netscape Certificate Management System (CMS) Installation and Deployment Guide"
URL:http://docs.iplanet.com/docs/manuals/cms/41/dep_gide/contents.htm (21 August 2001)

Netscape Communications Corp. "Introduction to SSL"
URL:http://developer.netscape.com/docs/manuals/security/sslin/index.htm (22 August 2001)

RSA Security. “RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1”
URL:http://www.rsasecurity.com/rsalabs/faq/ (24 August 2001)

RSA Security. “Public-Key Cryptography Standards”
URL:http://www.rsasecurity.com/rsalabs/pkcs (24 August 2001)

Sun. “Java™ Secure Socket Extension (JSSE) 1.0.2 API User's Guide”
URL:http://java.sun.com/products/jsse/doc/guide/API_users_guide.html#Troubleshooting
(24 August 2001)

Tremblett, Paul. “X.509 Certificates-Moving Toward Secure Communication”. Dr. Dobb’s Journal July 1999
URL:http://www.ddj.com/articles/1999/9907/9907c/9907c.htm (21 August 2001)

Last Updated: September 25th, 2020

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS October Singapore 2020	**Singapore, SG**	**Oct 12, 2020 - Oct 24, 2020**	**Live Event**
SANS Community CTF	**,**	**Oct 15, 2020 - Oct 16, 2020**	**Self Paced**
SANS SEC504 Rennes 2020 (In French)	**Rennes, FR**	**Oct 19, 2020 - Oct 24, 2020**	**Live Event**
SANS SEC560 Lille 2020 (In French)	**Lille, FR**	**Oct 26, 2020 - Oct 31, 2020**	**Live Event**
SANS Tel Aviv November 2020	**Tel Aviv, IL**	**Nov 01, 2020 - Nov 06, 2020**	**Live Event**
SANS Sydney 2020	**Sydney, AU**	**Nov 02, 2020 - Nov 14, 2020**	**Live Event**
SANS Secure Thailand	**Bangkok, TH**	**Nov 09, 2020 - Nov 14, 2020**	**Live Event**
APAC ICS Summit & Training 2020	**Singapore, SG**	**Nov 13, 2020 - Nov 21, 2020**	**Live Event**
SANS FOR508 Rome 2020 (in Italian)	**Rome, IT**	**Nov 16, 2020 - Nov 21, 2020**	**Live Event**
SANS Community CTF	**,**	**Nov 19, 2020 - Nov 20, 2020**	**Self Paced**
SANS Local: Oslo November 2020	**Oslo, NO**	**Nov 23, 2020 - Nov 28, 2020**	**Live Event**
SANS Wellington 2020	**Wellington, NZ**	**Nov 30, 2020 - Dec 12, 2020**	**Live Event**
SANS OnDemand	**OnlineUS**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**