Certificados Digitales

Acrónimos

- **BER (Basic Encoding Rules), CER (Canonical Encoding Rules), DER (Distinguished Encoding Rules):** certificado en forma binaria encodeados según la **ITU-T X.690** (Con base en ASN.1)
- **CRT (Certificate):** En referencia a certificados CER o DER.
- **CRL ("Certificate Revocation List"):** Lista de certificados revocados
- **PEM (Privacy-enhanced Electronic Mail):** RFC1421/22/23/24, usualmente aplicado en un certificado DER encodeado en Base64 contenido entre las líneas: `"-----BEGIN CERTIFICATE-----"` y `"-----END CERTIFICATE-----"`
- **JKS (Java Key Store):** Formato de repositorios de claves-certificados
- **PKCS (Public-Key Cryptography Standards):** Estándares definidos por RSA Security para el manejo de información con algoritmos asimétricos. La siguiente tabla enumera y sintetiza estos estándares

Por supuesto, aquí tienes el texto extraído de la segunda imagen:

Acrónimos

- **BER (Basic Encoding Rules), CER (Canonical Encoding Rules), DER (Distinguished Encoding Rules):** certificado en forma binaria encodeados según la **ITU-T X.690** (Con base en ASN.1)
- **CRT (Certificate):** En referencia a certificados CER o DER.
- **CRL ("Certificate Revocation List"):** Lista de certificados revocados
- **PEM (Privacy-enhanced Electronic Mail):** RFC1421/22/23/24, usualmente aplicado en un certificado DER encodeado en Base64 contenido entre las líneas: `"-----BEGIN CERTIFICATE-----"` y `"-----END CERTIFICATE-----"`
- **JKS (Java Key Store):** Formato de repositorios de claves-certificados
- **PKCS (Public-Key Cryptography Standards):** Estándares definidos por RSA Security para el manejo de información con algoritmos asimétricos. La siguiente tabla

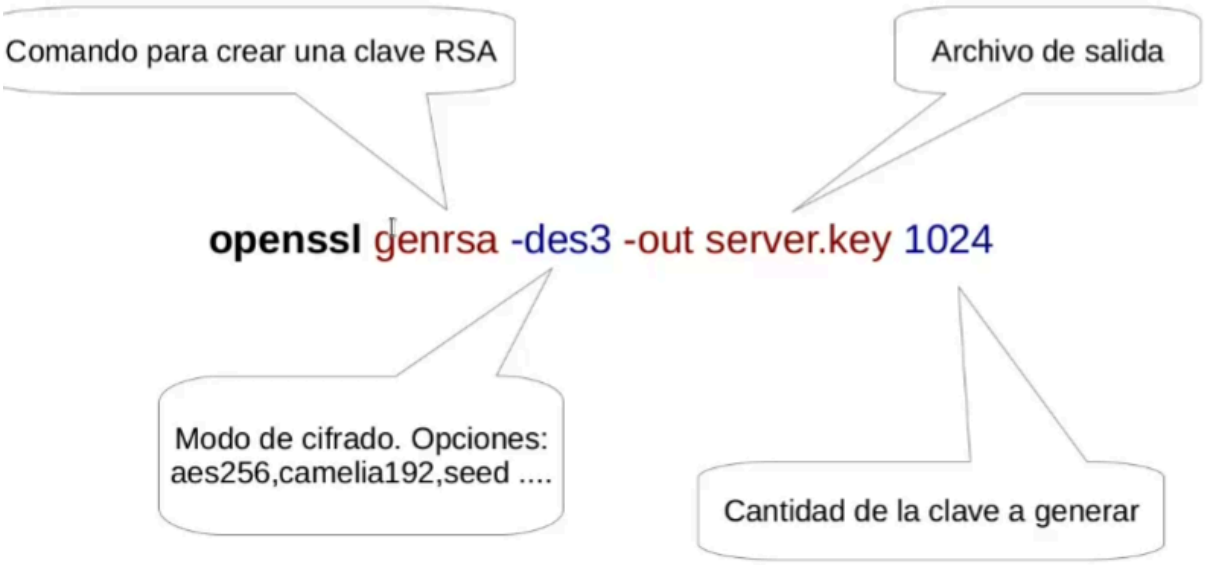
enumera y sintetiza estos estándares.

el **PKCS 12 es el que se usa para el repositorio**

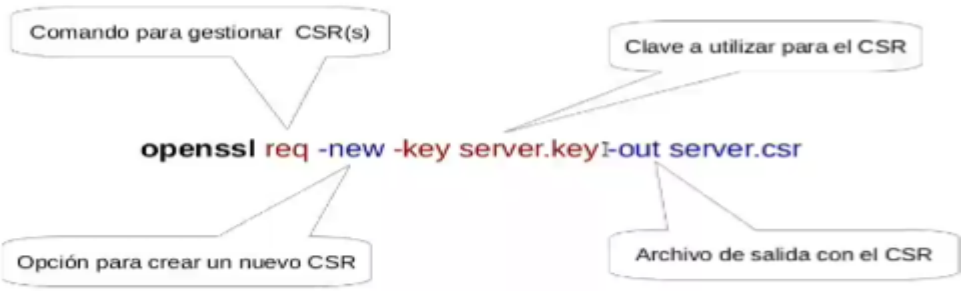
Claro, aquí tienes el contenido de la tabla con un formato organizado y legible:

Acrónimo	Contenido	Referencias y detalles
PKCS#1	Estándar criptográfico **RSA**	RFC 3447. Define el formato del cifrado **RSA**
PKCS#2	Obsoleto	Cifrado de resúmenes. Incorporado en **PKCS#1**
PKCS#3	Estándar de intercambio de claves	**Diffie-Hellman**
PKCS#4	Obsoleto	Sintaxis de claves. Incorporado en **PKCS#1**
PKCS#5	Estándar de cifrado basado en contraseñas	Padding. **RFC 2898** y **PBKDF2**
PKCS#6	Estándar de sintaxis de certificados	Extendidos X.509 v1. (No en V3)
PKCS#7	Estándar sobre la sintaxis del mensaje criptográfico	**RFC 2315**. Firmar y cifrar mensajes PKI (.p7b, .p7c)
PKCS#8	Estándar sobre la sintaxis de la información de **clave privada**	**RFC 5208**
PKCS#9	Tipos de atributos seleccionados	-
PKCS#10	Estándar de solicitud de certificación	-
PKCS#11	Interfaz de dispositivo criptográfico (“CryptographicTokenInterface” o cryptoki)	Define un **API** genérico de acceso a dispositivos criptográficos
PKCS#12	Estánd<ar de sintaxis de intercambio de información personal	Formato de repositorio de claves/certificados (.p12, .pfx)
PKCS#13	Estándar de criptografía de curva elíptica	-
PKCS#14	Generación de número pseudo-aleatorios	-
PKCS#15	Estándar de formato de información de dispositivo criptográfico	-

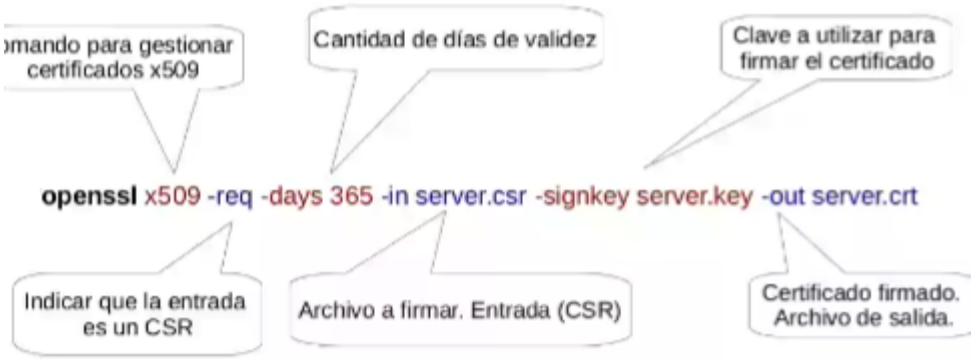
Generar clave RSA



Generar CSR(Certificate Signing Request)



Generar un certificado “Self-Signed”

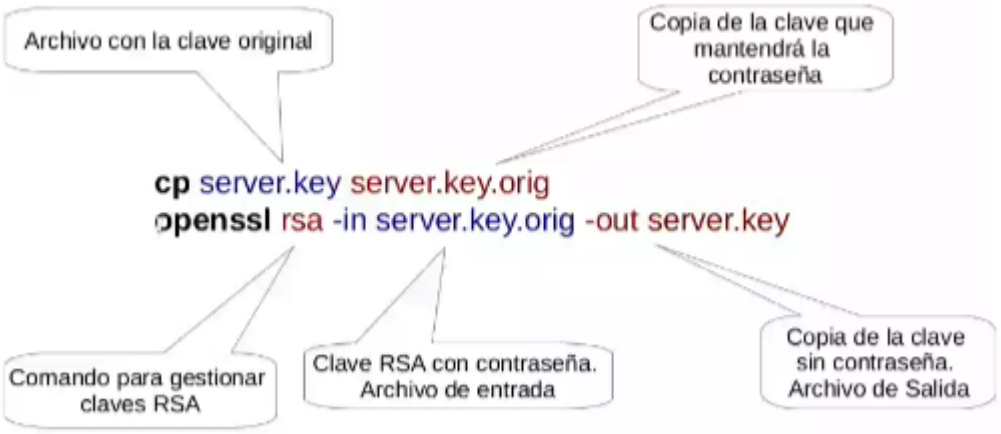


Esta es una alternativa a comprar o solicitar la firma de un tercero confiable (Ej. Verisign)

El problema de este certificado es que no te lo puede validar un tercero .

```
OwNGdp7PZaWwHIyazuJ4BfZfsseS19Qkg4VK0erICLgwcDB9Bl+KYSI13DmVY6Bb
J/2gUA==
-----END CERTIFICATE REQUEST-----
dagger@bluegene:~/scaw2025c1$ openssl x509 -req -days 365 -in server.csr
 -signkey server.key -out server.crt
Enter pass phrase for server.key:
Certificate request self-signature ok
subject=C = AR, ST = Buenos Aires, L = Buenos Aires, O = UNLAM, OU = DII
T, CN = scaw.diit.unlam.edu.ar, emailAddress = wureta@unlam.edu.ar
dagger@bluegene:~/scaw2025c1$ ls -l
total 12
-rw-r--r-- 1 dagger dagger 1017 Jun  9 20:26 server.crt
-rw-r--r-- 1 dagger dagger  729 Jun  9 20:19 server.csr
-rw------- 1 dagger dagger 1062 Jun  9 20:13 server.key
dagger@bluegene:~/scaw2025c1$ cat server.crt
-----BEGIN CERTIFICATE-----
MIICwjCCAisCFEvLoHOWC6/MsYddGhGxX+fglEc/MA0GCSqGSIb3DQEBCwUAMIGf
MQswCQYDVQQGEwJBUjEVMBMGA1UECAwMQnV1bm9zIEFpcmVzMRUwEwYDVQQHDAxC
dWVub3MgQWlyZXMxDjAMBgNVBAoMBVVOTEFNMQ0wCwYDVQQLDARESUlUMR8wHQYD
VQQDDBZzY2F3LmRpaXQudW5sYW0uZWR1LmFyMSIwIAYJKoZIhvcNAQkBFhN3dXJ1
dGFAdW5sYW0uZWR1LmFyMB4XDTI1MDYwOTIzMjYwMVoXDTI2MDYwOTIzMjYwMVow
gZ8xCzAJBgNVBAYTAkFSMRUwEwYDVQQIDAxCdWVub3MgQWlyZXMxFTATBgNVBAcM
DEJ1ZW5vcyBBaXJlczEOMAwGA1UECgwFVU5MQU0xDTALBgNVBAsMBERJSVQxHzAd
BgNVBAMMFnNjYXcuZGlpdC51bmxhbS51ZHUuYXIxIjAgBgkqhkiG9w0BCQEWE3d1
cmV0YUB1bmxhbS5lZHUuYXIwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMCc
m/HB6RUgS0uX1FGIqw3VyEk5RVI/uckhd9sa/l8xbS9tzCD6SZg1XdbVUuCgMkW0
Hk8dBKPt+2cxx5y8gp7DXG3qabgrsVLVGEqHCMq3e9nReQSHNRsBSu9+zdBVp9f0
dzpCdwa0N6cj/P4ETZBeXuIrFzmwpp3JYiy4IB95AgMBAAEwDQYJKoZIhvcNAQEL
BQADgYEABkz9tHEjbRcNo/4dZlRBcA/Y+/KErHvxHslu4WZJgK35hJaYvDosPHSd
QX29ea1tI7hnRM6oVD4mI8iiu0yjlx6OP4i3+dS8yEOPGOf9rJ57MT4YIczS7p/A
ypVFTBcuMhfeujMkdDwSDDgUpDxsuHfppPqKzG+32qb8ffhyeO8=
-----END CERTIFICATE-----
```

Remover la contraseña de una clave RSA



Claro, aquí tienes el texto extraído de la imagen en modo texto:

SSL en Apache Server (httpd)

Instalar la clave privada y su certificado

cp server.crt /usr/local/apache/conf/ssl.crt/
cp server.key /usr/local/apache/conf/ssl.key/

Configurar SSL en el Virtual-Host (httpd-vhosts.conf)

SSLEngine on
SSLCertificateFile /usr/local/apache/conf/ssl.crt/server.crt
SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/server.key
SetEnvIf User-Agent "^Mozilla.*$" nounsubscribe ssl-unsubscribe
CustomLog logs/ssl_request_log

```
%6t %SSLCipherSuite=%{SSL_CIPHER}x %SSL_PROTOCOL=%{SSL_PROTOCOL}x
%SSL_CIPHER_USEKEYSIZE=%{SSL_CIPHER_USEKEYSIZE}x
%SSL_TLS_COMPRESS=%{SSL_TLS_COMPRESS}x %SSL_ERROR=%{SSL_ERROR}x
```

Reiniciar el servidor Apache

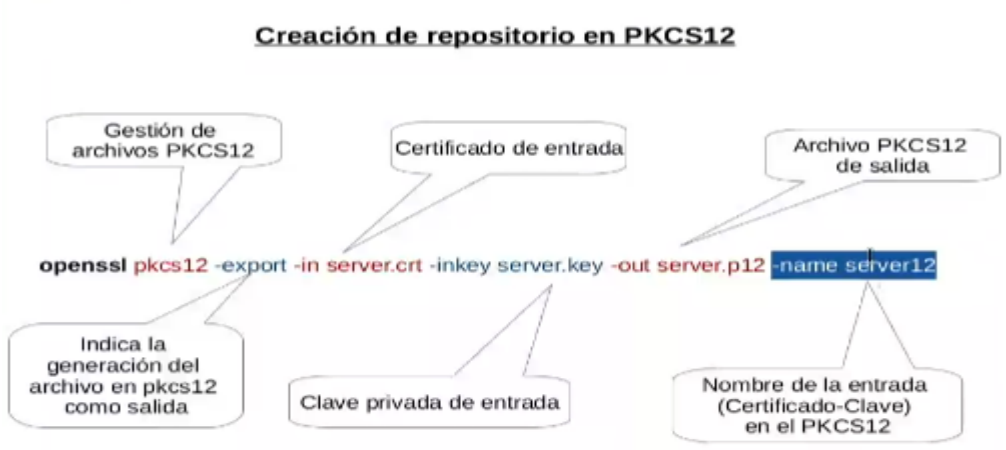
TLS en Apache Tomcat

Este es el ejemplo correspondiente a la implementación de APR, donde no utilizamos un repositorio sino los archivos del certificado y la clave en formato PEM.

<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
protocol="HTTP/1.1"
port="8443" maxThreads="200"
scheme="https" secure="true" SSLEnabled="true"
SSLCertificateFile="/usr/local/ssl/server.crt"
SSLCertificateKeyFile="/usr/local/ssl/server.key"
SSLCertIFICATEFile="/usr/local/ssl/server.crt"
SSLCertificateKeyFile="/usr/local/ssl/server.key"
SSLVerifyClient="optional" SSLProtocol="TLSv1"/>

aca TLSv1 se define que version vas a estar trabajando y restringir las compatibildia co nel servidor

Creacion de repositorio en PKC12



Comienzo de la unidad n 4:Aplicación de seguridad

Logging (nose toma mucho en cuenta por que lo vimos en la unidad 2)

Su finalidad se vincula a los siguientes puntos:

- Identificar incidentes de seguridad
- Monitorear violaciones a las políticas
- Asistir en controles de no-repudio
- Proveer información sobre problemas o situaciones atípicas
- Contribuir con información específica para la investigación de incidentes que no pueda obtenerse de otras fuentes
- Contribuir con la defensa ante vulnerabilidades y exploits mediante de la detección de ataques

---

Logging

¿Dónde registrar?
Sistemas de archivos, almacenamiento en la nube, bases de datos SQL y NoSQL

¿Qúe registrar?
Fallos de validación, autenticación, autorización, anomalías, fallas de aplicación, eventos legales...

¿Qúe NO registrar?
Código fuente, identificadores de sesión, credenciales y tokens de acceso, claves de cifrado, SPI...

Enmascaramiento de datos

Es el proceso mediante el cual se reemplazan los datos sensibles de un sistema con el objetivo de proteger esta información confidencial ante situaciones que escapen al control sobre la misma.

Según GDPR, *"seudonimización"*, es el tratamiento de datos personales de manera tal que ya no puedan atribuirse a un interesado sin utilizar información adicional, siempre que dicha información adicional figure por separado y esté sujeta a medidas técnicas y organizativas destinadas a garantizar que los datos personales no se atribuyan a una persona física identificada o identificable.

hay procesos para facilitar el enmascaramiento ,en un entorno real hay que hacer primero el proceso de enmascaramiento y despues trabajar con los datos

Aquí tienes el texto extraído de la imagen:

Logging

Debilidades asociadas:

- CWE-117 Improper Output Neutralization for Logs
- CWE-223 Omission of Security-relevant Information
- CWE-532 Insertion of Sensitive Information into Log File
- CWE-778 Insufficient Logging

Validación de entrada

La debilidad de seguridad más común en aplicaciones web es la falta de validación apropiada de las entradas del cliente o del entorno. Esta debilidad lleva a casi todas las principales vulnerabilidades en las aplicaciones web, tales como inyección a intérprete, ataques locale/Unicode, ataques al sistema de archivos y desbordamientos de memoria.

Nunca se debe confiar en los datos introducidos por el cliente, ya que tiene todas las posibilidades de manipular los datos.

Aquí tienes el texto extraído de la imagen:

Validación de entrada

La validación debe cubrir los aspectos **semánticos** como **sintácticos** de la información ingresada.
Se sugiere implementar una metodología de validación de tipo "White List" para solo validar

el ingreso de datos permitidos.
Para este fin se debe describir el tipo válido de datos, se sugiere implementarlo mediante listas de valores o expresiones regulares.

Aquí tienes el texto extraído de la imagen:

Validación de entrada

Expresiones regulares de validación de entrada por tipos:

- **URL:** ^(((https?|ftp?s?|gopher|telnet|nntp)://)(mailto:|news:))(%[0-9A-Fa-f]{2}|-(|!~*'/?:@&=+$,A-Za-z0-9])+)(.[.!?/?:,;[]:|abcknmpca-z0-9]+)?$

- **IP:** ^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$

- **EMAIL:** ^[a-zA-Z0-9&*-]+(?:.[a-zA-Z0-9_+-]+)*(@:[a-zA-Z0-9]+.[a-zA-Z]{2,7})$

- **NUMEROS:** ^(cero|uno|dos|tres|cuatro|cinco|seis|siete|ocho|nueve)$

Aquí tienes el texto extraído de la imagen:

Autenticación en la Web

Su objetivo es proveer servicios de autenticación segura a las aplicaciones Web, mediante:

- Vinculando una unidad del sistema a un usuario individual mediante el uso de una credencial
- Proveyendo controles de autenticación razonables de acuerdo al riesgo de la aplicación.
- Denegando el acceso a atacantes que usan varios métodos para atacar el sistema de autenticación.

Cuando hablamos de credenciales hablamos de un conjunto más amplio que de usuario y contraseña

cuando subo la afectó a la seguridad y disponibilidad

3 punto : autodefensa cuando la app detecta una anomalía y hace una contención

es muy perjudicial que bloquear el acceso total a un usuario ,en el caso de loguaerse si se equivoca 3 veces se lo bloquea temporalmente para que depues lo vuaelva a intentar,si se vuelve a equivocar si incremente el tiempo del bloqueo temporal ,esto para evitar ataques a fuerza bruta.

Se recomienda que si un usuario se crea una cuenta su nombre es por ejemplo Walterureta y viene otro walterureta ,lo ideal no deberia dejar que se cree el segundo usuario ,por que seria 2 walter ureta

---

Autenticacion en la web

Consideraciones generales

- La autenticación es solo tan fuerte como sus procesos de administración de usuarios
- Use la forma más apropiada de autenticación adecuada para su clasificación de bienes
- Re-autenticar al usuario para transacciones de alto valor y acceso a áreas protegidas
- Autenticar la transacción, no el usuario
- Las contraseñas son trivialmente rotas y no son adecuadas para sistemas de alto valor

---

Autenticación en la Web
Buenas prácticas

1. **User IDs:** Verificar que **NO** sean case sensitive, para evitar confusiones del tipo “juan” o “Juan”.
2. **Fortaleza de contraseñas:** su longitud mínima debería ser de al menos 10 caracteres y la máxima no inferior a 20, con recomendación de 128 caracteres. Evaluar su complejidad.
3. **Implementar métodos seguros de recuperación:** Con canales externos o diversos puntos de información no selectivos (preguntas de seguridad, de texto libre).
4. **Almacenar contraseñas de forma segura:** con un soporte criptográfico adecuado.
5. **Transmitir contraseñas solo sobre TLS:** la pagina de logueo y todas las posteriores deben operar bajo TLS.

es normal que tengamos rotacion de claves

aveces la info no viaja por canales seguros y una forma de que sea seguro temporalmente es el TLS

Autenticación en la Web
6) Solicitar re-autenticación: Para evitar ataques de CSRF y Hijacking de sesión se debe solicitar autenticación nuevamente antes de realizar operaciones sensibles.
7) Utilizar sistemas de autenticación de factor múltiple: Son aquellos que se

comprenden como autenticación fuerte, estando compuestos por más de un elemento de los siguientes tipos

- Algo que se sabe...
- Algo que se tiene...
- Algo que se es...
- Ubicación...

8) Manejo de mensajes de error: los mensajes de error deben ser genéricos a fin de no facilitar ningún indicio o información de los datos de autenticación del sistema.
9) Prevenir ataques por fuerza bruta: mediante captchas y controles de comportamiento (tiempo entre intentos, cantidad, etc)

Nos quedamos en la pag 12 de 51

Terminamos con la unidad 4 ,la clase que viene vemos la unidad 5 y 6

Claro, aquí tienes el texto extraído de la imagen:

Autenticación en la Web

Métodos de protección ante ataques de automatización:

• MFA o Autenticación de factor múltiple
• Bloqueo de cuenta
• CAPTCHA
• Preguntas de seguridad o palabras a memorables

Brute Force: Probar múltiples valores de contraseñas de un diccionario contra para una única cuenta.
Credential Stuffing: Probar combinaciones usuario-contraseña obtenidas de otras brechas de seguridad.
Password Spraying: Probar una contraseña débil sobre un gran número de cuentas.

Contraseñas de un solo uso

OTP "One Time Password" o "Contraseña de un solo uso", corresponde a un valor confidencial que no puede ser reutilizado.

HOTP
Algoritmo de generación de "Contraseñas de un solo uso basadas en HMAC". RFC-4226

TOTP
"Time-based One Time Password" o "Contraseña de un solo uso basada en tiempo", es un valor confidencial que no puede ser reutilizado y que además cuenta con un tiempo de vida acotado. RFC-6238.

Aquí tienes el texto extraído de la tercera imagen:

Técnicas de autenticación de Usuarios

1. Autenticación básica y segura (HTTP-Basic, HTTP-Digest) (
2. Autenticación basada en formularios (Usuario-Contraseña)
3. Autenticación integrada (ISS, ASP.NET, Active Directory)
4. Autenticación basada en certificado (x509)
5. Autenticación fuerte o de factor múltiple (Algo que sabes, algo que tienes, algo que eres, tu ubicación)

3 las app se apoya en un sistema central para dar el servicio

4 cada cliente debe configurar el certificado para acceder al sistemas ,seria un paso mas

Autenticacion en la web

- Re uso de contraseñas
- Autenticacion sin contraseñas

	OAuth2	OpenID	SAML
Formato del token	JSON o SAML2	JSON	XML
Autorización	Sí	No	Sí
Autentificación	Sí	Sí	Sí
Versión	OAuth2 (2005)	OpenClient (2006)	SAML 2.0 (2001)
Notas	Basado en TLS	-	-
Aplicación	Autorización de API	SSO (Consumidor)	SSO (Empresarial)

OpenID SOLO SIRVE PARA AUTENTICACION

Claro, aquí tienes el texto extraído de la imagen:

JWT – JSON Web Token

Es un estándar abierto (rfc7519) utilizado para transmitir de forma segura información en formato JSON. Está compuesto por tres partes: el encabezado que especifica el tipo de token y el algoritmo de firma, la carga útil que contiene los datos relevantes y la firma que verifica la integridad del token. Los tokens JWT se utilizan en aplicaciones web y servicios API para la autenticación y autorización, ya que son seguros, compactos y autocontenidos. Son portables y se pueden utilizar en diferentes sistemas siempre y cuando se comparta la clave necesaria para verificar la firma del token. Esto permite la interoperabilidad entre servicios y aplicaciones.

Aquí tienes el texto extraído de la imagen:

Autenticación en la Web (Contraseña olvidada)

- Utilizar un mensaje único ya sea que la cuenta exista o no.
- Asegurar que el tiempo de respuesta al usuario sea uniforme.
- Utilizar otro canal para informar el método de re-esteo de la contraseña.
- Utilizar tokens con URL s para una implementación simple y rápida.
- Asegurar que los tokens sean aleatorios, de longitud que provea resistencia a fuerza-bruta, almacenamiento seguro, con tiempo de expiración y de un solo uso.
- No alterar la cuenta del usuario hasta la presentación del token.

Aquí tienes el texto extraído de la imagen:

Autorización en la Web

Sus objetivos son

- Asegurar que únicamente usuarios autorizados puedan realizar acciones permitidas con su correspondiente nivel de privilegio.
- Controlar el acceso a recursos protegidos mediante decisiones basadas en el rol o el nivel de privilegio.
- Prevenir ataques de escalada de privilegios, como por ejemplo utilizar funciones de administrativas siendo un usuario anónimo o incluso un usuario autenticado.

Autorización en la Web

Métodos de control de acceso

1. Role Based Access Control (RBAC)

2. Discretionary Access Control (DAC)
3. Mandatory Access Control (MAC)
4. Attribute Based Access Control (ABAC)

Aquí tienes el texto extraído de la imagen:

Autorización en la Web

En Role Based Access Control (RBAC), las decisiones de acceso se basan en las funciones y responsabilidades de un individuo dentro de la organización o de la base de usuarios. El proceso de definición de las funciones se basa por lo general en el análisis de los objetivos y la estructura de una organización que por lo general está relacionada con la política de seguridad. Por ejemplo, en una organización médica, los diferentes roles de los usuarios pueden incluir aquellos como médico, enfermera, asistente, enfermera, pacientes, etc. Estos miembros requieren diferentes niveles de acceso para llevar a cabo sus funciones, sino también los tipos de las transacciones Web y su contexto permite variar mucho dependiendo de la política de seguridad y los reglamentos pertinentes (HIPAA, Gramm-Leach-Bliley, etc.)

Aquí tienes el texto extraído de la imagen:

Autorización en la Web

El Discretionary Access Control (DAC) es un medio para restringir el acceso a la información sobre la base de la identidad de los usuarios y/o la pertenencia a ciertos grupos. Decisiones de acceso se basan normalmente en las autorizaciones concedidas a un usuario basándose en las credenciales que presentó en el momento de la autenticación (nombre de usuario, contraseña, hardware / identificador de software, etc.) En los modelos más típicos del DAC, el propietario de la información o cualquier recurso es capaz de cambiar los permisos a su discreción (de ahí el nombre). DAC tiene el inconveniente de que los administradores no son capaces de gestionar de forma centralizada los permisos de los archivos / datos almacenados en el servidor web. Por ejemplo, el sistema de archivos de un sistema Unix (rwx).

Aquí tienes el texto extraído de la imagen:

Autorización en la Web

Mandatory Access Control (MAC) garantiza que la ejecución de la política de seguridad de la organización no se basa en el cumplimiento del usuario en la aplicación web. MAC asegura la información mediante la asignación de etiquetas de sensibilidad en la información

y comparando esto con el nivel de sensibilidad de un usuario está operando a. En general, los mecanismos de control de acceso MAC son más seguras que las DAC todavía tener soluciones de compromiso en el rendimiento y la conveniencia de los usuarios. Mecanismos MAC asignan un nivel de seguridad a toda la información, asignar un control de seguridad de cada usuario, y garantizar que todos los usuarios sólo tengan acceso a los datos pertinentes. MAC es generalmente apropiado para sistemas extremadamente seguras como aplicaciones militares seguras multinivel o aplicaciones de datos de misión crítica.

ESTE ES EL MAS JERARQUICO

Autorización en la Web

Attribute Based Access Control (ABAC) Es un sistema de control basado en atributos asignados a cualquier componente del sistema (usuarios, recursos, etc). El mismo establece políticas que utilizan la información de dichos atributos para establecer si un acceso está permitido, de esta forma se establece un sistema de control basado en relaciones. Los atributos son del tipo clave-valor, generalmente se utiliza un lenguaje de soporte como XACML para definir las reglas de acceso. Este modelo es muy utilizado en las plataformas actuales de Cloud. Ejemplo: "*Los usuarios pueden acceder a los servicios del proyectoA si pertenecen al grupo de DesarrolloA*". Aquí vemos que la política requiere tres atributos rol:usuario, servicio:proyectoA, grupo:desarrolloA.

No necesario esta obligado a utilizar el mismo tipo de acceso

El discrecional lo tenes en la parte de linux

Aquí tienes el texto extraído de la imagen:

Buenas prácticas de implementación

- Codificar el control en la actividad objetivo
- Disponer de un Controlador Centralizado (ACL)
- Utilizar un Control Central de Acceso, en las diferentes capas
- Verificar la política del lado del servidor (Server-side)

La politica se verifica siempre en el lado del servidor

Aquí tienes el texto extraído de la imagen:

Ataques de control de acceso

1. **Vertical Access Control Attacks** - Un usuario convencional obtiene accesos superiores o de administrador.

2. **Horizontal Access Control attacks** - Con el mismo rol o nivel el usuario puede acceder a información de otros usuarios.
3. **Business Logic Access Control Attacks** - Usar de una o más actividades para realizar una operación con un resultado no autorizado para ese usuario.

Administración de Usuarios y privilegios

Mejores prácticas

- Cuando se está diseñando aplicaciones, trazar la funcionalidad administrativa fuera y asegurarse de que los controles apropiados de acceso y auditoría están en su lugar.
- Considerar procesos – en algunas ocasiones, todo lo que se requiere es entender cómo los usuarios pueden ser prevenidos de utilizar una característica con la simple falta de acceso.
- Acceso de servicio de asistencia siempre es un término medio – ellos necesitan acceso para ayudar a los clientes, pero no son administradores.(no es una practica sana por que alguien puede forzar el acceso,no es una buena practica) .
- Diseñar cuidadosamente la funcionalidad de servicio de asistencia / moderador / soporte al cliente alrededor de una capacidad administrativa limitada y aplicación segregada o acceso.

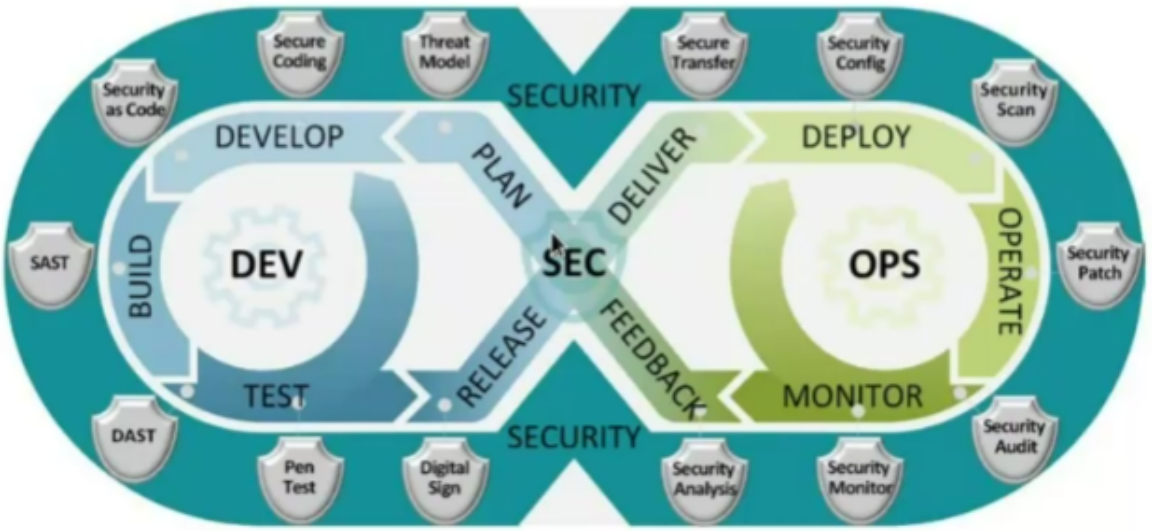
Aquí tienes el texto extraído de la imagen:

Administración de Usuarios y privilegios

- Todos los sistemas deberían tener aplicaciones separadas del acceso de los usuarios para los administradores. (Esto no es lo mas comun) .
- Sistemas de alto valor deberían separar estos sistemas en un servidor separado, que tal vez no sea accesible desde el amplio Internet sin acceso para la administración de redes, como a través del uso de una VPN fuertemente autenticada o desde la red de un centro de operaciones de confianza.

DevSecOps

Es un conjunto de prácticas que combinan el desarrollo de software (Dev), la seguridad (Sec), y las operaciones de tecnología de la información (Ops) para asegurar y acortar el ciclo de vida del desarrollo de software.



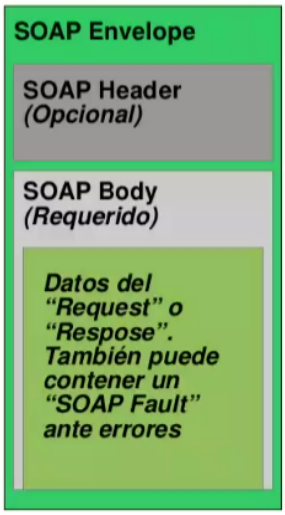
El pen test para que planifique distintos tipos de ataques

Aquí tienes el texto extraído de la imagen:

Esta tecnologia se usa mucho en el uso de software empresarial

Web Services

En el nivel más simple, los servicios web pueden ser vistos como aplicaciones web especializadas que difieren principalmente en la capa de presentación. Mientras que las aplicaciones web son típicamente basadas en HTML, los servicios web son basados en XML/SOAP.



Es muy flexible en cuanto a estructura ,tiene

Web Services

Comités de estándares

- W3C (http://www.w3.org), estándares de esquema XML (XML Schema), SOAP (Simple Object Access Protocol), XML-dsig, XML-enc y WSDL
- OASIS (http://www.oasis-open.org), estándares de WS-Security
- OASIS (http://uddi.xml.org), UDDI (Universal Description Discovery and Integration)
- OASIS (http://saml.xml.org), SAML (Security Assertion MarkUp Language)
- Grupo de interoperabilidad de servicios web (WS-I - http://www.ws-i.org/)

Aquí tienes el texto extraído de la imagen:

Web Services

Los servicios Web típicamente representan una interfaz pública funcional, que se llama de forma programática.

UDDI la gente no se uso mucho en la aparte de busqueda pero si lo de registro publico

---

Web Services

Definition WSDL - Parte 1

```
<definitions name="HelloService"
targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<message name="helloRequest">
  <part name="name" type="xsd:string"/>
</message>
<message name="helloResponse">
  <part name="return" type="xsd:string"/>
</message>
<portType name="hello_PortType">
  <operation name="hello">
    <input message="tns:helloRequest"/>
    <output message="tns:helloResponse"/>
  </operation>
</portType>
```

Web Services

Definition WSDL - Parte 3

```
<service name="hello_Service">
  <documentation>WSDL for Hello Web Service</documentation>
  <port binding="tns:hello_Binding" name="hello_Port">
    <soap:address location="http://server.net/hello/"/>
  </port>
</service>
```

Web Services

Implementacion en Java con Java EE 5, JAX-WS 2.0

```
package org; import javax.jws.WebService; import javax.jws.WebMethod; import
javax.jws.WebParam;

@WebService(serviceName = "NewWebService") public class NewWebService {
@WebMethod(operationName = "hello") public String hello(@WebParam(name = "name")
String txt) { return "Hola " + txt + "!"; } }

Aquí tienes el texto extraído de la imagen:
```

Web Services - WS-Security - WSS

El estándar WSS lidia con varias áreas modulares de seguridad, dejando muchos detalles a los llamados documentos perfil. Las áreas principales, ampliamente definidas por el estándar son:

- Maneras de agregar encabezados de seguridad (encabezados WSSE) a los sobres de SOAP
- Adjuntar testigos de seguridad y credenciales al mensaje
- Insertando un estampado de tiempo
- Firmar el mensaje
- Cifrado del mensaje
- Extensibilidad


```
<S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsse="..."> S11:Header <wsse:Security xmlns:wsse="..."> <xxx:CustomToken
wsud:id="MyID" xmlns:xxx="http://fabrikam123/token"> <FHIUORV.../>
</xxx:CustomToken> ds:Signature ds:SignedInfo <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/> <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/> <ds:Reference
URI="#MsgBody"> <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
ds:DigestValueLyLS0FPI4wPU...</ds:DigestValue> </ds:Reference> </ds:SignedInfo>
ds:SignatureValueDJbchm5gK...</ds:SignatureValue> ds:KeyInfo
wsse:SecurityTokenReference <wsse:Reference URI="#MyID"/>
</wsse:SecurityTokenReference> </ds:KeyInfo> </ds:Signature> </wsse:Security>
</S11:Header> S11:Body <tru:StockSymbol
xmlns:tru="http://fabrikam123.com/payloads"> QQQ </tru:StockSymbol> </S11:Body>
</S11:Envelope>
```

Aquí tienes el texto extraído de la imagen:

Web Services - WS-Security - WSS

WS-Policy: Describe capacidades y limitaciones de la seguridad, políticas e intermediarios (reglas de seguridad, algoritmos soportados, etc)

WS-Trust: Describe un framework para facilitar la interoperación de WS en forma segura.

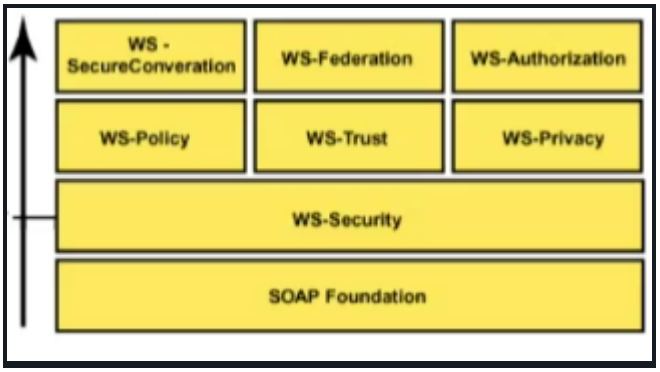
WS-Privacy: Describe el modelo sobre cómo los Web Services manejan las peticiones y preferencias de seguridad.

WS-SecureConversation: Describe como manejar y autenticar el intercambio de mensajes, el contexto de seguridad y claves de sesión.

WS-Federation: Describe cómo administrar y manejar las relaciones de confianza entre sistemas federados.

WS-Authorization: Describe cómo administrar la autorización de datos y políticas.

informativo (no profeundizand) .



Claro, aquí tienes el texto extraído de la imagen:

ReST

Es una técnica de arquitectura para sistemas distribuidos, su nombre es un acrónimo que deriva de *"Representational State Transfer"*. Su origen data del año 2000 siendo definido en una tesis doctoral de Roy Fielding.

Su objetivo fue evitar el uso de métodos complejos como CORBA, RPC y SOAP para interconexión de sistemas; con este fin las aplicaciones ReSTful usan llamados **HTTP** para las operaciones **CRUD (Create/Read/Update/Delete)** orientando su diseño a elementos en lugar de operaciones.

ReST

Es una técnica de arquitectura para sistemas distribuidos, su nombre es un acrónimo que deriva de *"Representational State Transfer"*. Su origen data del año 2000 siendo definido en una tesis doctoral de Roy Fielding.

Su objetivo fue evitar el uso de métodos complejos como CORBA, RPC y SOAP para interconexión de sistemas; con este fin las aplicaciones ReSTful usan llamados **HTTP** para las operaciones **CRUD (Create/Read/Update/Delete)** orientando su diseño a elementos en lugar de operaciones.

(despeus hay ejemplo de código que es de referencia al igual que el WADL

REFERENCIA(

WADL - Web Application Description Language

```
<resources base="http://example.com/widgets">
  <resource path="{widgetId}">
    <param name="customerId" style="query"/>
    <method name="GET">
      <request>
```

```
        <param name="verbose" style="query" type="xsd:boolean"/>
        <param name="text" style="query" type="xsd:string"/>
      </request>
      <response status="200">
        <representation mediaType="application/xml" element="yn:ResultSet"/>
      </response>
    </method>
  </resource>
</resources>
)
```

NOTA :
Se pide que en el parcial sepamos el manejo de las siglas y el conpectto ,no toma en el parcial mostrar el codigo y decir que puede pasar.

La sigueinete clase vamos a ver 2 unidades juntas ,que tiene poco contenido.

NOTA IR PREPARANDO PARA EL EXAMEN ,EN EL EXAMEN ENTRAR LA UNIDAD 3 TODO LOS ANEXOS (EL DE LA CRIPTOFGRAFIA ES EL UNICO QUE SE PUEDE OMITIR) ,4 ,5 Y 6