Web Application Security

Introduction

SRA University

Spring 2011

Luis Santos and Jaswant Singh

Agenda

- Introductions
- Security – So What?
- Course Objectives
- Sessions
- Do’s and Don’ts
- Q & A

Luis Santos, Tech Lead

- Over 20 years of experience in full life cycle software development
- Areas of Expertise
  - Agile Development
  - Application Security
  - Implementing emerging technologies
  - Web Applications and SOA Web Services
  - Mentoring, Enterprise Architecture and Software Architecture
- Education
  - BS, Computer Science Federal University of Rio de Janeiro, Brazil
- Contact Info
  - Tel: 703.677.8274
  - Email: Luis_Santos@sra.com Aol IM: luissantos1
  - LinkedIn: http://www.linkedin.com/in/luissantos1

Jaswant Singh, Chief Architect

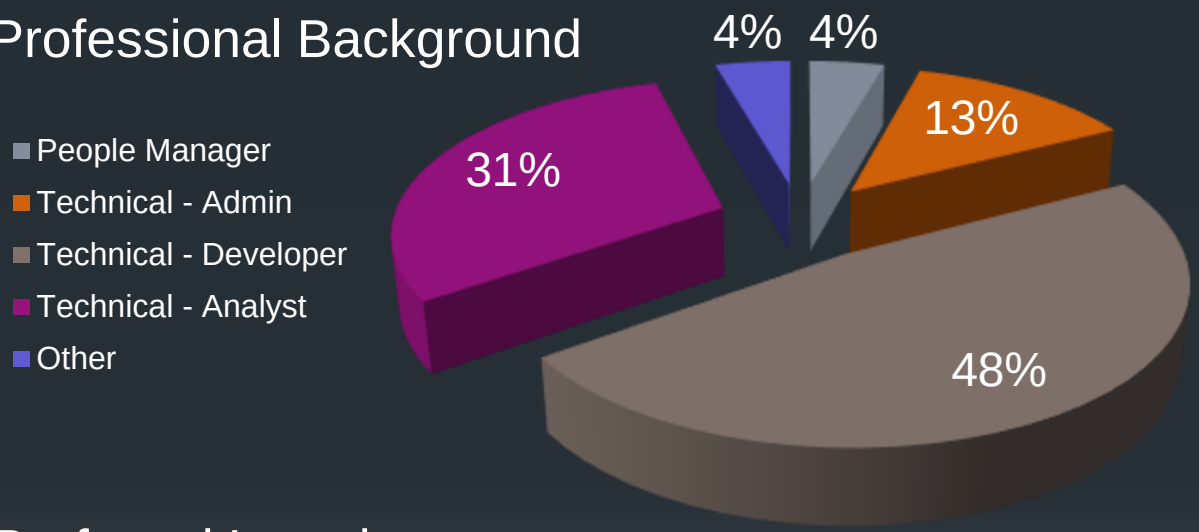
- Over fifteen years of leadership experience in conceptualizing, directing, and implementing advanced technology solutions
- Areas of Expertise
  - SOA Web Services and Internet Applications System Architecture and Engineering
  - Enterprise Architecture; Research and Development; Establishing Best Practices
  - Mentoring; Building and Leveraging Cross Functional Working Relationships.
  - Cloud / Utility Computing; Data Center Optimization and Consolidation
  - Data Analysis; Workflow Automation
- Education
  - MS Technology Management, George Mason University
  - MS Mathematics, Indian Institute of Technology Bombay
- Contact Info
  - Tel: 703.662.1097
  - Email: Jaswant_Singh@sra.com Aol IM: bagheljas
  - LinkedIn: http://www.linkedin.com/in/jaswantsingh

Participant Introduction(s)

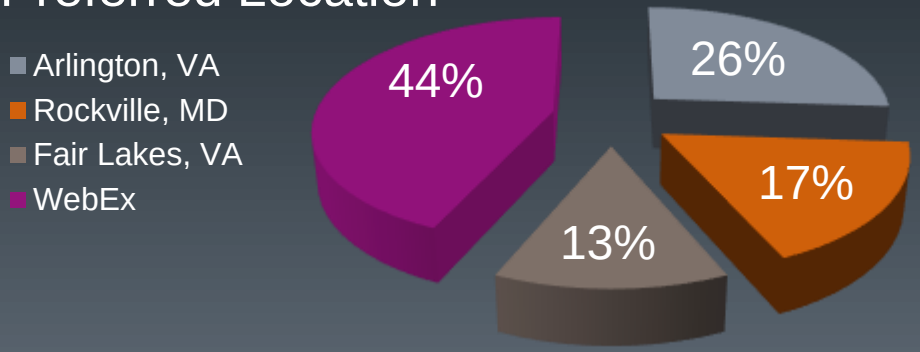
- 60 sec “commercials”

Survey Results

Professional Background

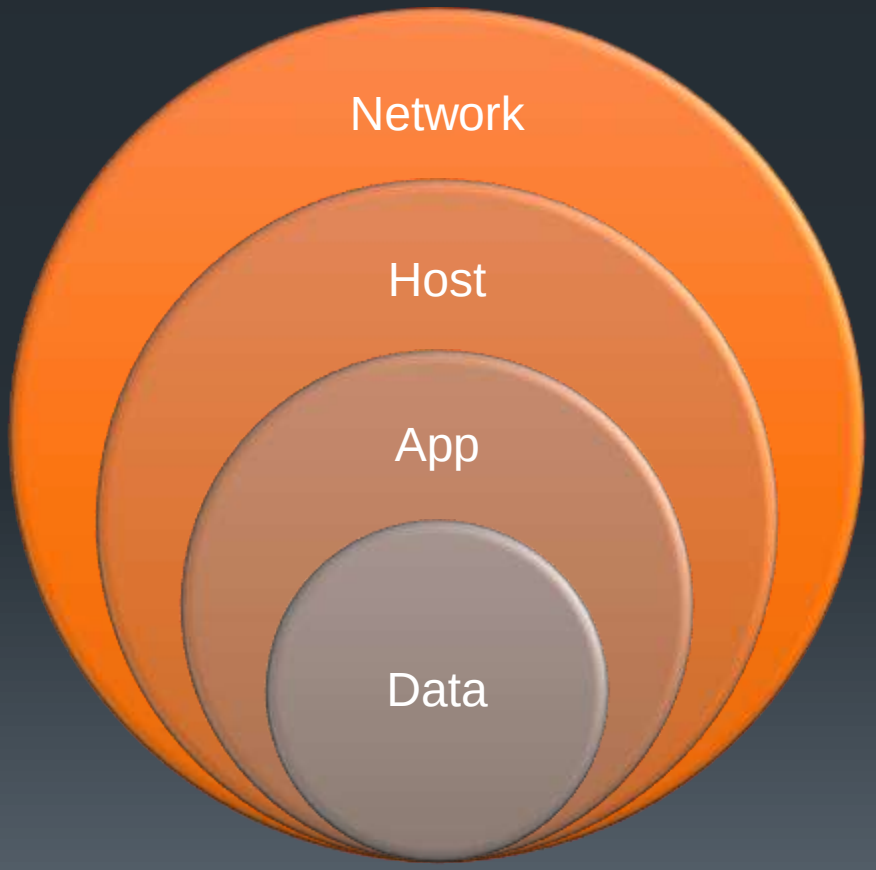


Preferred Location



Security - So What?

- Confidentiality
- Integrity
- Availability
- Authenticity
- Non-repudiation



Course Objectives

- Develop understanding of cryptography
- Understand cryptography role in securing web applications
- Networking with like minded people

Sessions

- Cryptography Overview
- Public Key Infrastructure (PKI)
- Application Server Encryption
- Web Services
- Encrypted File System
- Encrypted Database

Session Breakdown

- Lecture - Part One
- Q & A
- Bio Break - 10 min
- Lecture - Part Two
- Q & A

Do’s and Don’ts

- Please mute your phone line
- If you have question raise your hand on WebEx
- Please type your question in WebEx
- For attendance, we rely on your presence in
  - Classroom
  - WebEx
- Slides will be mailed after the session

Q & A

You may also send offline questions to us via email

Luis_Santos@sra.com
Jaswant_Singh@sra.com

Web Application Security

Cryptography Overview

SRA University

Spring 2011

Luis Santos and Jaswant Singh

Agenda

- Buzzwords
- Cipher Evolution
- Key Exchange and PKI
- Digital Certificates
- RSA/DSA/3DES
- Cryptographic Protocols
- Security Compliance Standards
- Q & A

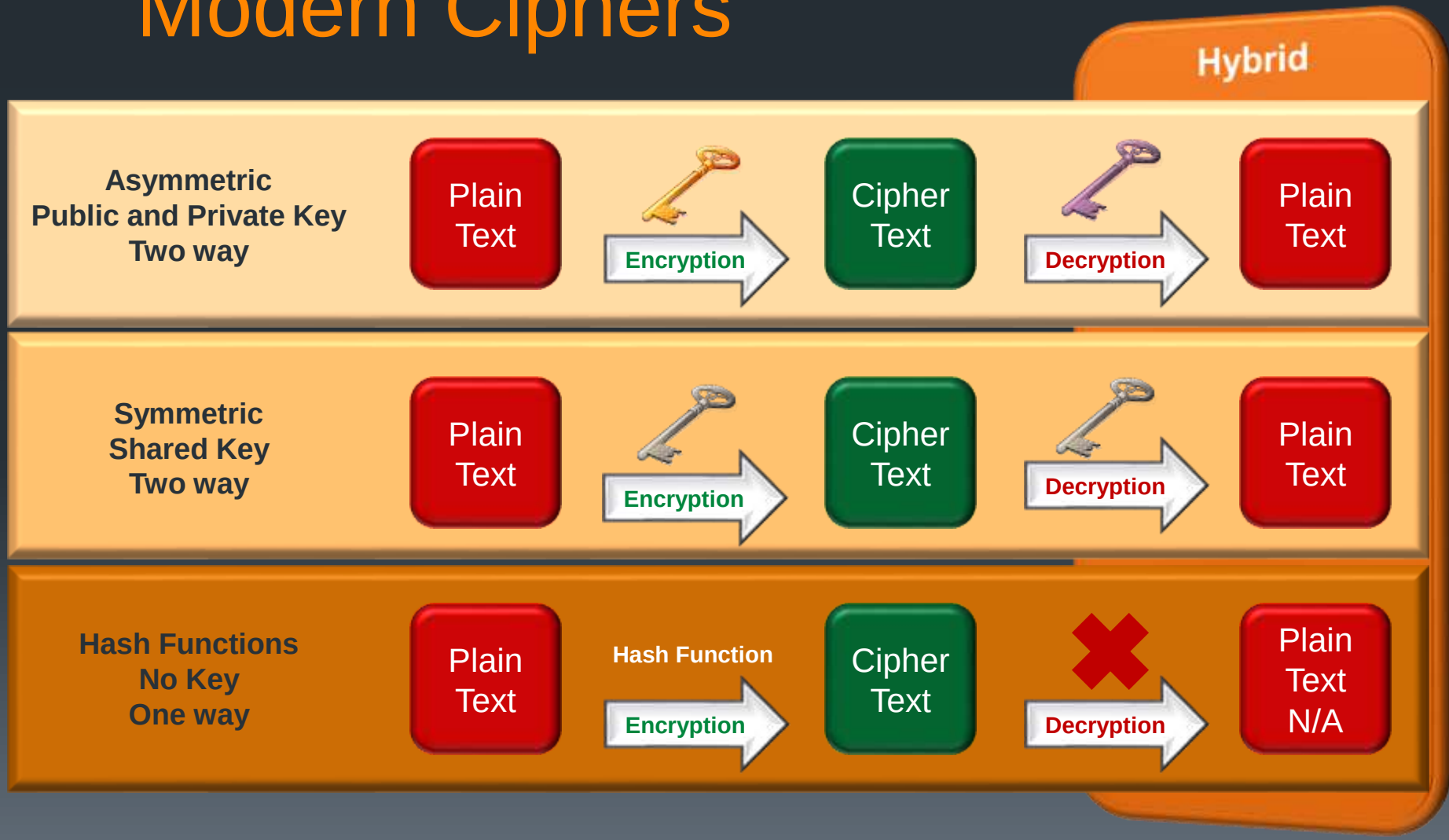
Buzzwords

- Plaintext
- Ciphertext
- Cipher
- Digital Certificates
- NIST
- NSA
- FISMA
- FIPS
- DIACAP
- PCI DSS
- C & A
- Hash function
- Message
- Digest
- Ephemeral key
- Static key
- Brute force
- CA
- CSR
- Trust Store
- PKI
- PKCS
- PEM vs. DER

Cipher Evolution

- Classical – Up to 19th Century
  - Substitution
  - Transposition
- Rotor Machines – Early 20th Century
  - Substitution
    - Additive
    - Polyalphabetic
  - Transposition
- Modern
  - Public and Private Key (Asymmetric)
    - RSA
    - DSA
  - Shared Private Key (Symmetric)
    - Data Encryption Standard (DES) -1978
    - Advanced Encryption Standard (AES) -1995
    - Stream
    - Block
  - Hash Functions (One-way)
  - Hybrid ?

Modern Ciphers



Key Exchange

- Whitfield Diffie & Martin Hellman (1976)
- Facilitates keys exchange over secure channels
- Named as Diffie-Hellman Key Exchange
  - Also known as D-H Key Exchange
- Didn’t address Identity Problem

Public Key Infrastructure (PKI)

- X.509 Standards Defines PKI
- Addressed the Identity Problem
- Foundation for Public-Key Cryptography
- PKI components
  - Public / Private keys
  - Digital Certificates
  - Certificate Authority (CA)
    - AOL Member CA, Equifax, VeriSign etc.

Public Key Cryptography Standards

- Published by RSA Security since 1991
- PKCS # 1 to PKCS #15
- SSL Digital Certificates
- PKCS # 7 Cryptographic Message Syntax Standard (PKI) v1.5
- PKCS #10 Certification Request Standard (CSR) v1.7
- PKCS #12 Personal Information Exchange Standard (PFX) v1.0
- Common filename extensions for X.509-certificates:
  - .der - DER encoded certificate
  - .pem - Base64 encoded DER certificate
  - .cer, .crt - Same as either .der or .pem form
  - .pfx, .p12 - PKCS#12 password protected certificate(s) & private key(s)
  - .p7b, .p7c - PKCS #7 SignedData structure with only certificate(s)

Digital Certificates

- Simply known as Certificates
- A Certificate Identifies Someone
- Contents of a Certificate
  - Identity Information
  - CA (Certificate Authority) Information
  - Public Key and Key Usage
  - Validity Period
  - Thumbprint - Algorithm and Hash

Key lengths and Cipher Modes

- Factors Deciding Key Lengths
  - Performance
  - Security from Brute Force
- Suggested Key Lengths

Cipher Modes	Few Years	Over Ten Years
Asymmetric	1024	2048
Symmetric	128	256

- Why 8 times key lengths for Asymmetric vs. Symmetric ?

RSA / DSA / 3DES

	RSA	DSA	3DES
Cipher Modes	Asymmetric	Asymmetric	Symmetric Block
Speed of Enc & Dec	Slow	Slow	Fast
Keys	Pair of P & P	Pair of P & P	3 Private Keys (56bits)
Full Name	Last Names initials of Ron Rivest, Adi Shamir, and Leonard Adleman	Digital Signature Algorithm	Triple Data Encryption Standard
Founded	1977	1991	1998
Organization	MIT	NIST	NIST
Known Security Threat	RSA Problem Integer Factorization Brute Force	Brute Force	Brute Force Keying Options
Sign/Verify Cost	1/1	1/10	N/A

Cryptographic Protocols

- Internet Protocol Security (IPSec)
- Secure Socket Layer/Transport Layer Security (SSL/TLS)
  - Provides the authentication of server identity
  - Provides the optional authentication of client identities
  - Encrypts the data for intended recipients
- OpenPGP, EFS and S/MIME
- Secure Shell (SSH)
- Kerberos

Security Compliance Standards

- Certification and Accreditation (C & A)
- Federal Information Security Management Act (FISMA)
- Federal Information Processing Standards (FIPS)
- Defense Information Assurance Certification and Accreditation Process (DIACAP)
- Health Information Portability and Accountability Act (HIPAA)
- Payment Card Industry Data Security Standard (PCI DSS)

Q & A

You may also send offline questions to us via email

Luis_Santos@sra.com
Jaswant_Singh@sra.com

Web Application Security

Public Key Infrastructure

SRA University

Spring 2011

Luis Santos and Jaswant Singh

Agenda

- Buzzwords
- Why and What is PKI? How ?
- Public Key Cryptography Standards
- Digital Signatures and Certificates
- Certificate File Formats
- Certificate Management Tools
- Reference Architectures
- Best Practices
- Q & A

Buzzwords

- PKI
- PKCS
- Intermediate CA
- Root CA
- Digital Signatures
- Digital Certificates
- OCSP
- Thumbprint
- CSR
- CA
- DER vs. PEM
- PFX
- JKS
- Java Keytool
- Open SSL
- CRLs

Why PKI?

- Need a solution for identity validation and key exchange
- Need for a platform that facilitates transmission of secured data over insecure networks like the Internet
- Need assurance of the quality of information sent and received

What is PKI?

- PKI stands for “Private Key Infrastructure”
- PKI provides the capability for identity validation and key exchange
- PKI allows the use of a mathematical technique called Public Key Cryptography in a community environment
- PKI facilitates secure communication among parties that do not have a previously established relationship

How it Works

- PKI uses a pair of mathematically related keys. One key is used to encrypt information, and the related key can decrypt that information.
- If a party knows one of the keys, it cannot easily calculate what the other one is.
- The Keys:
  - A “public” key. All recipients of the encrypted data have access to this key. It is used to encrypt the data.
  - A corresponding (and unique) “private” key. It is used to decrypt the data.
- Your private key enables you to prove, unequivocally, that you are who you claim to be.

Public Key Cryptography Standards

- First Published in 1991
- Devised and controlled by RSA Data Security
- Cryptographic standards serve two important goals:
  - make different implementations interoperable
  - avoid known pitfalls
- These standards cover RSA encryption, RSA signature, password-based encryption, cryptographic message syntax, private-key information syntax, selected object classes and attribute types, certification request syntax, cryptographic token interface, personal information exchange syntax, and cryptographic token information syntax.

Public Key Cryptography Standards - 1

	Version	Name	Comments
PKCS #1	2.1	RSA Cryptography Standard	See RFC 3447. Defines the mathematical properties and format of RSA public and private keys, and the basic algorithms and encoding/padding schemes for performing RSA encryption, decryption, and producing and verifying signatures.
PKCS #2	-	Withdrawn	
PKCS #3	1.4	Diffie–Hellman Key Agreement Standard	A cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel.
PKCS #4	-	Withdrawn	
PKCS #5	2.0	Password-based Encryption Standard	See RFC 2898 and PBKDF2
PKCS #6	1.5	Extended-Certificate Syntax Standard	Defines extensions to the old v1 X.509 certificate specification

Public Key Cryptography Standards - 2

	Version	Name	Comments
PKCS #7	1.5	Cryptographic Message Syntax Standard	See RFC 2315. Used to sign and/or encrypt messages under a PKI. Used also for certificate dissemination. Often used for single sign-on.
PKCS #8	1.2	Private-Key Information Syntax Standard	See RFC 5208. Used to carry private certificate keypairs (encrypted or unencrypted).
PKCS #9	2.0	Selected Attribute Types	See RFC 2985. Defines selected attribute types for use in PKCS #6 extended certificates, PKCS #7 digitally signed messages, PKCS #8 private-key information, and PKCS #10 certificate-signing requests.
PKCS #10	1.7	Certification Request Standard	See RFC 2986. Format of messages sent to a certification authority to request certification of a public key.
PKCS #11	2.20	Cryptographic Token Interface	Also known as "Cryptoki". An API defining a generic interface to cryptographic tokens (see also Hardware Security Module). Often used in single sign-on, Public-key cryptography and disk encryption systems.

Public Key Cryptography Standards - 3

	Version	Name	Comments
PKCS #12	1.0	Personal Information Exchange Syntax Standard	Defines a file format commonly used to store private keys with accompanying public key certificates, protected with a password-based symmetric key. This container format can contain multiple embedded objects, such as multiple certificates. Usually protected/encrypted with a password. Usable as a format for the Java key store and to establish client authentication certificates in Mozilla Firefox. Usable by Apache Tomcat, but not by Apache HTTP Server.
PKCS #13	-	Elliptic Curve Cryptography Standard	Under development
PKCS #14	-	Pseudo-random number generation	Under development
PKCS #15	1.1	Cryptographic Token Information Format Standard	Defines a standard allowing users of cryptographic tokens to identify themselves to applications, independent of the application's Cryptoki implementation (PKCS #11) or other API.

Digital Signatures

- Addresses tampering and impersonation.
- Tamper detection and related authentication techniques rely on a mathematical function called a one-way hash (also called a message digest) with the following characteristics:
  - Fixed Length
  - Altering data changes the message digest
  - Practically one can’t recover message from message digest - that is why it is called "one-way."
- The signing software creates a one-way hash of the data, then uses your private key to encrypt the hash. The encrypted hash, along with other information, such as the hashing algorithm, is known as a digital signature.

Digital Certificates - 1

- Simply known as Certificates
- A Certificate Identifies Someone
- Contents of a Certificate
  - Identity Information
  - CA (Certificate Authority) Information
  - Public Key and Key Usage
  - Validity Period
  - Thumbprint - Algorithm and Hash

Digital Certificates - 2

- Certificates work much the same way as familiar forms of identification.
- Certificate authorities (CAs) are entities that validate identities and issue certificates.
- The certificate issued by the CA binds a particular public key to the name of the entity the certificate identifies (such as the name of an employee or a server).
- Certificates help prevent the use of fake public keys for impersonation. Only the public key certified by the certificate will work with the corresponding private key possessed by the entity identified by the certificate.
- In addition to a public key, a certificate always includes the name of the entity it identifies, an expiration date, the name of the CA that issued the certificate, a serial number, and other information.
- Most importantly, a certificate always includes the digital signature of the issuing CA.

Certificate File Formats

- Personal Information Exchange (PKCS #12)
  - The Personal Information Exchange format (PFX, also called PKCS #12) supports secure storage of certificates, private keys, and all certificates in a certification path.
  - The PKCS #12 format is the only file format that can be used to export a certificate and its private key.
- Cryptographic Message Syntax Standard (PKCS #7)
  - The PKCS #7 format supports storage of certificates and all certificates in the certification path.
- DER-encoded binary X.509
  - The Distinguished Encoding Rules (DER) format supports storage of a single certificate. This format does not support storage of the private key or certification path.
- Base64-encoded X.509
  - The Base64 format supports storage of a single certificate. This format does not support storage of the private key or certification path.
- PEM (Unix World)
  - PEM stands for Privacy Enhanced Email
  - Encoded DER certificate as base64 encoding
  - Easy to cut and paste

Certificate Filename Extensions

- .der - DER encoded certificate
- .pem - Base64 encoded DER certificate
- .cer, .crt – Same as either .der or .pem form
- .pfx, .p12 - PKCS#12 password protected certificate(s) & private key(s)
- .p7b, .p7c – PKCS #7 SignedData structure with only certificate(s)

Certificate Management Tools

- Java Keytool - Oracle
  - http://download.oracle.com/javase/6/docs/technotes/tools/solaris/keytool.html
- Open SSL - Open Source
  - http://www.openssl.org
- IIS Web Server Certificate Wizard – Microsoft
  - http://support.microsoft.com/kb/299875

Open SSL

http://www.openssl.org

Open SSL Commands - 1

- How to find Open SSL details?
  - Version: openssl version –a
  - Dir: openssl version –d
- How do I get a list of the available commands?
  - List of main commands: openssl –help
  - List of sub commands: openssl –pkcs12 –help
- How do I benchmark Open SSL for my host?
  - openssl speed
- How to encrypt and decrypt data to email?
  - 256 bit AES in CBC Mode and Creating output in base64 encoding
    - Encrypt: openssl enc -aes-256-cbc -a -salt -in file.txt -out file.enc
    - Decrypt: openssl enc -d -aes-256-cbc -a -in file.enc

Open SSL Commands - 2

- How to generate Digital Certificate private key?
  - openssl genrsa 1024 > {domain}.key
- How to generate CSR?
  - openssl req -new –key {domain}.key > {domain}.csr
    - Country* (“C”): US
    - State or Province* (“ST”): Virginia
    - Locality* (“L”): Fair Lakes
    - Organization Name* (“O”): SRA International
    - *Optional:* Organizational Unit* (“OU”): Health and Civil Sector
    - Common Name of the server (“CN”): {domain}
    - *Optional:* E-mail Address:
- How to generate Self Signed Certificate ?
  - openssl req -x509 -key {domain}.key -in {domain}.csr > {domain}.crt
  - openssl req -x509 -key {domain}.key -in {domain}.csr –days DAYS > {domain}.crt
- How to create a single PEM file having key and cert ?
  - cat {domain}.key {domain}.crt > {domain}.pem
    - Maintain the order of key and crt file

Open SSL Commands - 3

- How to convert a certificates formats ?
  - From PEM to DER
    - openssl x509 -in {domain}.pem -inform PEM –out {domain}.der -outform DER
  - From DER to PEM
    - openssl x509 -in {domain}.der -inform DER –out {domain}.pem -outform PEM
  - From PEM to PFX (PKCS #12)
    - openssl pkcs12 –export -out {domain}.pfx -in {domain}.pem
  - From PFX (PKCS #12) to PEM
    - openssl pkcs12 -in {domain}.pfx -out {domain}.pem –nodes
- How to remove pass phrase from key?
  - openssl rsa -in key.pem -out newkey.pem
- How to remove pass phrase from key and cert?
  - openssl rsa -in mycert.pem -out newcert.pem
  - openssl x509 -in mycert.pem >>newcert.pem

Open SSL Commands - 4

- How do I verify a certificate?
  - openssl verify {domain}.pem
- How do I extract information from a certificate?
  - openssl x509 -text -in {domain}.crt
  - openssl x509 -text -in {domain}.pem
  - Who issued the cert?
    - openssl x509 -noout -in {domain}.pem -issuer
  - Who was the certificate issued for?
    - openssl x509 -noout -in {domain}.pem -subject
  - What dates is it valid?
    - openssl x509 -noout -in {domain}.pem –dates

Open SSL Commands - 5

- How to test SSLv2 is disabled?
  - openssl s_client –ssl2 -connect {domain}:{port}
  - Web/Application Server Example
    - openssl s_client –ssl2 -connect mail.aol.com:1043
      - CONNECTED(00000003)
      - 29470:error:1407F0E5:SSL routines:SSL2_WRITE:ssl handshake failure:s2_pkt.c:428:
  - Web Application Behind Netscaler Example
    - openssl s_client –ssl2 -connect mail.aol.com:443
      - Type “HEAD /_ns_/nstest.html” and Hit Enter

        HTTP/1.1 200 Ok
        Server: NS_6.1
        Connection: close
        Content Length: 309
        Content-Type: text/html
        <html><body>
        SSL Protocol Alert The SSL protocol version that your browser uses is SSLv2 and it is not compatible with the server settings. Please try the following: - Check the SSL protocol settings on your browser for SSLv3/TLSv1 protocol support and enable the same.
        </body></html>
        Closed
      - If HEAD /_ns_/nstest.html doesn’t work for you than replace /_ns_/nstest.html with the URI configured for L7 health checks.

Open SSL - Unix Shell Script

```
#!/bin/sh
# Authors: Jaswant Singh (Jas) & Luis Santos @ SRA International
DOMAIN=$1
#Parameters for your environments and organization and Do not remove  empty lines between <<EOF to EOF
COUNTRY=US
STATE=Virgnia
CITY="FAIR LAKES"
ORGNAME="SRA International"
if [ "$DOMAIN" = "" ] ; then
        echo "You need to define DOMAIN to create key and csr"
        echo "Usage: ./csr.sh DOMAIN"
        echo "For Example:"
        echo "Usage: ./csr.sh jas.sra.com"
else
keyFile=${DOMAIN}.key
openssl genrsa 2048 > ${keyFile} 2>/dev/null
echo ""
echo "Successfully created domain ${DOMAIN} key ${keyfile}"
echo ""
csrFile=${DOMAIN}.csr
openssl req -new -key ${keyFile} > ${csrFile} 2>/dev/null <<EOF
`echo ${COUNTRY}`
`echo ${STATE}`
`echo ${CITY}`
`echo ${ORGNAME}`

`echo ${DOMAIN}`



EOF
echo "The cert key and csr are ready for domain ${DOMAIN}"
echo ""
fi
```

Java Keytool

http://download.oracle.com/javase/6/docs/technotes/tools/solaris/keytool.html

Java Keytool Commands

- JDK JKS key stores default password: changeit
- Creating New Key
  - keytool -genkey -keysize 1024 -keyalg RSA -keystore ServerIdentity.jks -alias **{domain}**
  - keytool -genkey -keysize 1024 -keyalg RSA –keystore ServerIdentity.jks -validity **DAYS** -alias **{domain}**
- Listing Keys in KeyStore
  - keytool -list -keystore **ServerIdentity.jks**
- Deleting a Key
  - keytool -delete -keystore **ServerIdentity.jks** -alias **{domain}**
- Creating CSR
  - keytool -certreq -alias **{domain}** -file **{domain}**.csr -keystore **ServerIdentity.jks**
- Importing Cert
  - keytool -import -alias **{domain}** -file **{domain}**.der -keystore **ServerIdentity.jks**
- Exporting Cert
  - keytool -export -alias **{domain}** -file **{domain}** .der -keystore **ServerIdentity.jks**
- Trusted CA
  - keytool -import -alias **aliasfortrustedca** -trustcacerts -file **trustedcafilename.der** -keystore **cacerts**

Extracting Private Key - 1

- Copy the Java Code and Save in a file name as "DumpPrivateKey.java"

```
import java.io.FileInputStream;
import java.security.Key;
import java.security.KeyStore;
import sun.misc.BASE64Encoder;

public class DumpPrivateKey {
static public void main(String[] args) throws Exception {
        if ( args.length < 3 || args.length > 4) {
System.out.println("Usage: java DumpPrivateKey <keysotre filename> <keystore password> <key alias>");
                        System.exit( 1);
        }
        KeyStore ks = KeyStore.getInstance("jks");
        ks.load(new FileInputStream(args[0]), args[1].toCharArray());
        Key key = ks.getKey(args[2], args[1].toCharArray());
        String b64 = new BASE64Encoder().encode(key.getEncoded());
        System.out.println("-----BEGIN PRIVATE KEY-----");
        System.out.println(b64);
        System.out.println("-----END PRIVATE KEY-----");
    }
}
```

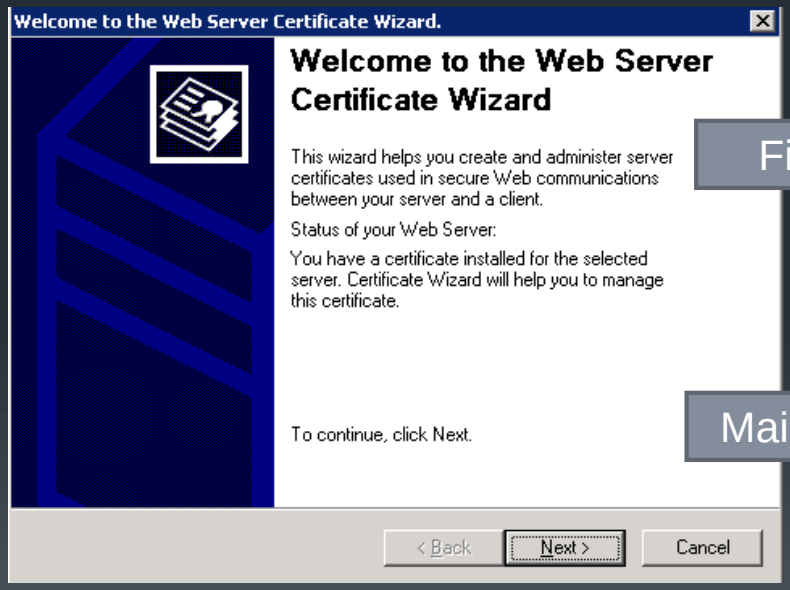
Extracting Private Key - 2

- Compile the DumpPrivateKey.java
  - javac DumpPrivateKey.java
- Run DumpPrivateKey.class
  - java DumpPrivateKey <keysotre filename> <keystore password> <key alias>
  - java DumpPrivateKey mykeystore.jks mypassword myalias
  - java DumpPrivateKey mykeystore.jks mypassword myalias > myalias.key
- Please note that it makes the assumption that you have java installed on the your box and java and javac in your path.
- To make it usable with Netscaler or Apache You will use OpenSSL to remove the passphrase from the private key
  - openssl rsa -in myalias.key –out newmyalias.key

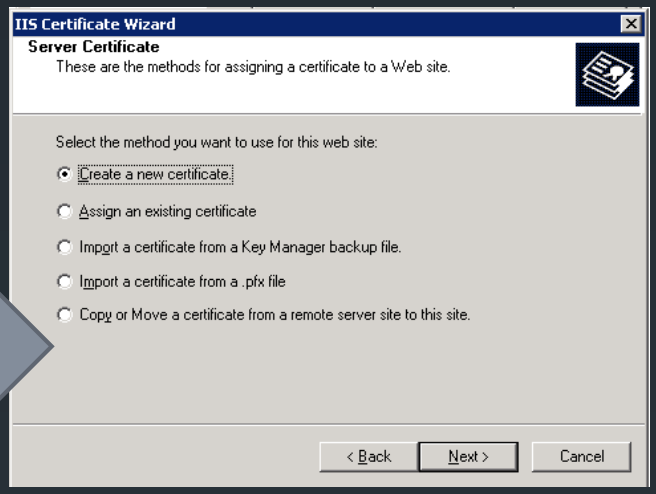
IIS Web Server Certificate Wizard

Visit Microsoft Knowledge Base
http://support.microsoft.com/kb/299875

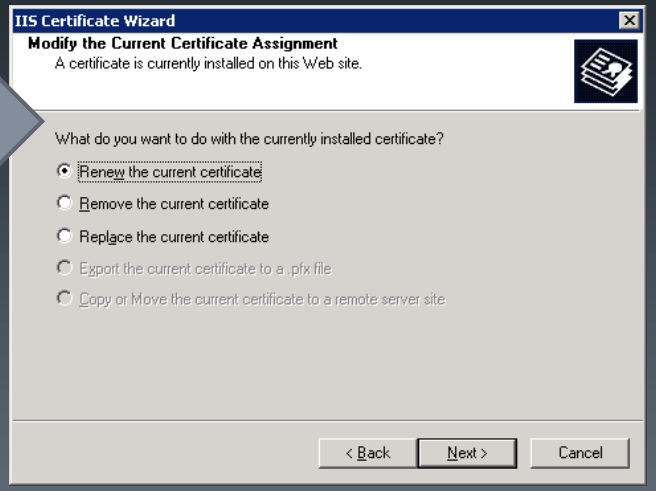
Server Certificate Wizard



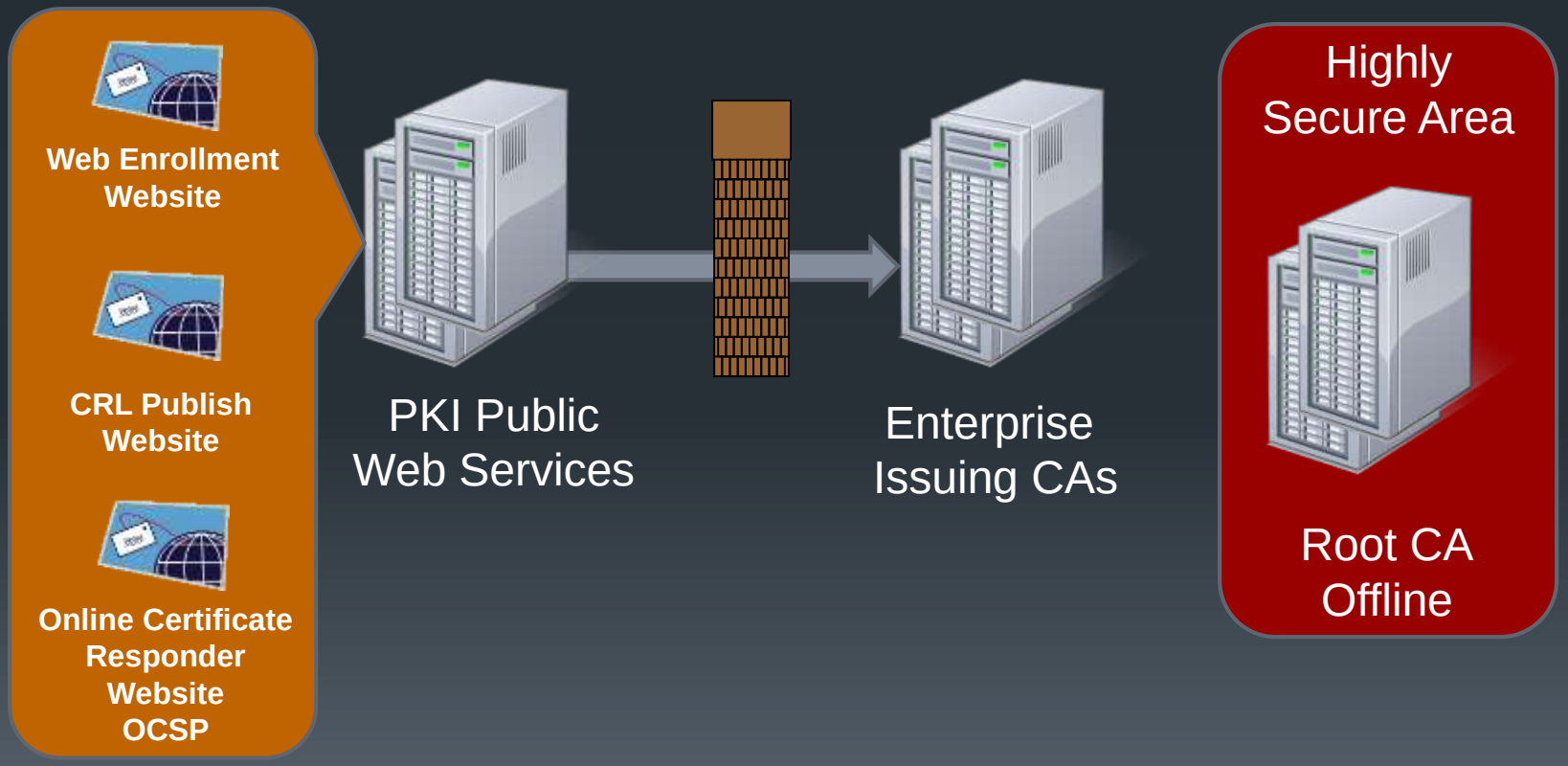
First Time



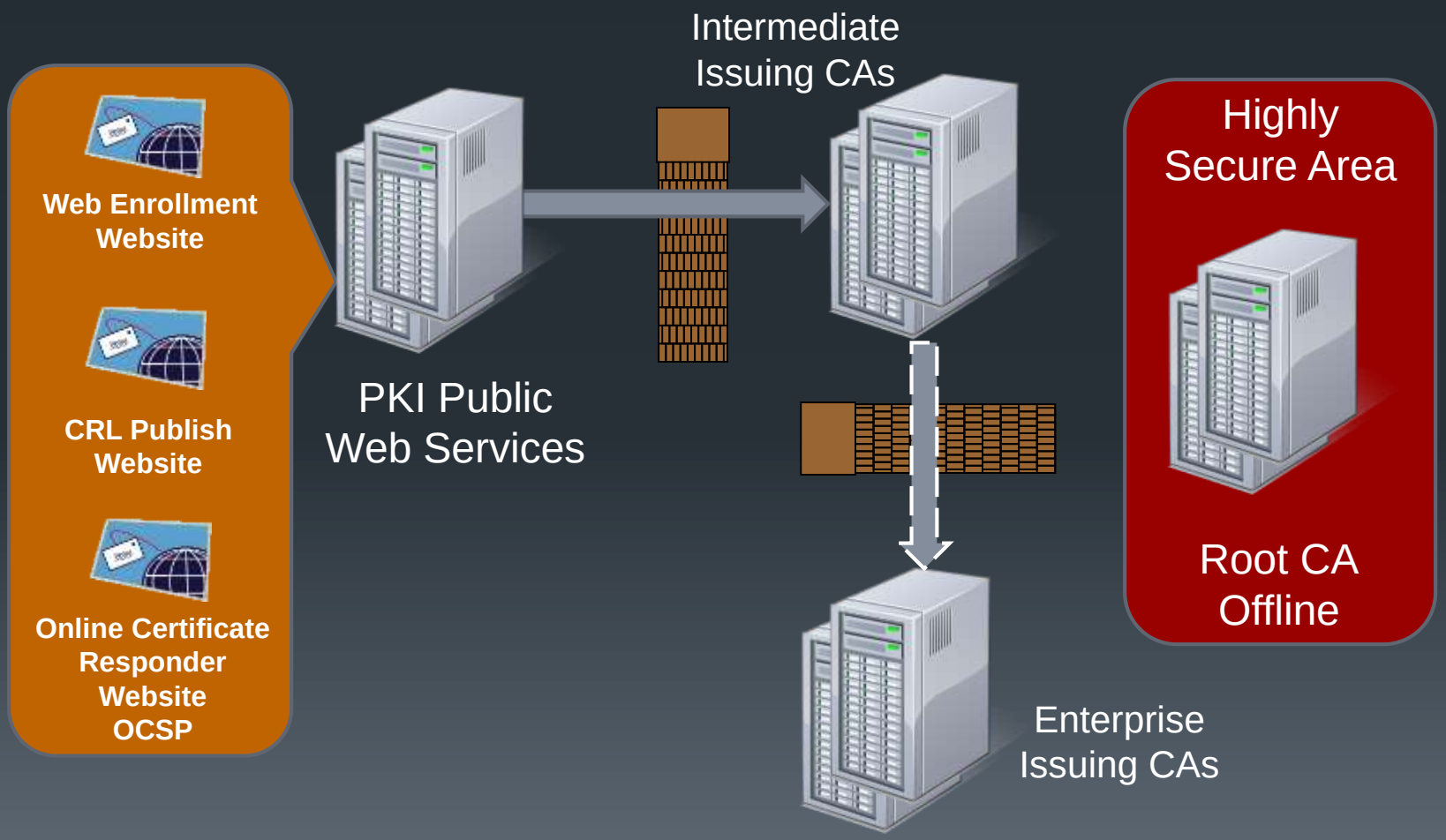
Maintenance



PKI Reference Architecture – 1



PKI Reference Architecture - 2



Sample Hierarchy and Templates

- Hierarchy
  - Root CA Server
    - Status: Offline Key Size: 4096 bits Cert Lifetime: 10 years
  - Enterprise Issuing CA Server
    - Status: Online Key Size: 2048 bits Cert Lifetime: 5 years
  - PKI Public Web Server
    - Status: Online Key Size: 2048 bits Cert Lifetime: 5 years
- Templates
  - Root Certificate Template
    - Key Size: 2048 bits
    - Cert Lifetime: 5 years
  - Enterprise Certificate Template
    - Key Size: 2048 bits
    - Cert Lifetime: 2 years

PKI Implementation Options

- In-House Private PKI
- Private PKI as SaaS
- Public PKI as SaaS
- In-House Public PKI
- Partnered Public PKI

Best Practices – Creating a PKI

- Planning
  - Do you have a security policy?
  - Define certificate policies
    - Who issues the certificates
    - How they are distributed
    - How they expire and how to deal with renewal and maintenance
  - Internal and External PKIs provide the same level of security – choose what is appropriate for your business case
- Design
  - Define your CA hierarchy
  - How you protect the CA's private keys
  - Define the publication points
- Installation
- Testing and Troubleshooting
- Monitoring

Best Practices – Using a PKI

- Protect your Private Key
  - Don't EVER email or use any clear text channel to share keys
  - In cases when you must send keys to someone, use "split channels"
    - encrypted email and phone
    - encrypted email and fax
    - encrypted email and regular mail
  - At renewal, don't reuse a private key – generate a new one.
- Certificates are distributed in PKCS #12-format files
  - These files are only password protected
  - After installing a certificate, move the files to removable media and securely delete them from the system media.
- Before disposing of an old computer, remove all certificates from it
- Monitoring

Q & A

You may also send offline questions to us via email

Luis_Santos@sra.com
Jaswant_Singh@sra.com

Web Application Security

Application Server Encryption

SRA University

Spring 2011

Luis Santos and Jaswant Singh

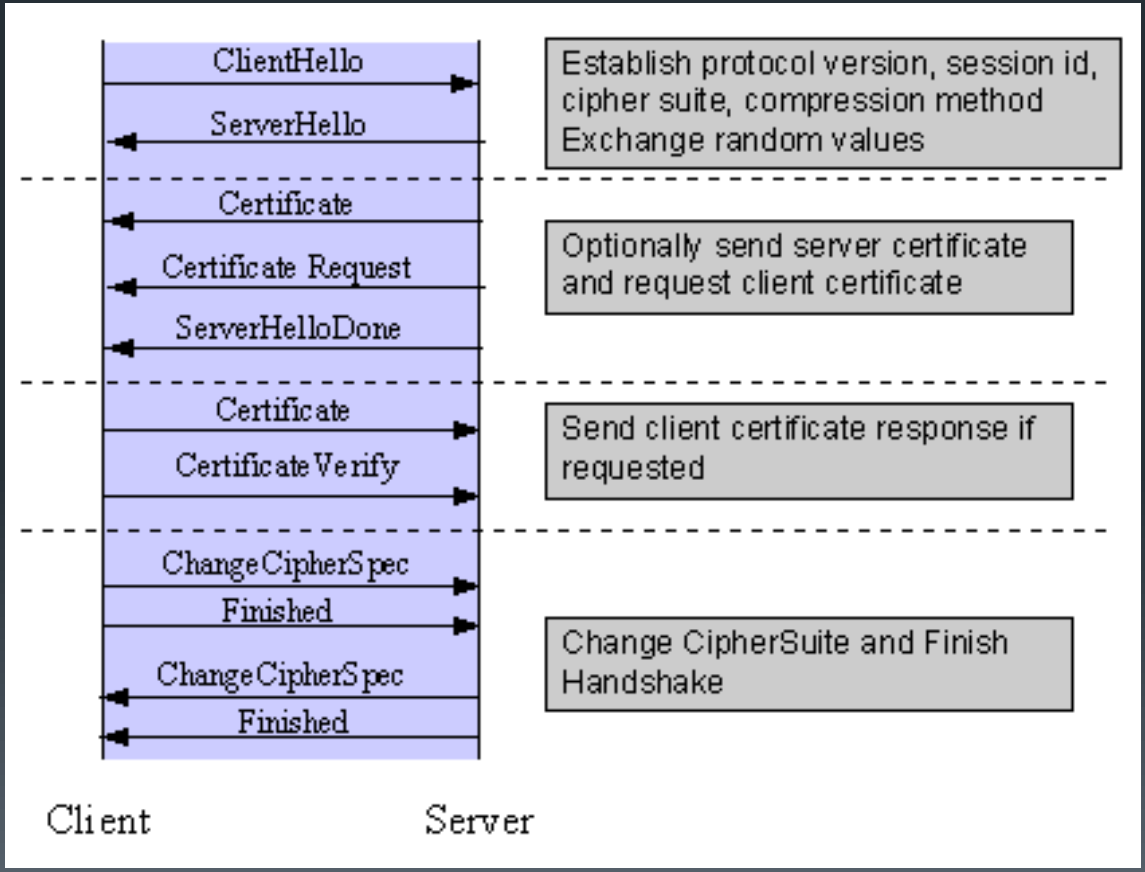
Agenda

- Why Application Server Encryption?
- SSL / TLS Overview
- SSL / TLS Options
- TLS on Commonly used Web Application Server
- Best Practices
- Q & A

Why Application Server Encryption?

- Status Quo
- Regulatory Requirements
- Information Security

TLS / SSL Overview



SSL / TLS Options

- Client Server Authentication
- Built-in TLS Implementation
- TLS Offload

SSL / TLS History

- Netscape Corporation
  - SSL v1 never published
  - SSL v2 published in 1995
  - SSL v3 published in 1996
- Internet Engineering Task Force
  - TLS v1.0 published in 1999 (RFC 2246)
  - TLS v1.1 published in 2006 (RFC 4346)
  - TLS v1.2 published in 2008 (RFC 5246)
    - Latest version in production
  - TLS v1.3 working draft

SSL / TLS Components

- Digital Certificate
  - SSL Key
  - Certificate Signed Request (CSR)
  - Certificate Authority
  - CA Certificates
  - Intermediate Certificate
  - CA Signed Certificate
- Application Setup for SSL
  - Setting up to listen over SSL
  - Setting up trust store to talk over SSL

SSL / TLS Versions

- Why disable SSL v2.0 ?
  - Malformed Client Key Remote Buffer Overflow Vulnerability
    - Execute arbitrary code as the vulnerable server process
      - Possible to write and execute a code to extract server private key
      - Possible to create denial-of-service condition
- Why TLS v 1.0 over SSL v 3.0 ?
  - Key-Hashing for Message Authentication Code (HMAC)
    - HMAC vs. MAC
  - Enhanced Pseudorandom Function (PRF)
    - PRF is defined with HMAC with two hash algorithm
  - Improved finished message verification
    - TLS uses PRF and HMAC values
  - Consistent certificate handling
    - TLS expects a Type of Certificate
  - Specific alert messages
    - TLS has better alerts messaging than SSL v 3.0

IIS Web Server Certificate Wizard

Visit Microsoft Knowledge Base
http://support.microsoft.com/kb/299875

Apache Web Server

Basic SSL Setup

- Order SSL Certificate using Open SSL
- Copy the files onto the host
- Modify the httpd.conf
  - SSLCertificateFile fullPathTo_domainname.crt
  - SSLCertificateKeyFile fullPathTo_domainname.key
  - SSLCertificateChainFile fullPathTo_intermediate.crt
- Restart Apache Web Server

SSL Setup: One Way

```
<VirtualHost _default_:1043>
    DocumentRoot "/data/applications/bssssl"
    ServerName bss.ops.aol.com
    ErrorLog "logs/bss-error_log"
    CustomLog "logs/bss-access_log" common
    CustomLog  logs/ssl_request_log "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x
  \"%r\" %b“
    SSLEngine on
    SSLOptions +StdEnvVars
    SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:-
  SSLv2:+EXP:+eNULL
    SetEnvIf User-Agent ".*MSIE.*" \
            nokeepalive ssl-unclean-shutdown downgrade-1.0 force-response-1.0

    SSLCertificateFile          conf/bss.ops.aol.com.crt
    SSLCertificateKeyFile       conf/bss.ops.aol.com.key
    SSLCertificateChainFile     conf/intermediate.crt
</VirtualHost>
```

Please Note: Make -SSLv2 instead of +SSLv2(Default) to disable SSLv2.

SSL Setup: Two Way

```
SSLCACertificateFile conf/trusted_ca.crt

SSLVerifyClient require
SSLVerifyDepth 2
SSLRequireSSL

SSLRequire %{SSL_CLIENT_S_DN_O}  eq "AOL LLC" \
                    and %{SSL_CLIENT_S_DN_CN} in
  {"App.prepay.prod"}
```

Apache Tomcat Server

SSL Setup for Apache Tomcat Server

- Create the java keystore for your site domain SSL Cert by using the key alias "tomcat“. It’s a requirement.
- Open "$JAKARTA_HOME/conf/server.xml" in a text editor.
  - Find the following section:

<Connector className="org.apache.catalina.connector.http.HttpConnector" port="8443" minProcessors="5" maxProcessors="75” enableLookups="true" acceptCount="10" debug="0" **scheme="https"** secure="true">

<Factory className="org.apache.catalina.net.SSLServerSocketFactory" **clientAuth="false" protocol="TLS” keystoreFile=“full path to the keystore" keystorePass=“xxxxx"**/>

  - clientAuth Options
    - true : 2 Way SSL
    - want: 2 Way Optional SSL
    - false: 1 Way SSL
- Restart Tomcat

WebLogic Server

SSL Setup for WebLogic Server

- Create the Java Keystore with SSL Certificate
- Copy the Java Keystore to each managed server
- Login to Admin Console
  - Save the configuration with the following changes for keystores:
    - Click On Servers Tab > ServerName > Configuration > Keystores
    - Select Keystores: Custom Identity & Java Standard Trust
    - Custom Identity keystore: full path
    - Store Type: jks
    - Keystore passphrase
  - Save the configuration with the following changes for SSL:
    - Click On Servers Tab > ServerName > Configuration > SSL
    - Identity and Trust Locations: Keystores
    - Private Key Alias: obi.ops.aol.com
    - Private Key Passphrase: xxxxx
  - You need to repeat the following steps for all the managed servers
- Restart the all the Managed Servers
- 2 Way Client Cert Behavior Options:
  - Client Cert Not Requested: 1 Way SSL
  - Client Cert Request But Not Enforced: 2 Way Optional SSL
  - Client Cert Request and Enforced: 2 Way SSL

Best Practices

- Do's
  - Always create new CSR with new private key
- Don't
  - Email or ftp Private Keys
  - Don't use any email address in CSR

Q & A

You may also send offline questions to us via email

Luis_Santos@sra.com
Jaswant_Singh@sra.com

Web Application Security

Web Services

SRA University

Spring 2011

Luis Santos and Jaswant Singh

Agenda

- Buzzwords
- What and Why Web Services?
- Securing Web Services
- WS_Security
- Open Web Application Security Project (OWASP)
- Service Oriented Architecture (SOA)
- Q & A

Buzzwords

- Web Services
- SOA
- WSDL
- REST and RESTful
- UDDI
- SSO
- SAML
- Samy and AntiSamy
- CSF

What is “Web Services” ?

a software system
designed
to
support interoperable
machine-to-machine interaction
over network

Why Web Services?

- Status Quo
- System Integration
- Reusability

Web Services Standards

- Representational State Transfer (REST)
- Simple Object Access Protocol (SOAP)
- Web Services Description Language (WSDL)
- Universal Description, Discovery and Integration (UDDI)
- Web API
  - Movement to move away from SOAP to REST
- Extensible Markup Language (XML)

Securing Web Services

- Access Controls
  - URL access policies
  - J2EE Servlet declarative security constraints
  - Data Storage access controls
- Data Transport
  - HTTPS
  - Client side certificates
  - Security built in the XML message text
    - SSO header tokens and session details
    - SAML authentication and attribute assertions
- Data Storage
  - Encryption
    - At-rest encryption (EFS, Encrypted Databases)

WS-Security Overview - 1

- Extension to SOAP that adds security to Web Services
- Published by OASIS (www.oasis-open.org)
- Specifies how integrity and confidentiality can be enforced
- Uses XML Signature and XML Encryption

- Three main mechanisms:
  - Signing SOAP messages (integrity)
  - Encrypting SOAP messages (confidentiality)
  - Attaching security “tokens” to affirm the sender’s identity

WS-Security Overview - 2

- Allows various signature formats and encryption algorithms
- Open to various token models:
  - X.509 Certificates
  - Kerberos tickets
  - User id / password credentials
  - SAML assertions
  - Custom tokens
- It is NOT a complete solution for Web Services security.

WS-Security Overview - 3

- Considerations:
  - WS-Security adds a performance penalty:
    - Message sizes are larger
    - Cryptographic processing requires more computing resources

OWASP Overview

- “Open Web Application Security Project”
- Focus is on “improving the security of application software”.
- They have several “projects”:
  - AntiSamy Project
  - Enterprise Security API Project
  - ModSecurity Core Rule Set Project

- Visit: http://www.owasp.org

Service Oriented Architecture

Service Oriented Architecture

- Simply known as SOA
- It is all about breaking systems down into their fundamental pieces and then re-assembling them based on shared, decoupled building blocks that are meaningful to the business of the enterprise.
- This does require
  - A change in mindset (i.e., education),
  - A change in roles and the way we do things (i.e., flexibility)
  - Coordination among stake holders (i.e., collaboration).

Why SOA?

- Business
  - Economies of Scale
  - Economies of Scope
- Technical
  - Ease of Reusability
  - Ease of System Integration
  - Ease of Operations and Maintenance
  - Expertise

SOA - CSFs

- Service abstraction by the business domain not by a project or application and owned by the enterprise
- Collaboration among stake holders and work as a one team than independent silos
- Continuous learning and education among stack holders to make informed decisions
- SOA Governance and Compliance
- Agile development and Agile Architecture

SOA Implementation

- Implementation Types
  - Status Quo / Tactical / Project
  - Enterprise
  - Cross Enterprise
- Implementation Challenges
  - Lack of knowledge and common understanding of SOA within stake holders
  - Lack of alignment of business and system processes to SOA
  - System is not able to live the Service and Availability promised

SOA - Milestones

SOA Governance
Overarching SOA Implementation

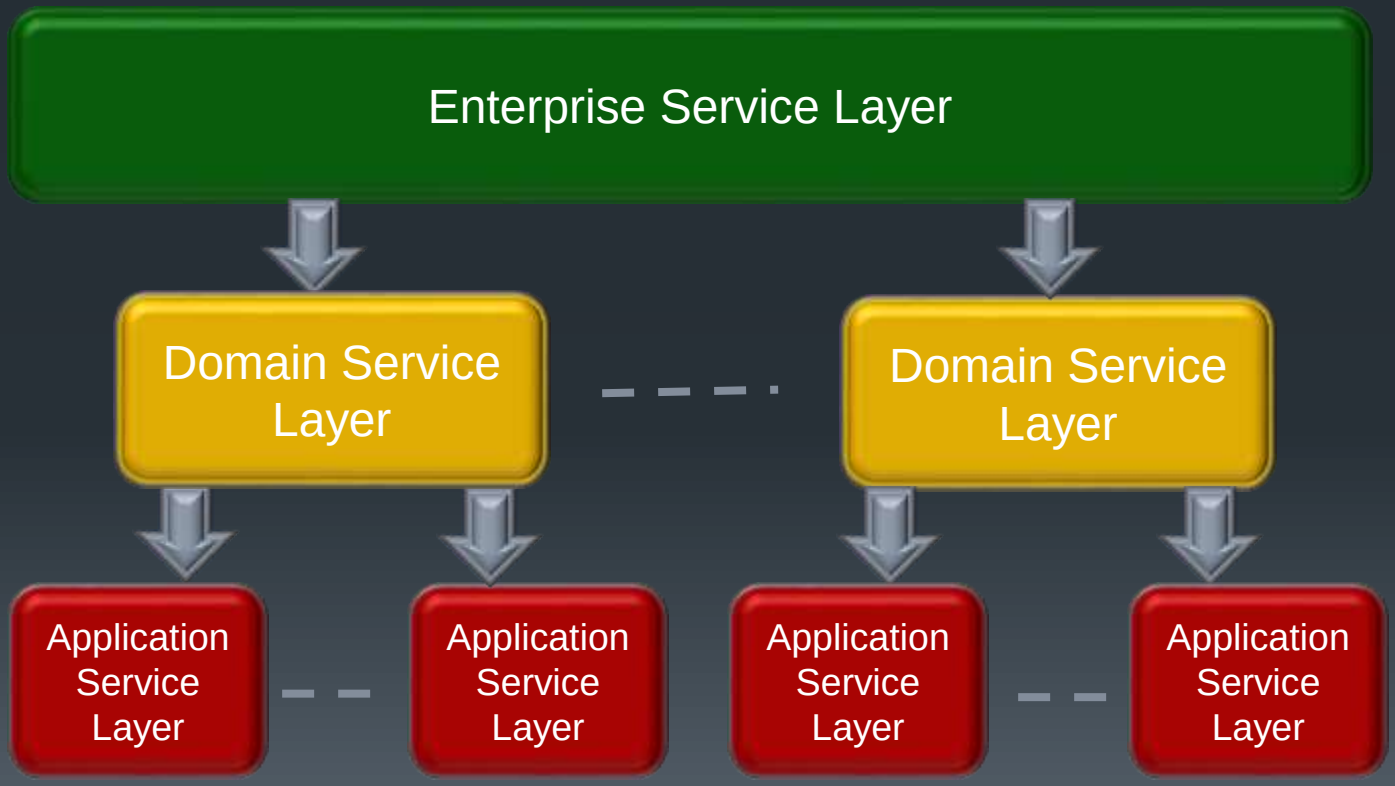
Service Catalog

Engineering of Services

- Federated vs. Non-federated
- Backward Compatibility
- Stateless Services
- Elasticity
- Availability

Community of Development

SOA – Reference Architecture



Q & A

You may also send offline questions to us via email

Luis_Santos@sra.com
Jaswant_Singh@sra.com

Web Application Security

Encrypted File System

SRA University

Spring 2011

Luis Santos and Jaswant Singh

Agenda

- Buzzwords
- What is EFS?
- Why EFS?
- How EFS Works?
- Reference Model
- Implementation Steps
- Best Practices
- Q & A

Buzzwords

- EFS vs. PGP
- DRAs
- Transparent Data Encryption (TDE)
- FISMA
- FIPS

What is EFS?

A solution
to protect data at rest
even when you have authorized access
to the server
that
enables transparent
encryption and decryption of files

Why EFS?

- Status Quo
- Regulatory Requirements
- Information Security at Rest
  - EFS is available through standard offering in most modern Operating System like MS Windows, Linux and Unix
  - EFS standard implementation uses advanced and standard cryptographic algorithms usually FISMA and FIPS compliant
  - EFS facilitates Transparent Data Encryption (TDE)

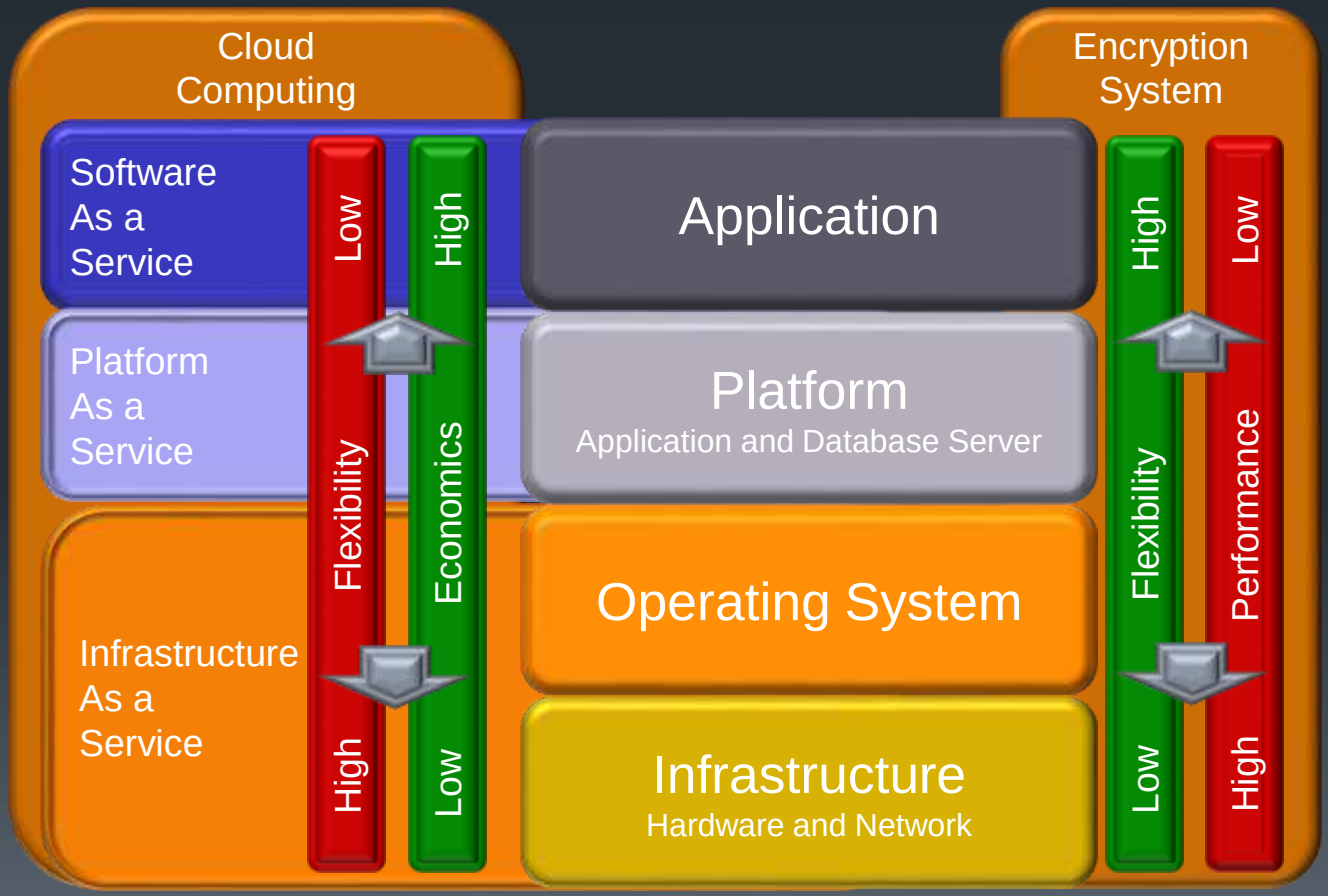
How EFS Works?

- Create File
  - EFS provider produces a symmetric key for file encryption also known as “file encryption key”
  - EFS provider gets the Encryption User(s) and DRAs public key
  - Uses these public key(s) to encrypt the “file encryption key”
  - Using “file encryption key” encrypts the file
  - Stores the file encryption keys and encrypted file
- Read File
  - EFS provider looks for user profile to get users private key
  - Using users’ private key decrypts “file encryption key”
  - Using decrypted “file encryption key” decrypts the file for encryption user
- Modify File
  - It may do exactly like Create File or reuse the previous “file encryption key”

Implementation Steps

- Understand the Business Problem
  - Optimize the data encryption needs
  - BCP needs
  - Data Size and Replication
  - Backup and Recovery
- Private vs. Public vs. Self Signed PKI
  - Key Length and Key Management
  - Setup Encryption User Profile
  - Setup two Disaster Recovery Agents (DRAs)
- Platform and Application Setup
  - Run them as Encryption User
- Migration
  - Identify Resources
    - CPU, Memory, Storage and Network Pipe
  - Make Fall Back Plans
  - Data Transformation and Replication
  - Timeline
  - System Availability

Reference Model



Best Practices

- Setup at least two DRAs
- Setup offline Recovery System
- Test Recovery System
- File System Backup and Replication
  - Block Level vs. Full File
- Certificate and Key Management
  - Suggested Key Length - 2048
  - Setup Safe Deposit for DRAs private keys
  - Public vs. Private PKI
  - Setup Certificate Templates
    - File Encryption
    - File Encryption Recovery Agent
- Migration Planning
  - Ship over network vs. hard disk
  - Encrypting Data
    - Distributed Computing vs. Throttling Resources
    - Throttling - Network and CPU

Q & A

You may also send offline questions to us via email

Luis_Santos@sra.com
Jaswant_Singh@sra.com

Web Application Security

Encrypted Database

SRA University

Spring 2011

Luis Santos and Jaswant Singh

Agenda

- Buzzwords
- What is Database Encryption?
- Why Database Encryption?
- Implementation Steps
- Q & A

Buzzwords

- Data at Rest
- Data in Transit
- Transparent Data Encryption (TDE)
- Perimeter
- Network Zone

What is Database Encryption?

A Database is said to be
encrypted
when the data stored in it is
encrypted and decrypted
transparently
for
its authorized users

Why Database Encryption?

- Securing the perimeter may not be an option
  - The network may have to be open to accommodate suppliers, partners, or customers
- Even if the perimeter is secure, another layer of security may be desired
- Regulatory Compliance: your requirements may dictate that “Data at Rest” must be encrypted

Considerations

- Encryption increases storage needs
- Encryption has a performance penalty
  - To access data, you need to decrypt it
  - To store or update data, you need to encrypt it
- Do you need to encrypt everything?
- Encrypting indexed fields may have significant performance impact
- Do you need to protect data in transit as well as at rest?
  - It's a good idea!

More Considerations

- How many encryption keys will you need?
- How will you manage the keys?
- Where will the keys be stored?
- How will you protect access to the keys?
- How often should the keys change?

Even More Considerations

- With DB-supported encryption, applications don't need to be changed
- Check your database software features:
  - Some vendors support "all or nothing"
  - Other support selective encryption – you can pick and choose
- This is really a solution for securing data at rest

- Bear in mind that encrypting / decrypting requires additional processing as compared to no encryption

Implementation Steps

- Plan Ahead
- Pick an implementation that works for you
  - Most databases provide TDE, e.g., Oracle, SQL Server, MySQL
  - Fallback options:
    - Encrypted File System (EFS) + sTunnel
    - EFS + IPSec
    - Application-level Encryption
- If you don’t use a TDE-compliant database, you need a key management strategy

Q & A

You may also send offline questions to us via email

Luis_Santos@sra.com
Jaswant_Singh@sra.com