

SecureTX

**Aprovechando OWASP
para certificar una aplicación de pagos**

Álvaro Rodríguez

@alvrod

PayTrue



PayTrue

- Desarrolla soluciones para la industria de medios de pago (PCI) desde 2003
- Sistemas integrales de procesamiento de tarjetas (crédito, débito, prepago)
- Sistemas de prevención / detección de fraude
- Emisores, Adquirentes, Procesadores
- Bancos o grandes cadenas de retail





Contexto

PCI standards

Tokenización

Payment Card Industry

- MasterCard, VISA, American Express
- Esquema descentralizado de cobertura mundial
- Iniciativas para nuevos negocios, control de fraude
 - Chip & PIN (EMV), “Verified By Visa”, etc.
- **PCI Council**
 - PCI DSS
 - Data Security Standard
 - Aplicable a ambientes productivos
 - PA DSS
 - Aplicable a aplicaciones de pagos



Vida de una transacción



PCI DSS

Estándares de seguridad para proteger información sensible de tarjetas y prevenir fraude

Obligatorios para proveedores de servicio, comercios, etc.

Apuntan a impedir el acceso pero también a desvalorizar la información



PCI DSS

PCI Data Security Standard – High Level Overview

Build and Maintain a Secure Network	<ol style="list-style-type: none">1. Install and maintain a firewall configuration to protect cardholder data2. Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	<ol style="list-style-type: none">3. Protect stored cardholder data4. Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	<ol style="list-style-type: none">5. Use and regularly update anti-virus software or programs6. Develop and maintain secure systems and applications
Implement Strong Access Control Measures	<ol style="list-style-type: none">7. Restrict access to cardholder data by business need to know8. Assign a unique ID to each person with computer access9. Restrict physical access to cardholder data
Regularly Monitor and Test Networks	<ol style="list-style-type: none">10. Track and monitor all access to network resources and cardholder data11. Regularly test security systems and processes.
Maintain an Information Security Policy	<ol style="list-style-type: none">12. Maintain a policy that addresses information security for all personnel.



Datos sensibles (fraude)

		Data Element	Storage Permitted	Render Stored Account 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 ¹	Full Magnetic Stripe Data ²	No	Cannot store per Requirement 3.2
		CAV2/CVC2/CVV2/CID	No	Cannot store per Requirement 3.2
		PIN/PIN Block	No	Cannot store per Requirement 3.2



Situación

- N° de tarjeta, tracks, PIN, en claro, “por todos lados”

CRM, contabilidad, facturación, etc. etc.

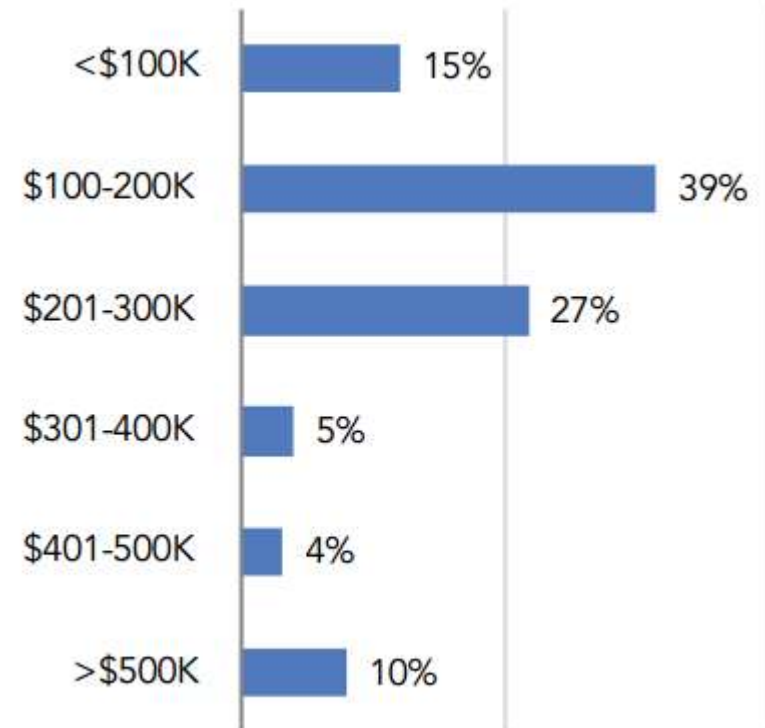
- Sistemas no preparados para cumplir con la norma PCI DSS
 - Enorme costo de adaptación
 - La norma en sí es cara y difícil de cumplir



Costo

El estándar tiene más de 260 requisitos individuales, afectando desde políticas de RRHH hasta el control de qué servicios se ejecutan en cada computadora dentro del “ambiente de tarjetas PCI”

- Análisis GAP: 2 meses
 - Proyecto PCI: 7 meses
 - Certificación / ajustes: 2 meses
 - ~2 millones U\$S
- +
- 200.000 – 500.000+ U\$S / año en auditorías



QSA

- Encargados de realizar auditorías y certificaciones
- 318 a nivel global, para LAC hay registrados 37, los que hemos encontrado más son
 - 403 Labs
 - Trust Wave



Dificultad técnica

Cada módulo de software que deba certificarse tiene un gran sobrecosto por los requisitos técnicos.

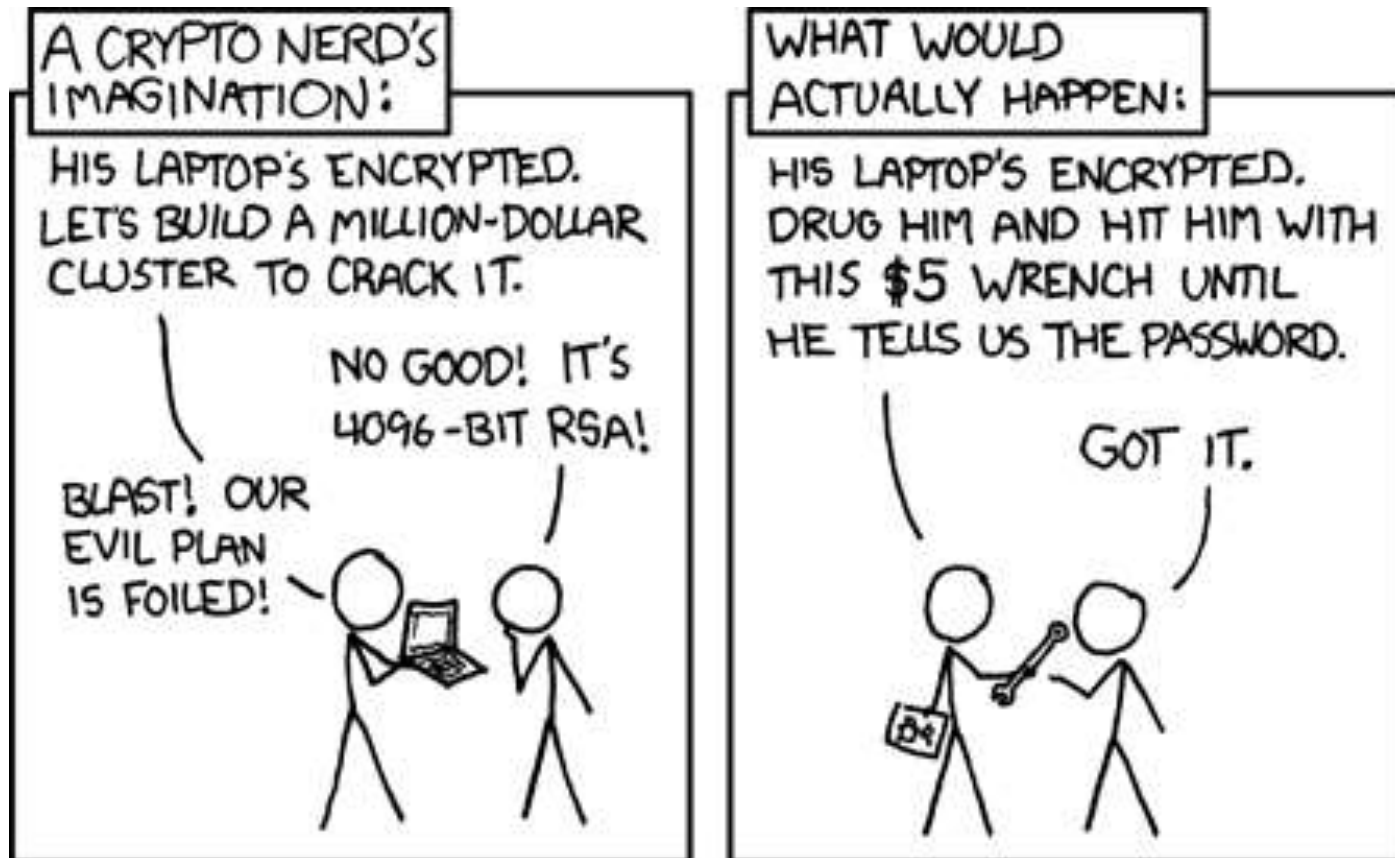
- Documentación
- Detalles de logging, seguridad, estándares
- Evidencia de procesos de desarrollo seguro, etc.

- Almacenar datos con criptografía es relativamente fácil, pero introduce el problema de la gestión de claves



Gestión de claves

Hay 12 items del standard destinados específicamente a gestión de claves



Tokenización

PCI aplica sólo a “sistemas que almacenan, procesan o transmiten el n° de tarjeta”

La forma más fácil de “certificar PCI” entonces es evitar almacenar, procesar o transmitir el n° de tarjeta

Estrategia de reducción del alcance a través de tokenización



Vade retro, scope PCI

Qué es


```
def tokenize(card:String): String
```

```
5588 3201 2345 6789
```

->

```
1000 0000 4365 6789
```

Producto **middleware tokenizador**
“SecureTX”



Security Standards Council™

Standard:	PCI Data Security Standard (PCI DSS)
Version:	2.0
Date:	August 2011
Author:	Scoping SIG, Tokenization Taskforce PCI Security Standards Council

Information Supplement:
PCI DSS Tokenization Guidelines





SecureTX



TOKENIZE

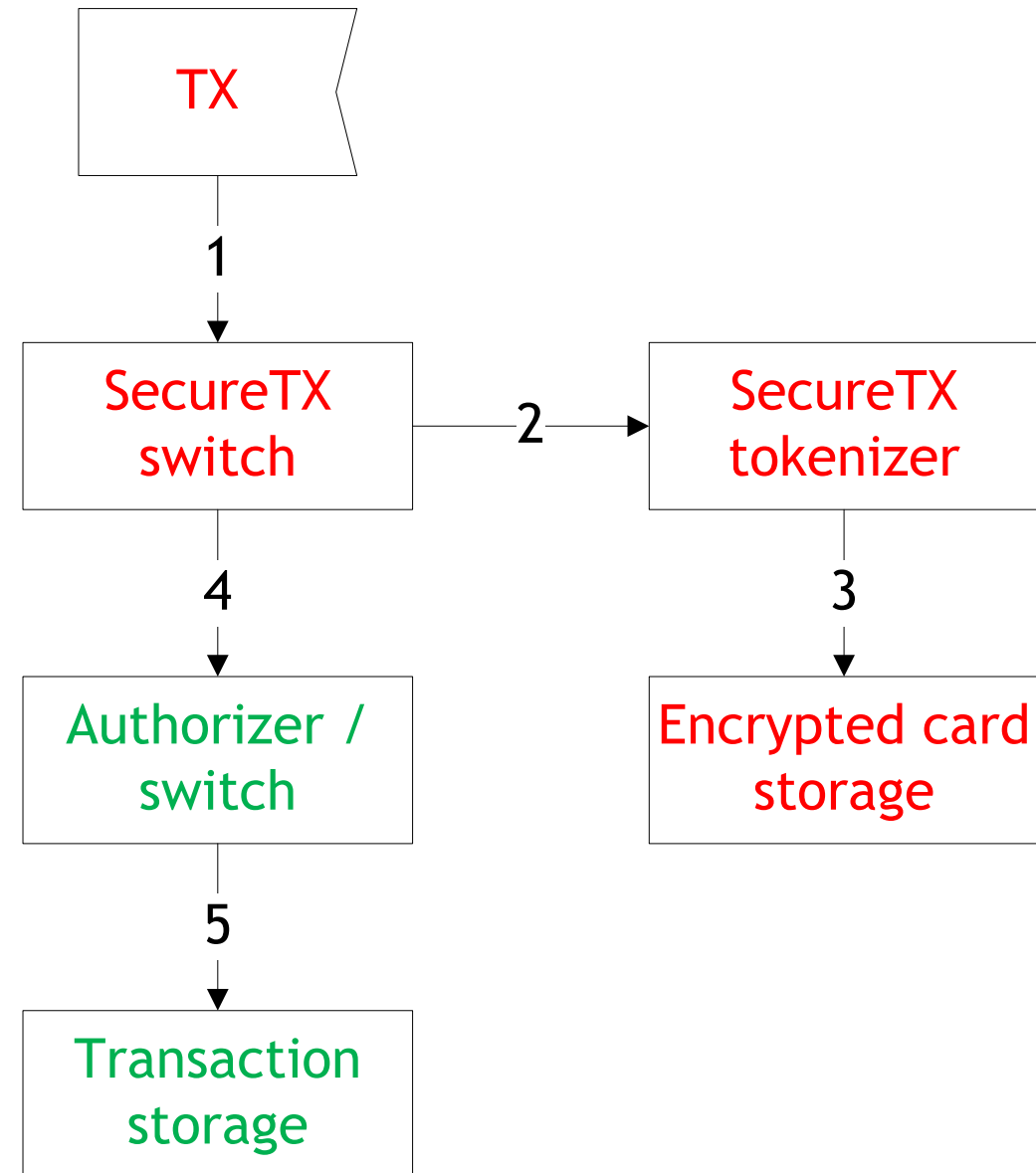


ALL THE THINGS

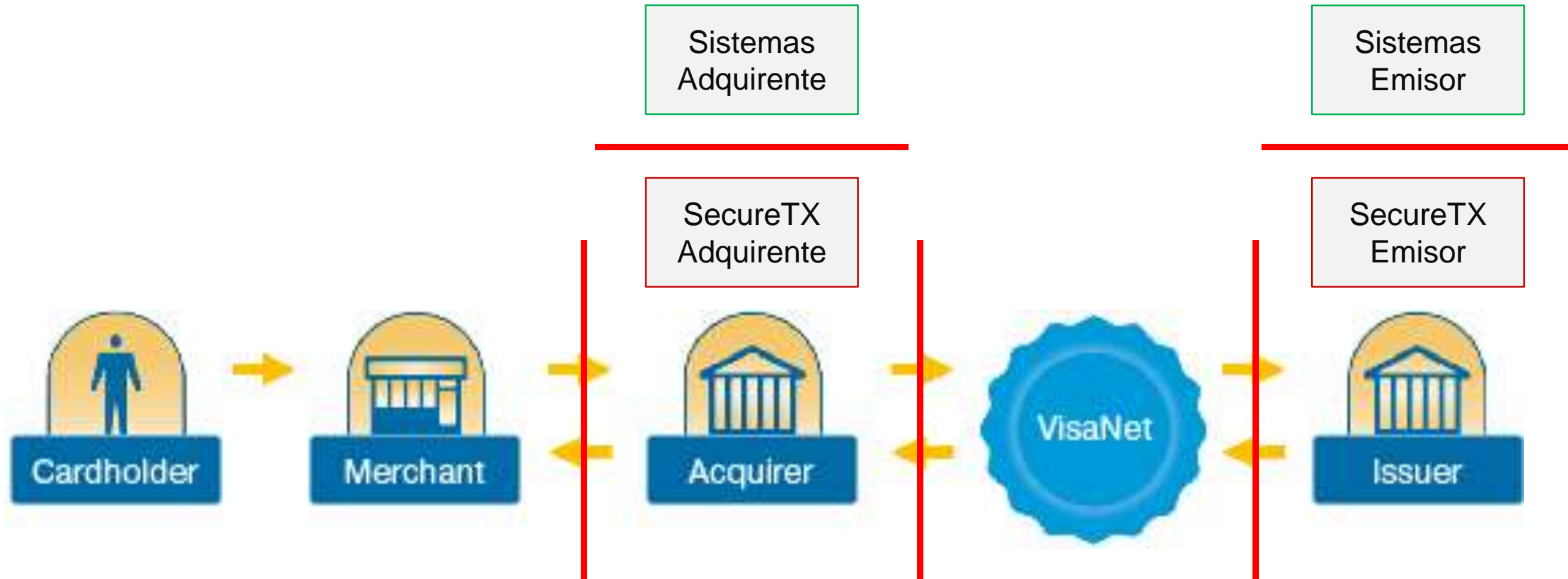


Qué hace

- TX - fuente de transacciones (VISA, red de POS...)
- Tokenizer - servicio de tokenización
- SecureTX Switch - middleware
- “Authorizer / switch” podría ser un procesador de pagos cualquiera
- El esquema de la derecha es minimal, se dan escenarios más complejos



Dónde juega



Tecnología

Plataforma JVM

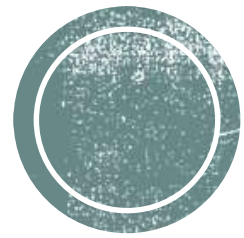
Tokenizador

- Java
- Jersey / Grizzly
- Oracle / Sql Server
- API REST
 - /cardid/{tarjeta}
 - /cardnumber/{token}

Switch

- Scala
- Akka
- Apache Camel
- Múltiples canales de comunicación usando componentes y rutas Camel
- Soporte nativo ISO 8583
 - Tanto el **formato** como el **protocolo**





SDLC

Microsoft SDL

Relación con OWASP

Desarrollo seguro

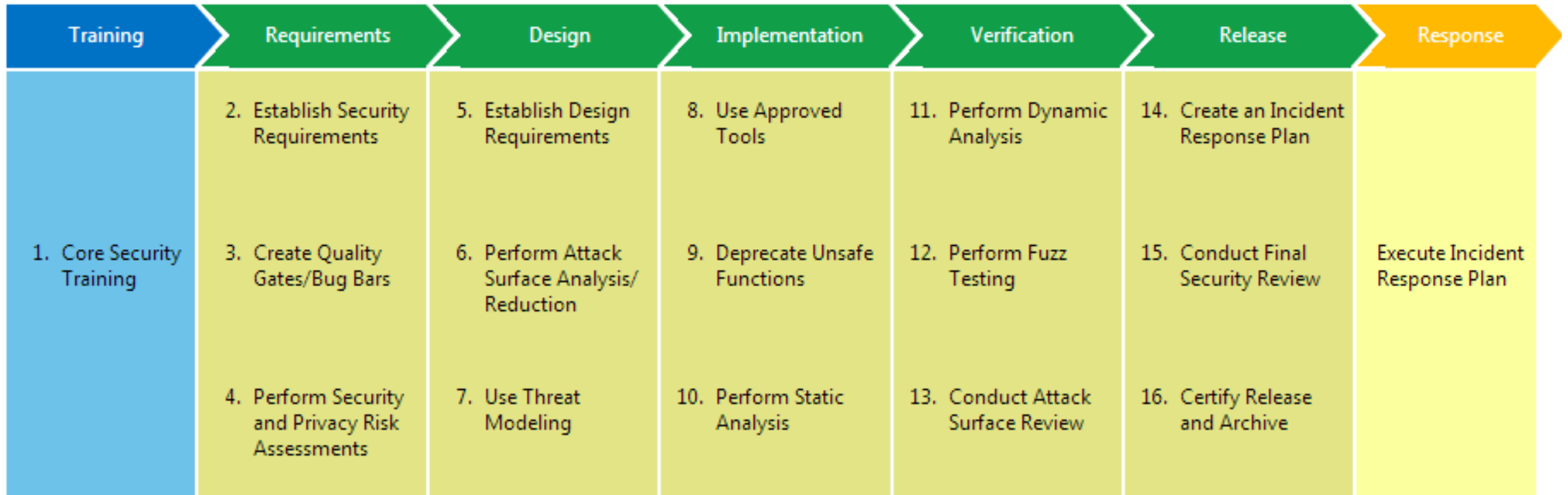
PA-DSS 5: Develop secure payment applications

- Basado en *industry best practices*
- Incorporando seguridad de la información en todo el proceso
- Previniendo vulnerabilidades comunes (referencia explícita a OWASP Top Ten)
- Control de cambios
- Utilizar servicios o librerías seguras



Microsoft SDL

- Es una *industry best practice* aceptable
- Mucha documentación y herramientas de uso libre
- Perfectamente compatible con OWASP



OWASP

Aunque nuestra aplicación no es web igual OWASP nos fue de mucha ayuda

- Documentación
 - Hay un enorme valor en los documentos y guías de OWASP, concretamente:
 - Top Ten
 - Secure Coding Practices
 - Development Guide
 - Testing Guide
 - Coding Review Guide
- Herramientas
 - jBroFuzz
 - dependency-check



jBroFuzz

- Lo terminamos usando como librería, dentro de tests automatizados
- jBroFuzz nos permite generar fácilmente cientos de inputs inválidos representando vulnerabilidades típicas (SQL Injection, XSS, etc.)
- Enviamos inputs inválidos a las *trust boundaries* identificadas en el *threat model*
- Importante ayuda para cumplir con PA-DSS 5



dependency-check

- "Remedio" para OWASP A-9: Using Components with Known Vulnerabilities
- Visto en: Black Hat 2013 Arsenal
- Identifica CVE's en librerías Java a partir de nombre / hash de los .jar
- Importante ayuda para cumplir con PA-DSS 7

Published Vulnerabilities

[CVE-2013-4330](#)

Severity: Medium

CVSS Score: 6.8

CWE: CWE-94 Improper Control of Generation of Code ('Code Injection')

Apache Camel before 2.9.7, 2.10.0 before 2.10.7, 2.11.0 before 2.11.2, and 2.12.0 (2) FTP producer.

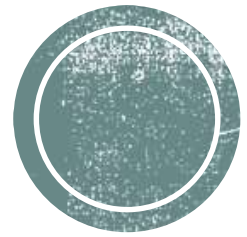
- CONFIRM - <http://camel.apache.org/security-advisories.data/CVE-2013-4330>
- FULLDISC - [20130930 CVE-2013-4330: Apache Camel critical disclosure v](#)
- MISC - <http://packetstormsecurity.com/files/123454/>
- OSVDB - [97941](#)
- SECUNIA - [54888](#)
- XF - [apache-camel-cve20134330-code-exec\(87542\)](#)



Otras herramientas

- Análisis estático de código
 - PMD
 - FindBugs
 - ScalaStyle
- Scan de datos
 - Spider
- Los tests automáticos, dependency-check, y análisis estáticos de código, corren ante cada cambio en el build server





Preguntas

