OWASP

Open Web Application Security Project

WHOAMI?

Oscar Martínez Ruiz de Castilla
Chalaco
Ingeniero Electrónico
Magister en Ciencias de
CISM, C)ISSO
OSCP, C|EH, C|HFI, C
C)DFE, OSEH
y Sophos Certified Eng

Especialista en Segurid
Con más de 10 años de

Network / Web applicati

oscarmrdc@gmail.com
fiery-owl.blogspot.com
@oscar_mrdc

Tu desarrollador/analista -> también defiendes!

1. Parafraseando a Bielsa
2. Qué dicen los bancos?
3. Cazadores de mitos
4. Qué dice PCI?
5. Mecanismos de Defensa
6. OWASP / ASVS / ESAPI (intro)

Cuántos trabajan atacando?

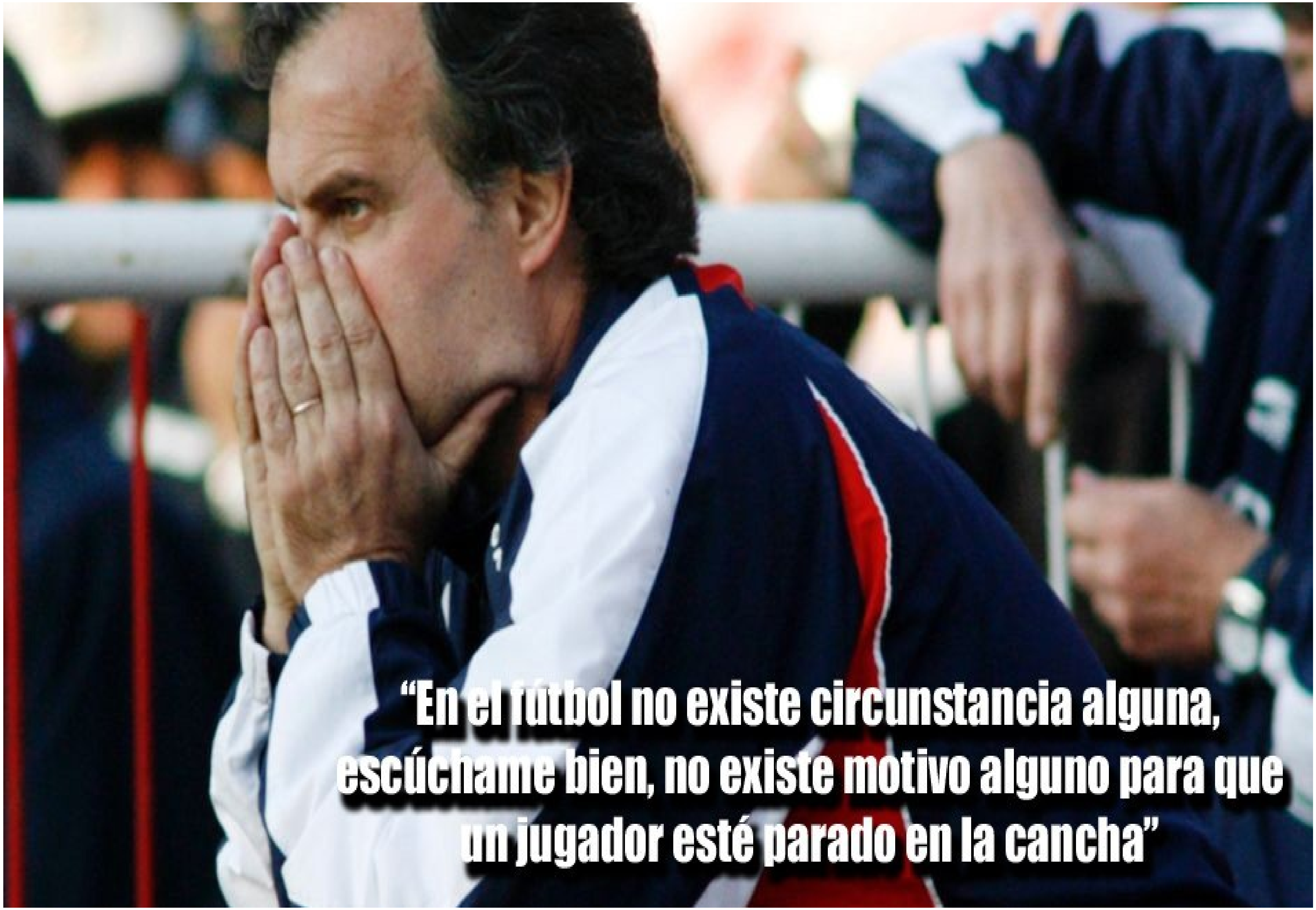
Cuántos trabajan defendiendo?

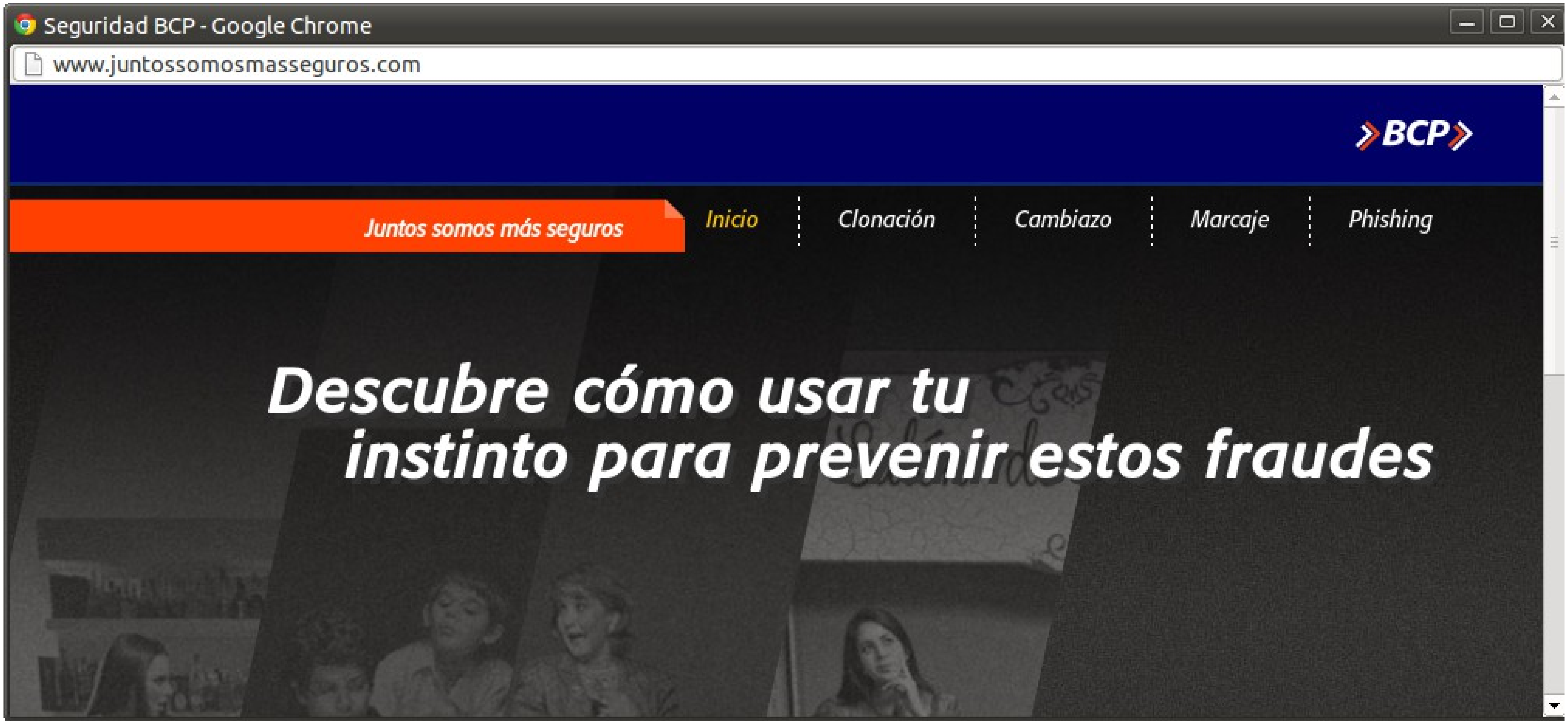
Cuántos son analistas / programadores?

Cuántos en QA?

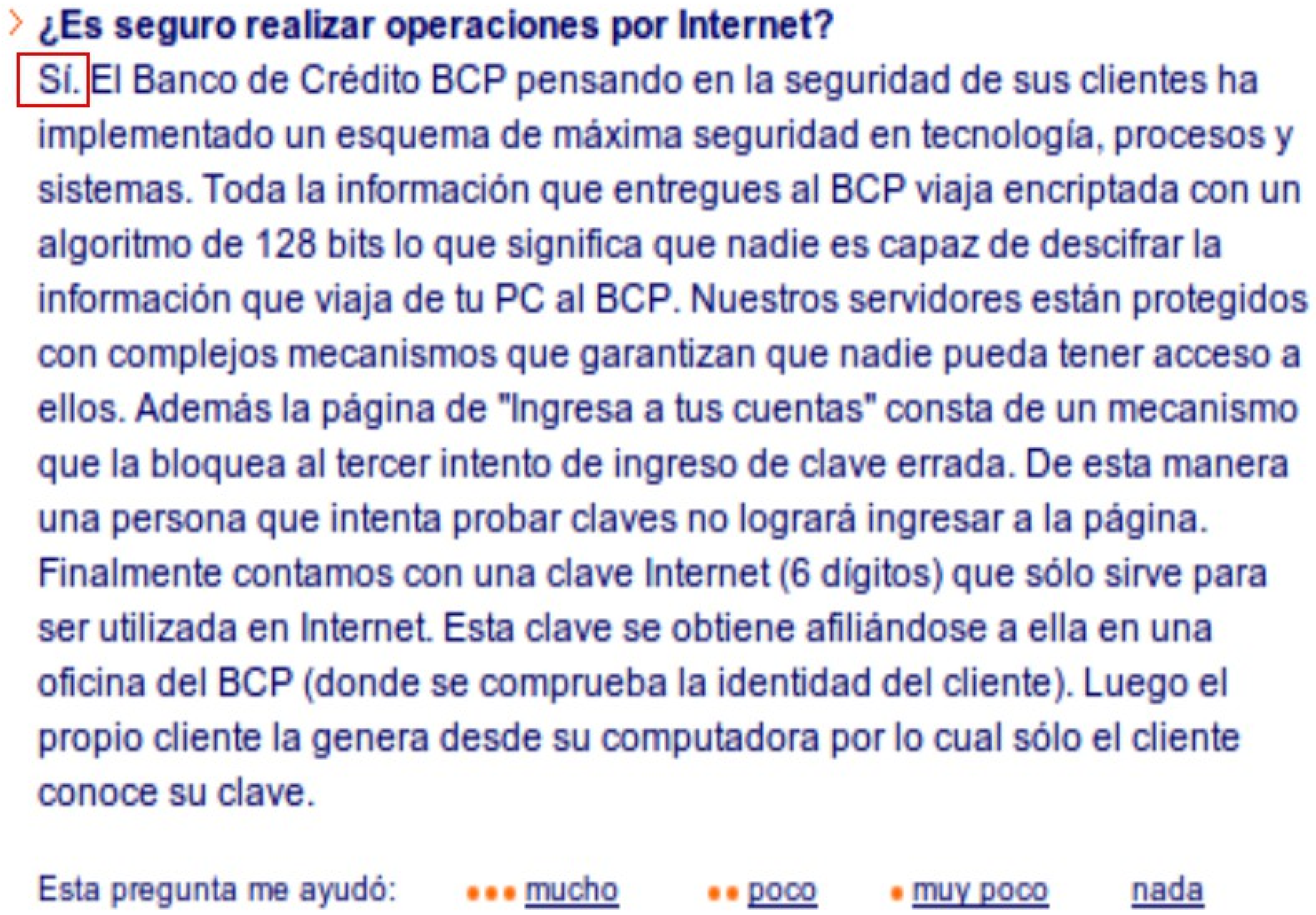
Los demás?

Y DALE
DIO DE D
Anchor
INSTANTVEA
PIEERS
CLUB ALIANZA LIMA 1901
AO
TRIATHLON
Caluca


“En el fútbol no existe circunstancia alguna,
escúchame bien, no existe motivo alguno para que
un jugador esté parado en la cancha”



Descubre cómo usar tu instinto para prevenir estos fraudes



¿Es seguro realizar operaciones por Internet?

Sí. El Banco de Crédito BCP pensando en la seguridad de sus clientes ha implementado un esquema de máxima seguridad en tecnología, procesos y sistemas. Toda la información que entregues al BCP viaja encriptada con un algoritmo de 128 bits lo que significa que nadie es capaz de descifrar la información que viaja de tu PC al BCP. Nuestros servidores están protegidos con complejos mecanismos que garantizan que nadie pueda tener acceso a ellos. Además la página de "Ingresa a tus cuentas" consta de un mecanismo que la bloquea al tercer intento de ingreso de clave errada. De esta manera una persona que intenta probar claves no logrará ingresar a la página. Finalmente contamos con una clave Internet (6 dígitos) que sólo sirve para ser utilizada en Internet. Esta clave se obtiene afiliándose a ella en una oficina del BCP (donde se comprueba la identidad del cliente). Luego el propio cliente la genera desde su computadora por lo cual sólo el cliente conoce su clave.

Acceso seguro

Firewall

› Son dispositivos o programas con los que cuenta Interbank para protegerse de ataques o accesos no permitidos a nuestra Banca por Internet. Para lograr ello, Interbank define reglas de acceso en base a un conjunto de normas y políticas de seguridad de información.

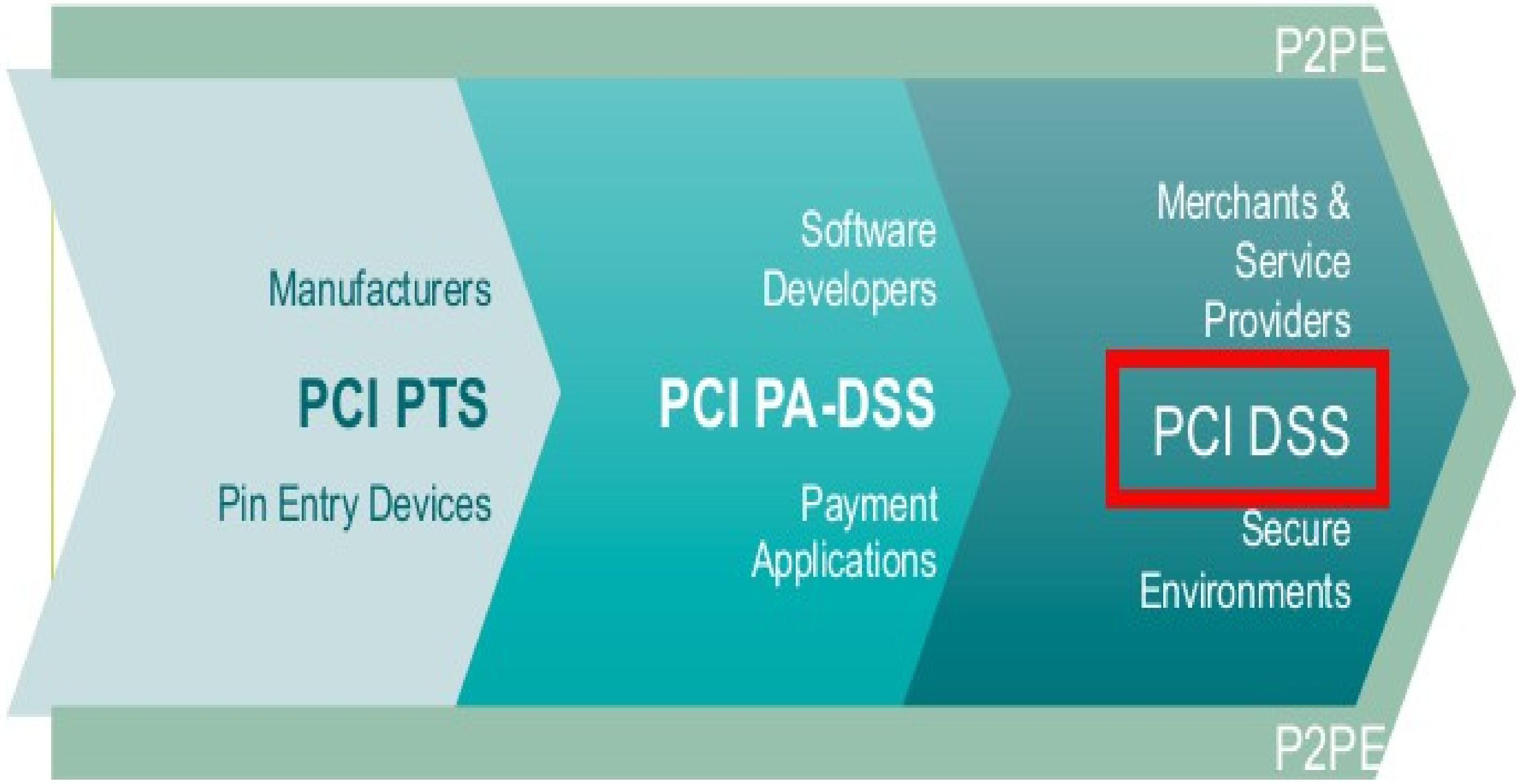
Transmisión Segura de Datos Encriptados

MYTHBUSTERS

“El cumplimiento de PCI / NTP27001 es un proyecto para TI”

“Soy seguro porque cumplo PCI / NTP 27001”

PCI compliance is a business issue, not a technology issue. There is no single technology solution that will make your organization PCI compliant. Because it is a business issue that affects the entire organization, PCI compliance calls for a multidisciplinary team including at least Finance, IT, and likely Internal Audit.


P2PE
Manufacturers
PCI PTS
Pin Entry Devices
Software Developers
PCI PA-DSS
Payment Applications
Merchants & Service Providers
PCI DSS
Secure Environments
P2PE

PCI Data Security Standard – High Level Overview

Build and Maintain a Secure Network and Systems	**1.** Install and maintain a firewall configuration to protect cardholder data **2.** Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	**3.** Protect stored cardholder data **4.** Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	**5.** Protect all systems against malware and regularly update anti-virus software or programs **6.** Develop and maintain secure systems and applications
Implement Strong Access Control Measures	**7.** Restrict access to cardholder data by business need to know **8.** Identify and authenticate access to system components **9.** Restrict physical access to cardholder data
Regularly Monitor and Test Networks	**10.** Track and monitor all access to network resources and cardholder data **11.** Regularly test security systems and processes
Maintain an Information Security Policy	**12.** Maintain a policy that addresses information security for all personnel

Merchant Level Determines Validation

Level	Visa and MasterCard	Amex
1 >6 Million trans/yr, by brand	• Annual on-site assessment • Quarterly network scan by Approved Scanning Vendor (ASV) • Report on Compliance (ROC)	• Annual on-site Security Audit • Quarterly network scan by ASV
2 >1 Million trans/yr, by brand	Visa: • Annual Self-Assessment Questionnaire (SAQ) • Quarterly network scan by ASV MasterCard: • Same as Level 1	• Quarterly network scan ASV
3 >20K ecommerce	• Annual SAQ • Quarterly network scan by ASV	• Recommend quarterly network scan by ASV
4	Determined by acquirer: • Annual SAQ • Quarterly network scan by ASV	

		Data Element	Storage Permitted	Render Stored Data Unreadable per Requirement 3.4
Account Data	Cardholder Data	Primary Account Number (PAN)	Yes	Yes
		Cardholder Name	Yes	No
		Service Code	Yes	No
		Expiration Date	Yes	No
	Sensitive Authentication Data[2]	Full Track Data[3]	No	Cannot store per Requirement 3.2
		CAV2/CVC2/CVV2/CID[4]	No	Cannot store per Requirement 3.2
		PIN/PIN Block[5]	No	Cannot store per Requirement 3.2

6.3 Develop internal and external software applications (including web-based administrative access to applications) securely, as follows: • In accordance with PCI DSS (for example, secure authentication and logging) • Based on industry standards and/or best practices. • Incorporating information security throughout the software-development life cycle ***Note**: this applies to all software developed internally as well as bespoke or custom software developed by a third party.*	**6.3.a** Examine written software-development processes to verify that the processes are based on industry standards and/or best practices.	Without the inclusion of security during the requirements definition, design, analysis, and testing phases of software development, security vulnerabilities can be inadvertently or maliciously introduced into the production environment. Understanding how sensitive data is handled by the application—including when stored, transmitted, and when in memory—can help identify where data needs to be protected.
	6.3.b Examine written software-development processes to verify that information security is included throughout the life cycle.	
	6.3.c Examine written software-development processes to verify that software applications are developed in accordance with PCI DSS.	
	6.3.d Interview software developers to verify that written software-development processes are implemented.	

PCI DSS Requirements	Testing Procedures
6.5 Address common coding vulnerabilities in software-development processes as follows: • Train developers in secure coding techniques, including how to avoid common coding vulnerabilities, and understanding how sensitive data is handled in memory. • Develop applications based on secure coding guidelines. ***Note:** The vulnerabilities listed at 6.5.1 through 6.5.10 were current with industry best practices when this version of PCI DSS was published. However, as industry best practices for vulnerability management are updated (for example, the OWASP Guide, SANS CWE Top 25, CERT Secure Coding, etc.), the current best practices must be used for these requirements.*	**6.5.a** Examine software-development policies and procedures to verify that training in secure coding techniques is required for developers, based on industry best practices and guidance.
	6.5.b Interview a sample of developers to verify that they are knowledgeable in secure coding techniques.
	6.5.c Examine records of training to verify that software developers received training on secure coding techniques, including how to avoid common coding vulnerabilities, and understanding how sensitive data is handled in memory.
	6.5.d. Verify that processes are in place to protect applications from, at a minimum, the following vulnerabilities:

6.5.1 Injection flaws, particularly SQL injection. Also consider OS Command Injection, LDAP and XPath injection flaws as well as other injection flaws.

6.5.2 Buffer overflows

6.5.3 Insecure cryptographic storage

6.5.4 Insecure communications

6.5.5 Improper error handling

6.5.7 Cross-site scripting (XSS)

6.5.8 Improper access control (such as insecure direct object references, failure to restrict URL access, directory traversal, and failure to restrict user access to functions).

6.5.9 Cross-site request forgery (CSRF)

6.5.10 Broken authentication and session management

6.5.10 Broken authentication and session Management

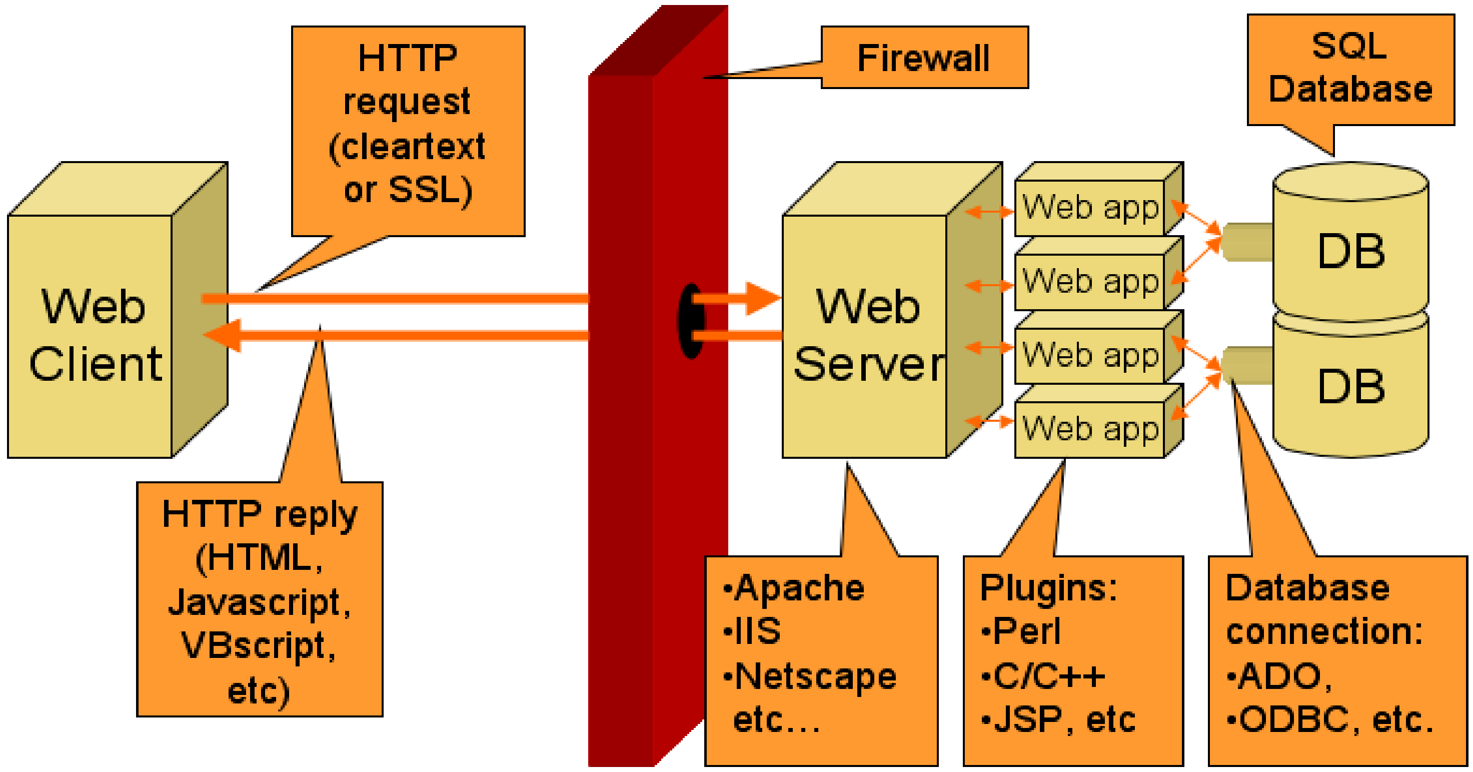
Note: Requirement 6.5.10 is a best practice until June 30, 2015, after which it becomes a requirement.

PERUFAIL.COM

La Realidad en Perú?

Muchos con SQLi, XSS, credenciales débiles, etc.

Vulnerabilidades en el propio software, desarrollado por la propia empresa o un tercero (Lógica y código fuente)



No esta relacionada necesariamente con la plataforma (puertos 80 y 443)

Mecanismos de Defensa

Gestionar el acceso del usuario (a las funcionalidades y datos)

Gestionar los datos ingresados por el usuario

Gestionar los ataques (medidas defensivas y ofensivas)

Gestionar el acceso del usuario (a las funcionalidades y datos)

Autenticación
(formularios web, certificados, tokens, etc)
Login, pero también: recuperación de cuenta, cambio de contraseña, auto registro, etc.

Manejo de sesiones
Http no es orientado a la conexión
Tokens de sesión, campos de formulario ocultos, etc.
Timeout.

Control de accesos
Decidir si el usuario esta autorizado para usar un recurso

El mecanismo es tan fuerte como el más débil de sus componentes

Gestionar los datos ingresados por el usuario

Variedad de datos: Nombres, edades, fechas, etc.
Tipo
Longitud
Expresiones regulares

Listas negras

Listas blancas

Gestionar los ataques (medidas defensivas y ofensivas)

Igual ocurrirán errores -> anticiparlos:

Manejar errores

Try-catch, errores genéricos

Mantener logs de auditoría

Para entender que pasó (requerimientos de seguridad / diseño)

Alertas a los administradores

Para tomar una acción inmediata y no esperar a revisar los logs

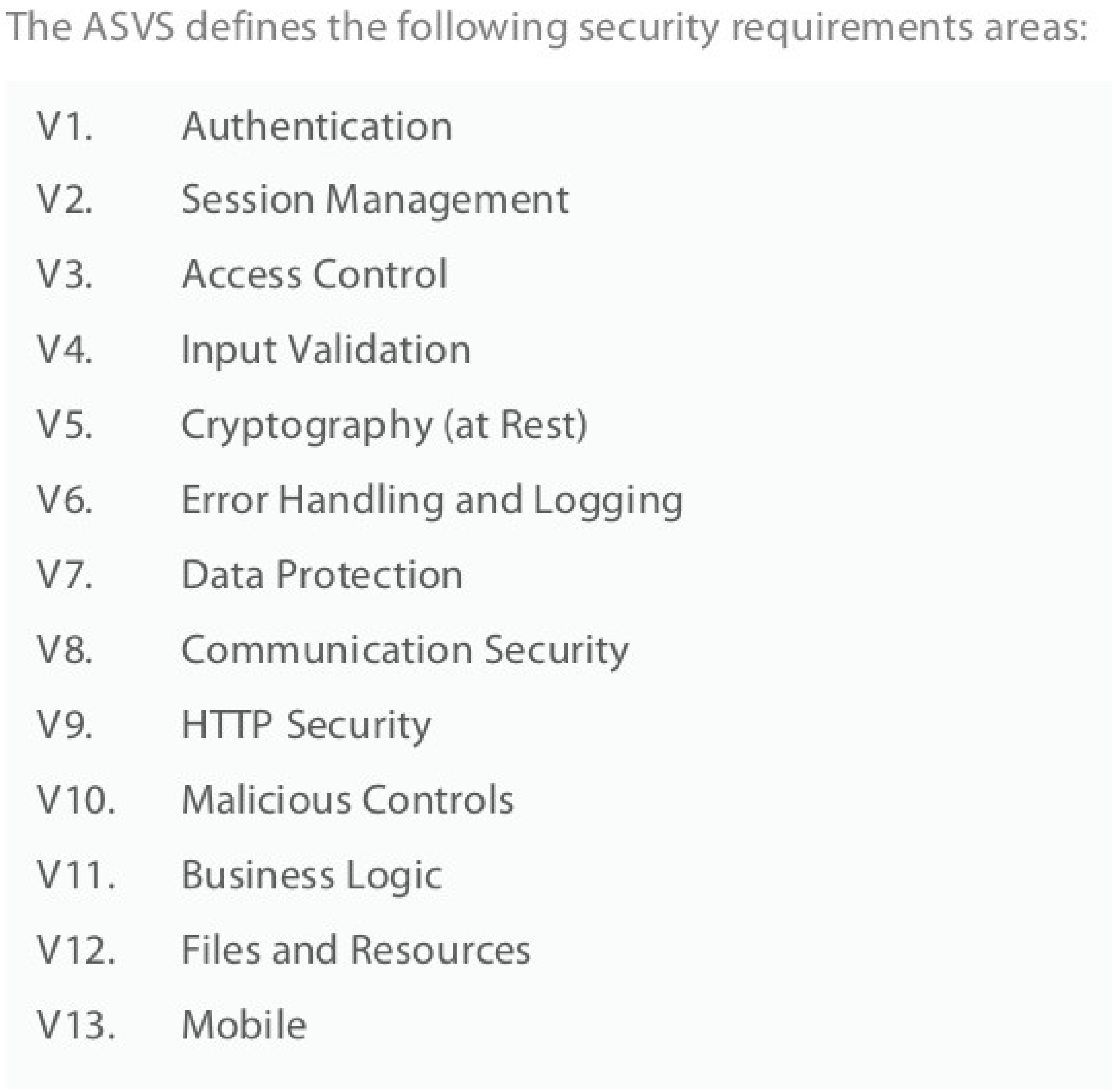
Reaccionar a los ataques

Terminar la sesión, bloquear al usuario, etc.

Requerimientos de seguridad en Aplicaciones Web (OWASP ASVS)

Application Security Verification Standard

ASVS puede ser utilizado para establecer un nivel de confianza en la seguridad de aplicaciones web



The ASVS defines the following security requirements areas:

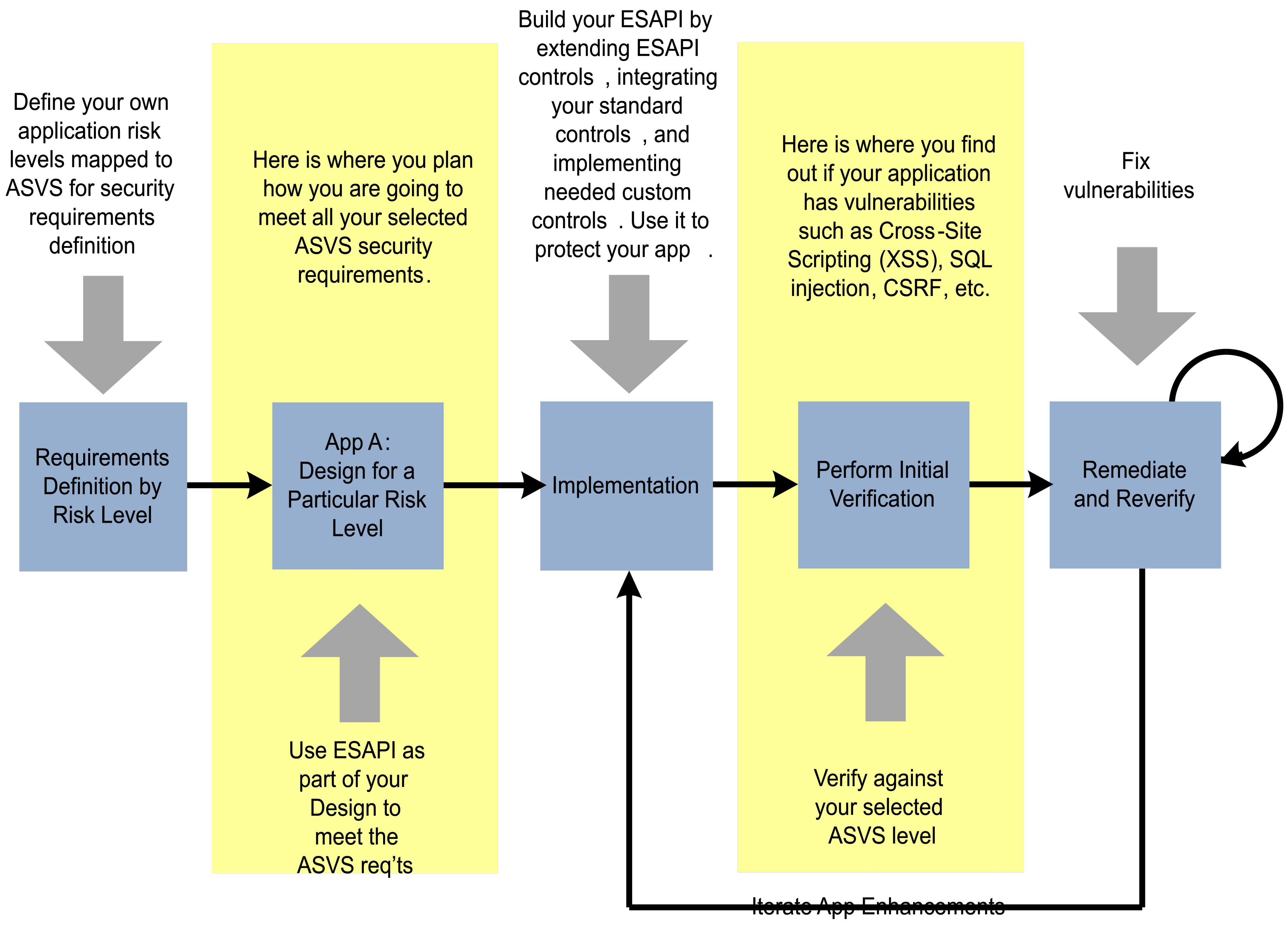
- V1. Authentication
- V2. Session Management
- V3. Access Control
- V4. Input Validation
- V5. Cryptography (at Rest)
- V6. Error Handling and Logging
- V7. Data Protection
- V8. Communication Security
- V9. HTTP Security
- V10. Malicious Controls
- V11. Business Logic
- V12. Files and Resources
- V13. Mobile

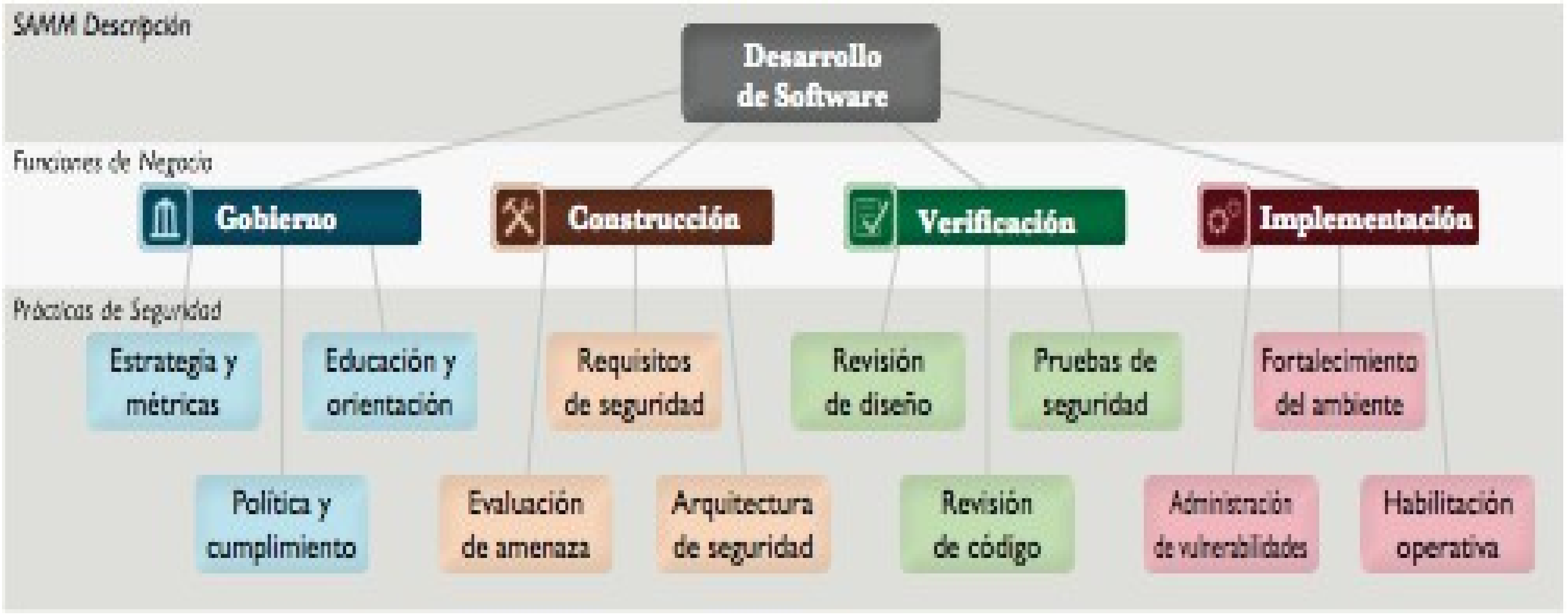
	AUTHENTICATION VERIFICATION REQUREMENT	LEVELS		
		1	2	3
V1.1	Verify all pages and resources require authentication except those specifically intended to be public (Principle of complete mediation).	✓	✓	✓

V9: HTTP Security Verification Requirements

The table below defines the corresponding verification requirements that apply for each of the verification levels. Verification requirements for Level 0 are not defined by this standard.

	HTTP SECURITY VERIFICATION REQUREMENT	LEVELS		
		1	2	3
V9.1	Verify that the application accepts only a defined set of HTTP request methods, such as GET and POST and unused methods are explicitly blocked.	✓	✓	✓
V9.2	Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8).	✓	✓	✓
V9.3	Verify that HTTP headers and / or other mechanisms for older browsers have been included to protect against click jacking attacks	✓	✓	✓
V9.4	Verify that HTTP headers in both requests and responses contain only printable ASCII characters.		✓	✓


Define your own application risk levels mapped to ASVS for security requirements definition
Here is where you plan how you are going to meet all your selected ASVS security requirements.
Build your ESAPI by extending ESAPI controls, integrating your standard controls, and implementing needed custom controls. Use it to protect your app.
Here is where you find out if your application has vulnerabilities such as Cross-Site Scripting (XSS), SQL injection, CSRF, etc.
Fix vulnerabilities
Requirements Definition by Risk Level
App A: Design for a Particular Risk Level
Implementation
Perform Initial Verification
Remediate and Reverify
Use ESAPI as part of your Design to meet the ASVS req'ts
Verify against your selected ASVS level
Iterate App Enhancements


SAMM Descripción
Desarrollo de Software
Funciones de Negocio
Gobierno
Construcción
Verificación
Implementación
Prácticas de Seguridad
Estrategia y métricas
Política y cumplimiento
Educación y orientación
Evaluación de amenaza
Requisitos de seguridad
Arquitectura de seguridad
Revisión de diseño
Revisión de código
Pruebas de seguridad
Administración de vulnerabilidades
Fortalecimiento del ambiente
Habilitación operativa

Beneficios de implementar seguridad en aplicaciones?

- ✓ Reduce costos de desarrollo, recuperación ante incidentes y parches.
- ✓ Reduce costo de testeo de seguridad de terceros.

Validar longitud, tipo, etc:

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

String code= request.getParameter(“code”);

String codevalid="";
Pattern pat = Pattern.compile("[0-9]{1,2}");
Matcher mat = pat.matcher(code);
if (mat.matches()) {
    codevalid=code;
} else {
    codevalid="";
    //response.sendRedirect("office2.jsp");
}
```

SQLi

Usar sentencias SQL precompiladas:

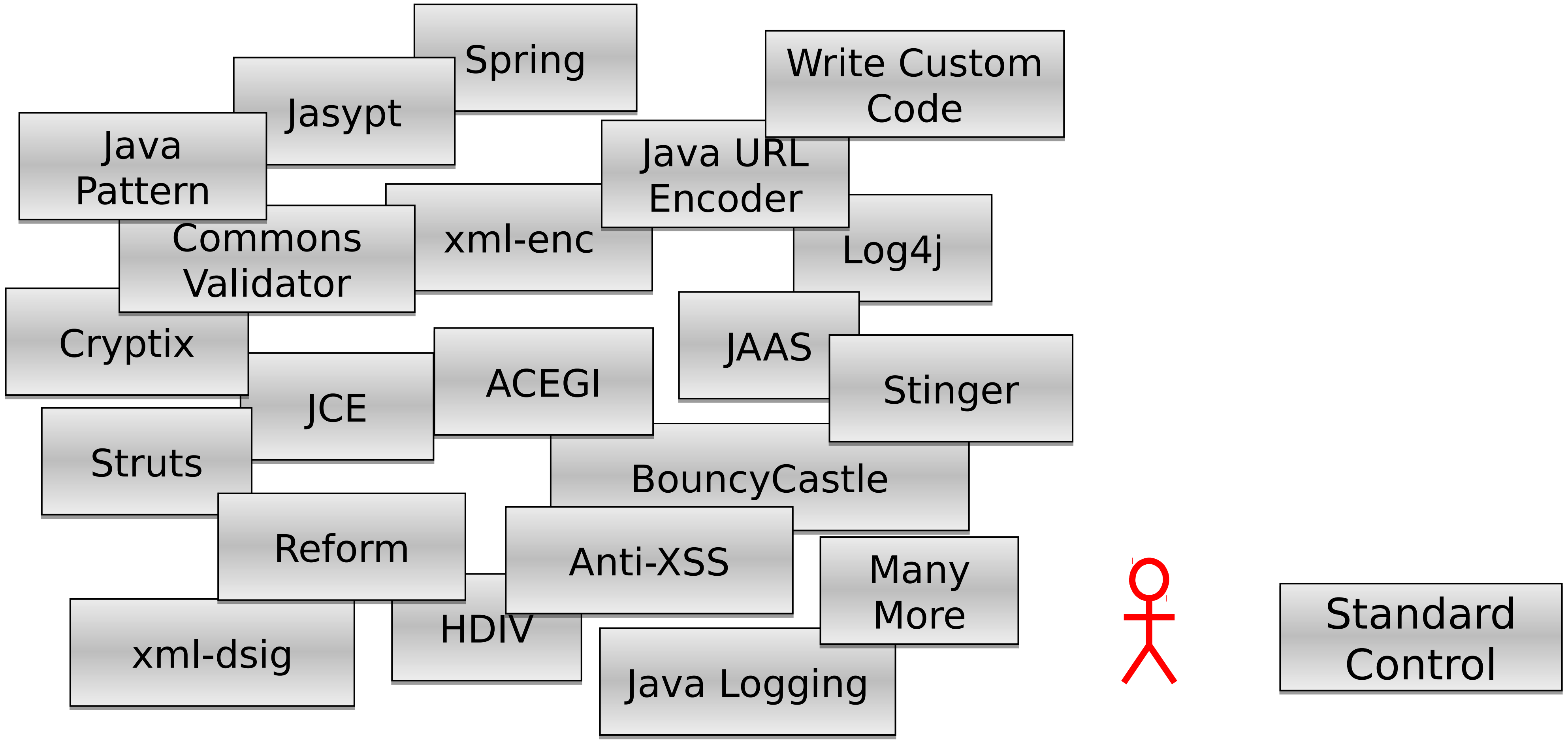
```
PreparedStatement pstmt = con.prepareStatement("update
empleado set sueldo = ? where id_empleado = ?");
pstmt.setDouble(1, 153833.00);
pstmt.setInt(2, 110592);
```

Codificar datos de salida:

XSS

Antes: <script>alert(1)</script>

Después: &lt;script&gt;alert&#x28;1&#x29;&lt;&#x2f;script&gt;



NO Intuitivo, Integrado o Amigable (para el desarrollador).

Según las buenas prácticas en el desarrollo seguro de aplicaciones, se recomienda el uso de librerías ó APIs como ESAPI (Enterprise Security API - OWASP) la cual implementa una biblioteca de controles que facilita a los programadores a escribir aplicaciones web de menor riesgo.

Las bibliotecas ESAPI están diseñadas para facilitar a los programadores, adaptar la seguridad en las aplicaciones web existentes.

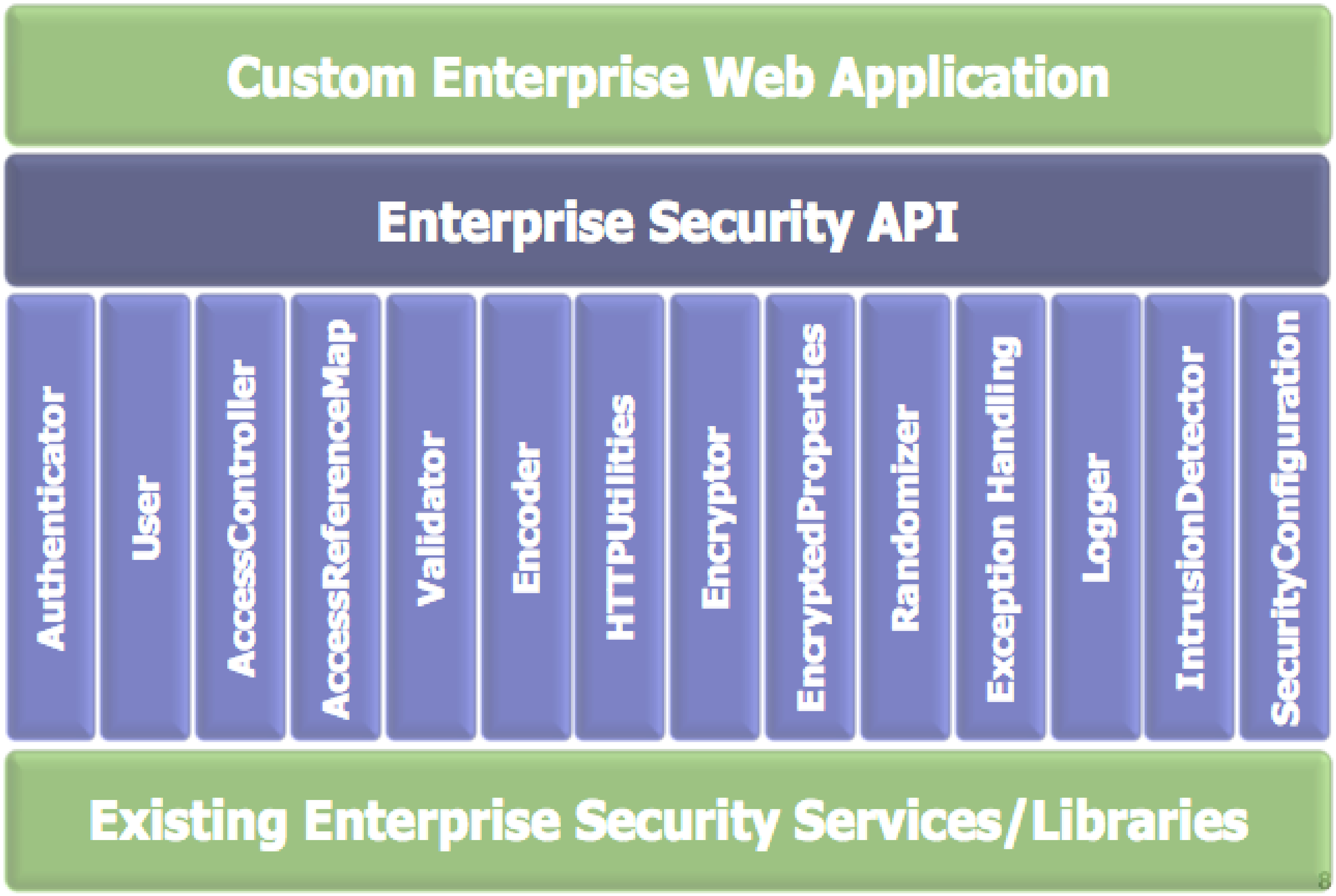
Actualmente la versión para Java EE se encuentra en la versión 2.1.0 de Setiembre de 2013.

Referencias:

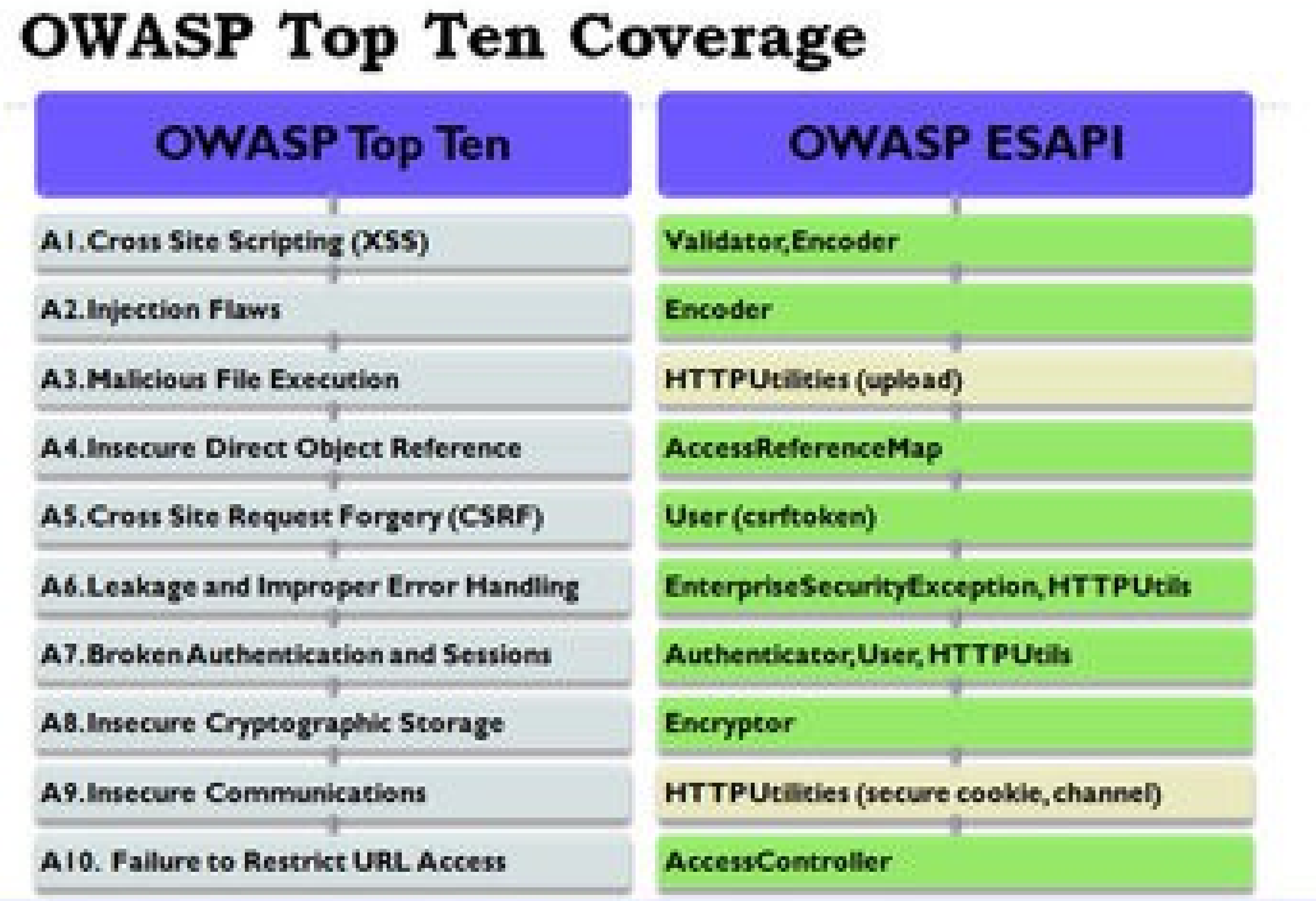
https://www.owasp.org/index.php/Esapi#tab=Java_EE

https://code.google.com/p/owasp-esapi-java/

Implementación de Controles

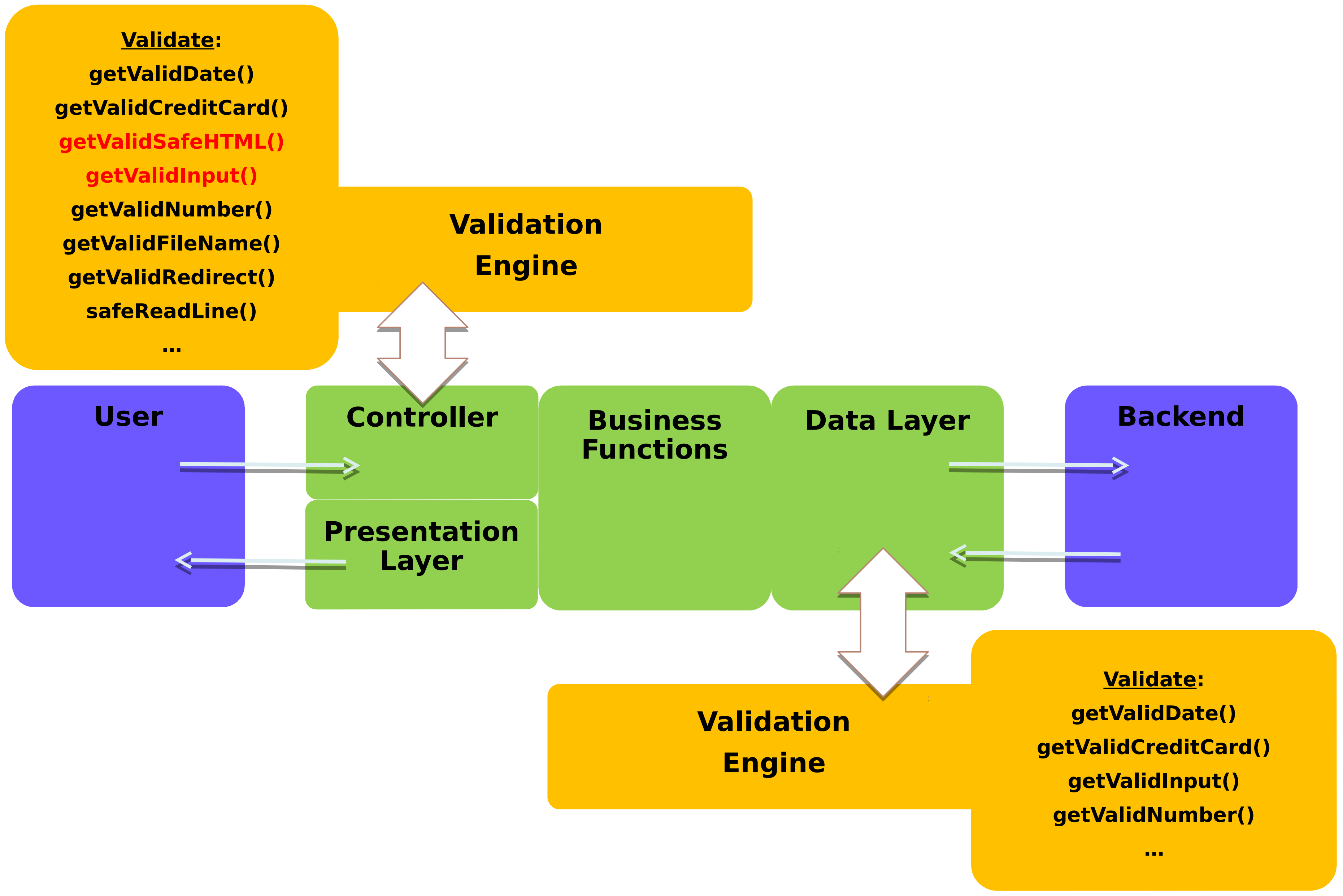


Implementación de Controles



OWASP Top Ten Coverage

OWASP Top Ten	OWASP ESAPI
A1.Cross Site Scripting (XSS)	Validator,Encoder
A2.Injection Flaws	Encoder
A3.Malicious File Execution	HTTPUtilities (upload)
A4.Insecure Direct Object Reference	AccessReferenceMap
A5.Cross Site Request Forgery (CSRF)	User (csrftoken)
A6.Leakage and Improper Error Handling	EnterpriseSecurityException, HTTPUtils
A7.Broken Authentication and Sessions	Authenticator,User, HTTPUtils
A8.Insecure Cryptographic Storage	Encryptor
A9.Insecure Communications	HTTPUtilities (secure cookie, channel)
A10. Failure to Restrict URL Access	AccessController


Validate:
getValidDate()
getValidCreditCard()
getValidSafeHTML()
getValidInput()
getValidNumber()
getValidFileName()
getValidRedirect()
safeReadLine()
...
Validation Engine
User
Controller
Presentation Layer
Business Functions
Data Layer
Backend
Validation Engine
Validate:
getValidDate()
getValidCreditCard()
getValidInput()
getValidNumber()
...

getValidInput

```
java.lang.String getValidInput(java.lang.String context,
                               java.lang.String input,
                               java.lang.String type,
                               int maxLength,
                               boolean allowNull)

                throws ValidationException,
                       IntrusionException
```

Returns canonicalized and validated input as a String. Invalid input will generate a descriptive ValidationException, and input that is clearly an attack will generate a descriptive IntrusionException.

Parameters:

context - A descriptive name of the parameter that you are validating (e.g., LoginPage_UsernameField). This value is used by any logging or error handling that is done with respect to the value passed in.

input - The actual user input data to validate.

type - The regular expression name that maps to the actual regular expression from "ESAPI.properties".

maxLength - The maximum post-canonicalized String length allowed.

allowNull - If allowNull is true then an input that is NULL or an empty string will be legal. If allowNull is false then NULL or an empty String will throw a ValidationException.

Returns:

The canonicalized user input.

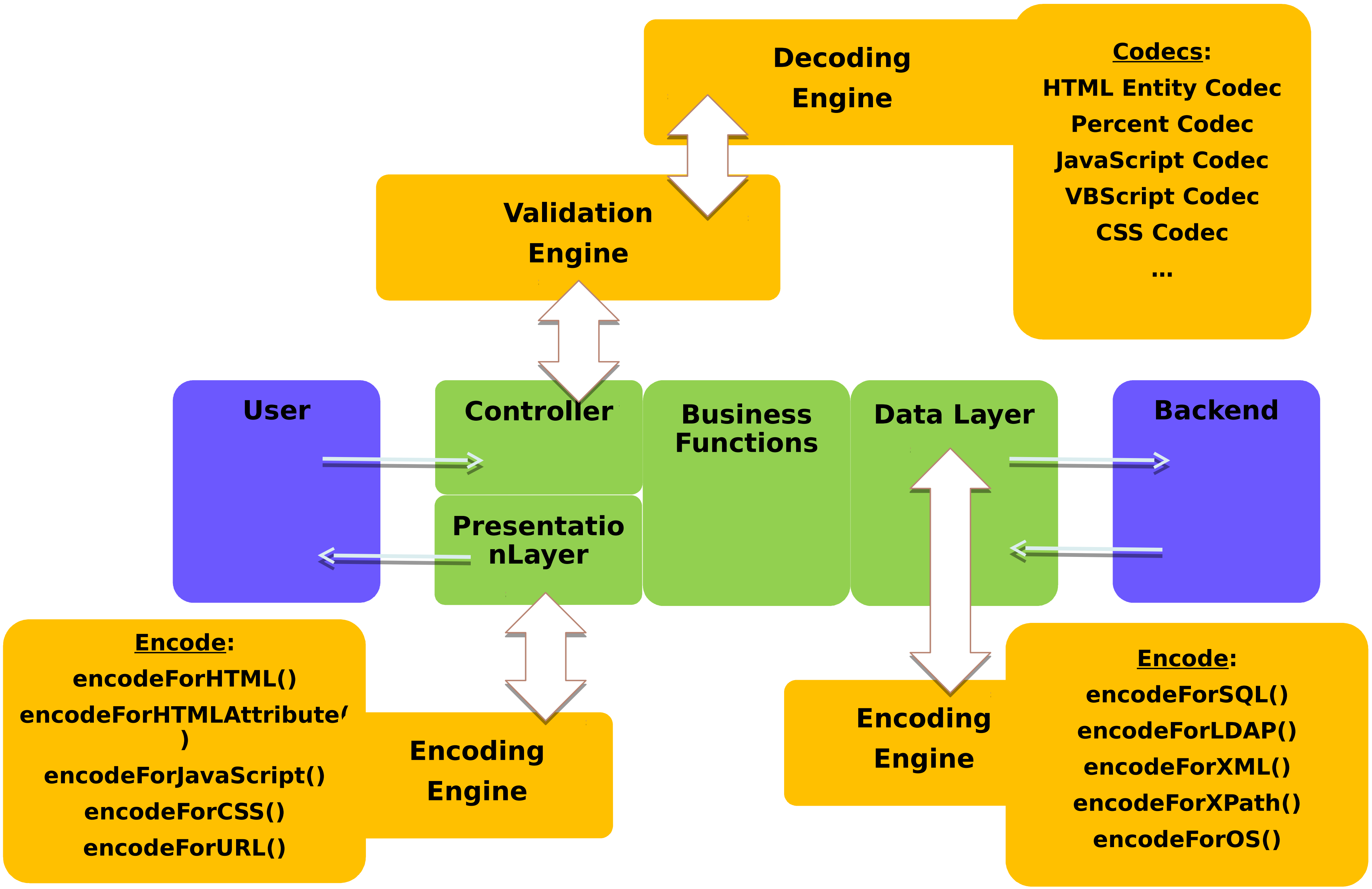
Validator:

Para validar los datos de entrada ingresados por el usuario:

String validatedFirstName =
ESAPI.validator().getValidInput("FirstName",
myForm.getFirstName(), "FirstNameRegex", 255, false);

String cleanComment =
ESAPI.validator().getValidInput("comment",
request.getParameter("comment"), "CommentRegex", 300, false);

validation rules in the .esapi\validation.properties file



Decoding Engine
Codecs:
HTML Entity Codec
Percent Codec
JavaScript Codec
VBScript Codec
CSS Codec
...
Validation Engine
User
Controller
Business Functions
Data Layer
Backend
Presentatio nLayer
Encode:
encodeForHTML()
encodeForHTMLAttribute()
encodeForJavaScript()
encodeForCSS()
encodeForURL()
Encoding Engine
Encoding Engine
Encode:
encodeForSQL()
encodeForLDAP()
encodeForXML()
encodeForXPath()
encodeForOS()

encodeForHTML

```
java.lang.String encodeForHTML(java.lang.String input)
```

Encode data for use in HTML using HTML entity encoding
Note that the following characters: 00-08, 0B-0C, 0E-1F, and 7F-9F cannot be used in HTML.

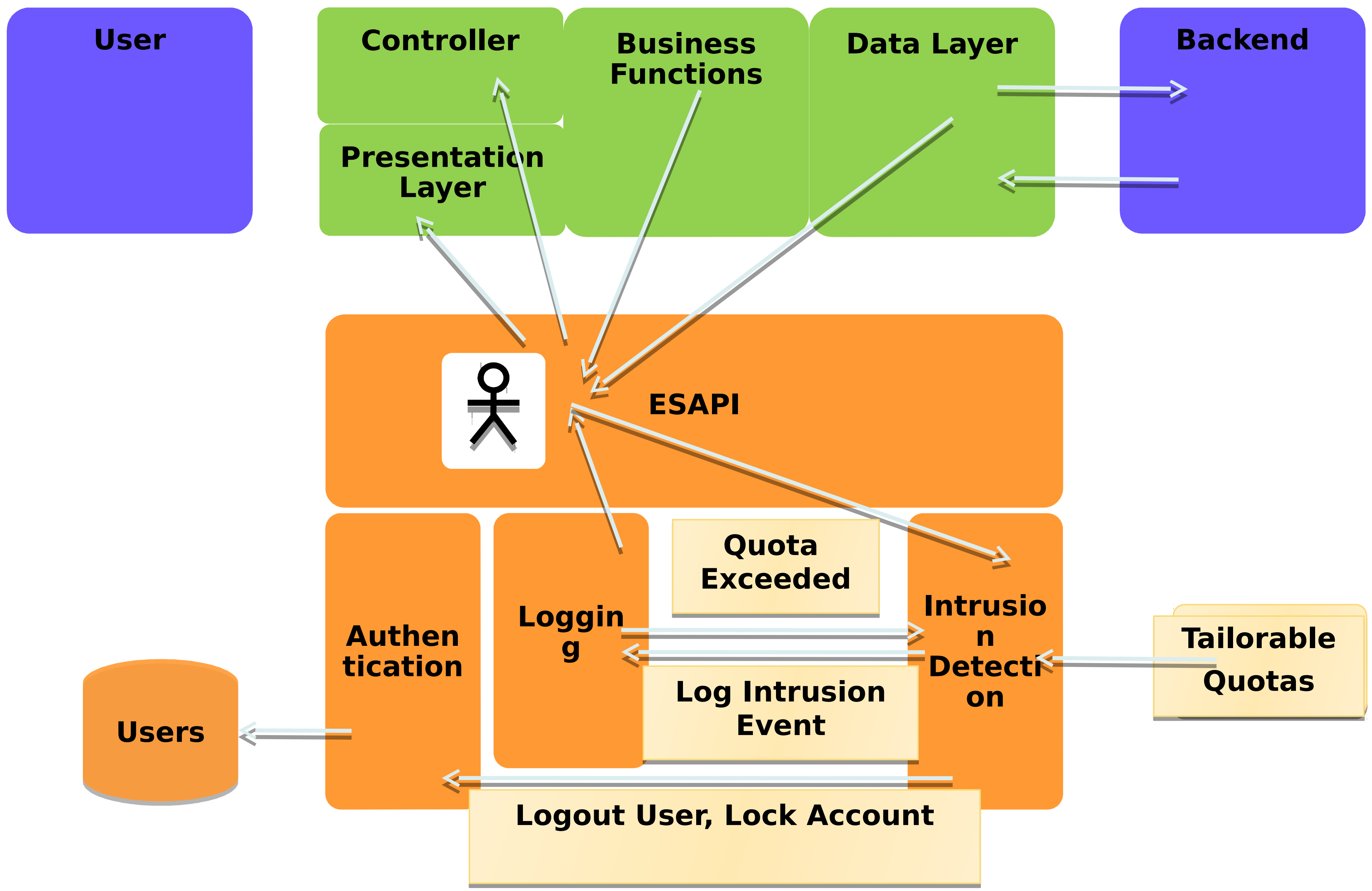
Parameters:
input - the text to encode for HTML

Returns:
input encoded for HTML

Encoder:

Para codificar los datos de salida:

```
String safeOutput =
ESAPI.encoder().encodeForHTML( cleanComment );
```


User
Controller
Business Functions
Data Layer
Backend
Presentation Layer
ESAPI
Quota Exceeded
Authen tication
Loggin g
Intrusio n Detecti on
Tailorable Quotas
Log Intrusion Event
Users
Logout User, Lock Account

EnterpriseSecurityException is the base class for all security related exceptions.

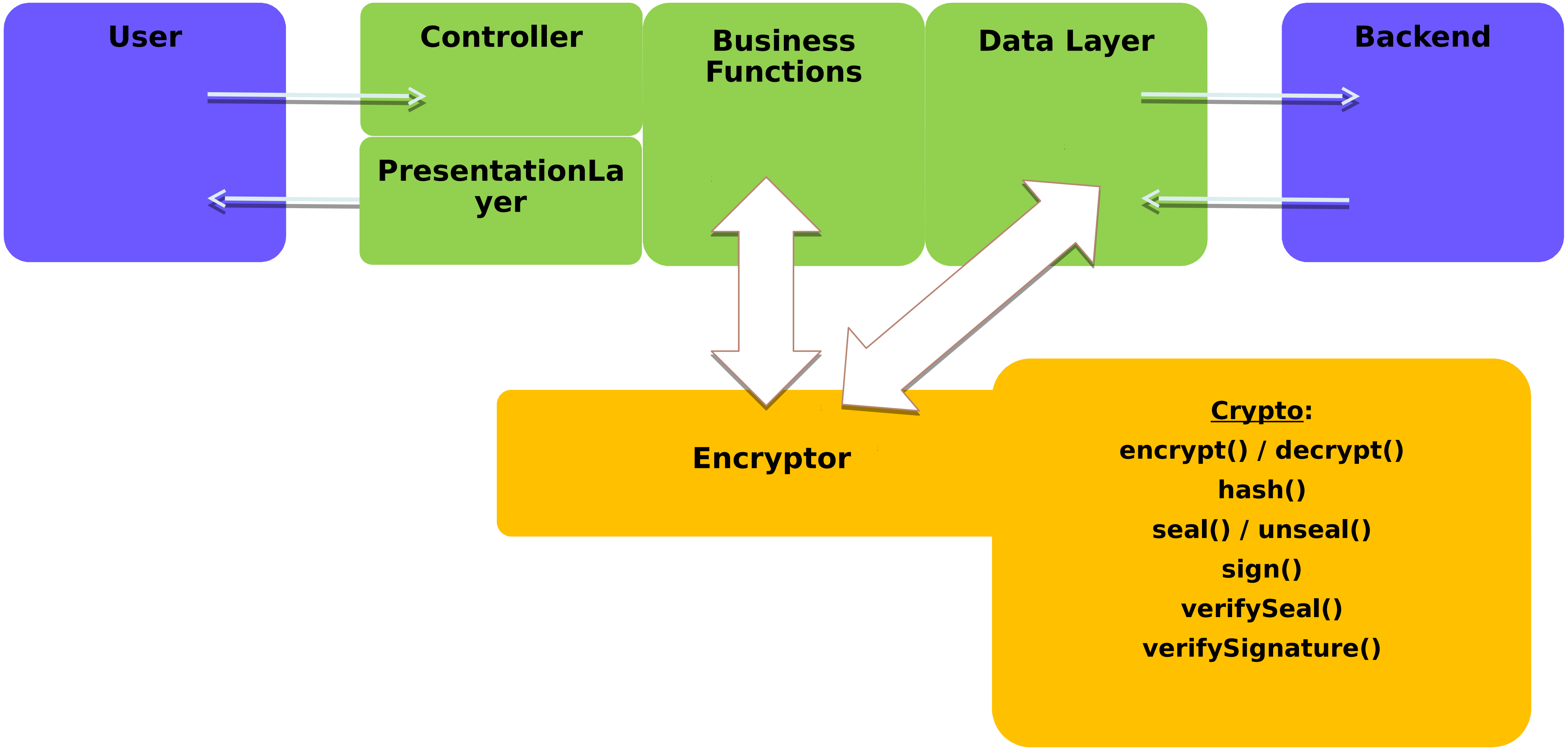
All EnterpriseSecurityExceptions have two messages, one for the user and one for the log file.

Method Summary

getLogMessage()
Returns a message that is safe to display in logs, but probably not to users

getUserMessage()
Returns message meant for display to users Note that if you are unsure of what set this message, it would probably be a good idea to encode this message before displaying it to the end user.

Codificar también los datos enviados a los logs!


User
Controller
PresentationLayer
Business Functions
Data Layer
Backend
Encryptor
Crypto:
encrypt() / decrypt()
hash()
seal() / unseal()
sign()
verifySeal()
verifySignature()

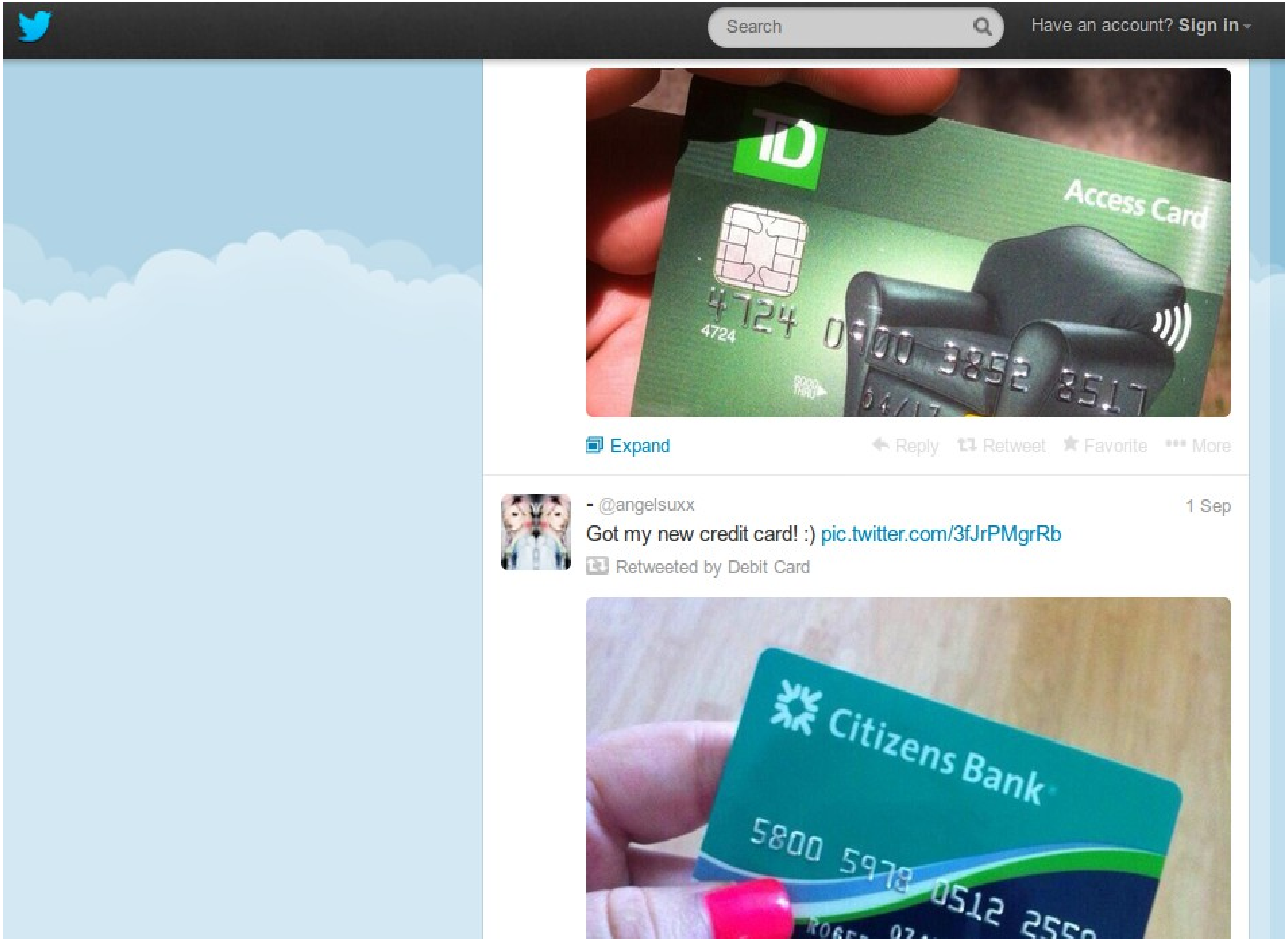
encrypt

```
CipherText encrypt(PlainText plaintext)
                   throws EncryptionException
```

Encrypts the provided plaintext bytes using the cipher transformation specified by the property Encryptor.CipherTransformation and the master encryption key as specified by the property Encryptor.MasterKey as defined in the ESAPI.properties file.
This method is preferred over encrypt(String) because it also allows encrypting of general byte streams rather than simply strings and also because it returns a CipherText object and thus supports cipher modes that require an Initialization Vector (IV), such as Cipher Block Chaining (CBC).

Parameters:
plaintext - The PlainText to be encrypted.

Returns:
the CipherText object from which the raw ciphertext, the IV, the cipher transformation, and many other aspects about the encryption detail may be extracted.



- @angelsuxx 1 Sep

Got my new credit card! :) pic.twitter.com/3fJrPMgrRb

Retweeted by Debit Card