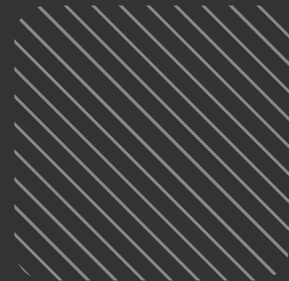




Bootcamp Talk WebSec

Luciano Ciattaglia
Fabrizio Faggiani
Ezequiel Moreno
Lautaro Colombo



Schedule



Web Security

01 Trainings
Status

02 Vulnerabilidades

03 Mecanismos de Protección
Fundamentales

04 Threat
Modeling

05 Security
Tips

IT BOARDING

BOOTCAMP

// Security Training Status

IT BOARDING

BOOTCAMP

(Non)Secure Developers



// Vulnerabilidades

¿Qué son?

IT BOARDING

BOOTCAMP

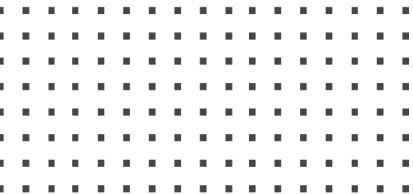
Definiciones

Vulnerabilidad

“Una vulnerabilidad es una debilidad o fallo de un sistema informático que puede ser utilizada para causar un daño”

Riesgo

“Combinación de la probabilidad de que se aproveche una vulnerabilidad y sus consecuencias negativas o impacto”






Definiciones

Cálculo del riesgo cualitativo

		CLASIFICACIÓN DE RIESGOS		
IMPACTO	ALTO	MEDIO	ALTO	CRÍTICO
	MEDIO	BAJO	MEDIO	ALTO
	BAJO	BAJO	BAJO	MEDIO
		BAJA	MEDIA	ALTA
		PROBABILIDAD		

Prioridades en MELI

En Meli utilizamos 4 tipos de prioridades, en donde cada una de ellas determina el tiempo de resolución de una vulnerabilidad.

Prioridad	SLA
Crítica 	48 hs
Alta 	1 semana
Media 	2 semanas
Baja	1 mes

Reporte

Cada reporte contiene:

- La **iniciativa** responsable de gestionar la vulnerabilidad.
- Fecha de vencimiento del **SLA**.
- Enlace a **JIRA** para gestionar el reporte.
- Un **ID** para trackear el reporte.

 Seguridad Web **creó** una incidencia

SSRF en /pkm/listas/api/redirectmiddleend-redirect

Iniciativa	{INICIATIVA}
<u>Vencimiento de SLA</u>	19/12/2019
Severidad	Crítica
JIRA	WSMOREMTP-7
ID	061f3659-0666-4cc2-ae2f-32ba4357f72c

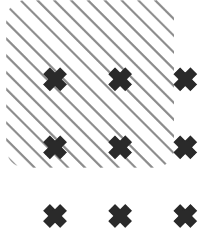
// Web Application (In)Security

¿Por qué es tan difícil crear una aplicación segura?

IT BOARDING

BOOTCAMP

Problema Core

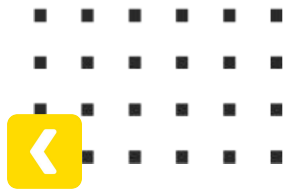


Las aplicaciones están frente a un problema fundamental:

Los usuarios pueden enviar input arbitrarios

Esto se manifiesta en:

- Un usuario puede interferir con cualquier información entre el cliente y el servidor.
Cualquier control implementado client-side es trivial de bypassar.
- Un usuario puede enviar request en cualquier orden y parámetros en diferentes momentos de los que se espera, más de una vez o nunca.
Cualquier suposición hecha por los developers de cómo van a interactuar con la app puede no cumplirse.
- Un usuario no necesariamente va a usar un Browser o App únicamente para acceder.
Existen herramientas preparadas para atacar aplicaciones que superan cualquier funcionalidad de un Browser.



Factores Key

Múltiples factores exacerbaron el problema:

- Awareness insuficiente
- Desarrollos *custom*
- Simplicidad engañosa
- Restricciones de tiempos y recursos
- Demandas de funcionalidades más complejas

Hoy las aplicaciones web (APIs, frontends, etc) son el perímetro de una organización.

Un atacante puede comprometer a una organización únicamente enviando un payload



Mecanismos de Defensa

Involucran:

- **Manejo de accesos de usuarios** y funcionalidades para prevenir de ganar accesos no autorizados.
- **Manejo de input de usuario** para prevenir el procesamiento de input mal formado y que cause un comportamiento no deseado.
- **Monitoreo activo** de la aplicación en sí misma, para permitir a los owners de la misma reaccionar ante errores o ataques dirigidos





Manejo de Accesos

- **Autenticación**
Determinar quién es el usuario que está haciendo una acción.
Ejemplo: Implementar 2FA en cuentas de usuario
- **Session Management**
Permitir al usuario mediante un token identificador, “mantener” una sesión activa.
Ejemplo: Cookie session-id.
- **Access Control**
Determinar lógica que permita a los usuarios consultar recursos que le pertenezcan sólo a ellos y no a otros.
Ejemplo: Utilización de OAuth.

Un error muy común es el de autenticar pero no autorizar recursos.





Manejo de Input de Usuario

Es la práctica de validar que todos los valores no originados en nuestra aplicación están bien formados previo a procesarlos, guardarlos o transmitirlos.

- **Validaciones sintácticas:**

El valor se encuentra correctamente formado con respecto al tipo o estructura de dato esperado. Ejemplo: `Long orderId = (Long) req.orderId`

- **Validaciones semánticas:**

El valor es correcto en términos del contexto en el que se está utilizando.
Ejemplo: El monto de una transacción no debe ser negativo.

El mayor % de las vulnerabilidades ocurren debido a que procesamos todo tipo de valores sin antes verificar que los mismos sean los que esperamos





Monitoreo Activo

- **Errores**
Reaccionar públicamente de forma genérica.
Ejemplo: Evitar que el HTTP Response contenga información the debug o StackTrace.
- **Logs, logs, logs!**
Ante algún incidente, logs efectivos deberían permitir entender qué pasó.
Ejemplo: Logueo de eventos clave: autenticación, transacciones, cambio de datos, intentos de acceso, etc.
- **Alertas**
Permiten tomar acciones inmediatas, en real-time.
Ejemplo: Ataques automatizados, brute-force, payloads conocidos, etc.
- **Reacción ante Ataques**
Mecanismos que ejecutan acciones ante detección de ataques.
Ejemplo: Captchas.

Es un error común el logging de información PII!



// Threat Modeling

¿Qué es?

IT BOARDING

BOOTCAMP



Threat Model

Objetivo

El ejercicio de Threat Model tiene como objetivo detectar potenciales fallas de seguridad lógicas y técnicas en el diseño de un producto.

Beneficios

De esta forma es posible evitar vulnerabilidades estructurales, las cuales de ser detectadas en etapas más tardías del ciclo de vida, serían mucho más caras de mitigar.

✖ ✖ ✖

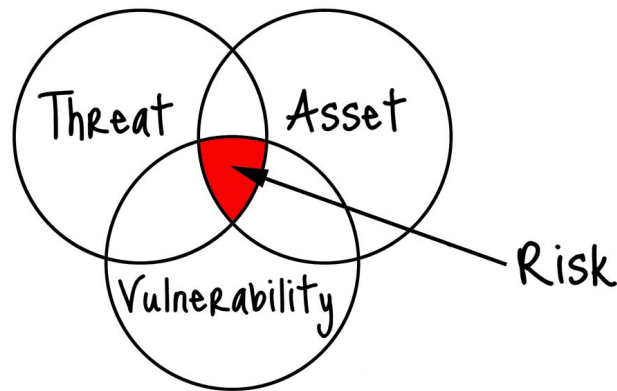
✖ ✖ ✖

✖ ✖ ✖

Threat Model - Estructura

Durante el ejercicio y con el detalle del diseño/implementación de mi aplicación, intentamos responder las preguntas:

- **What are we building?**
- **What can go wrong?**
- **What are we going to do about it?**



// Threat Model DELUXE

¿Qué identificamos en la app que realizamos?

IT BOARDING

BOOTCAMP

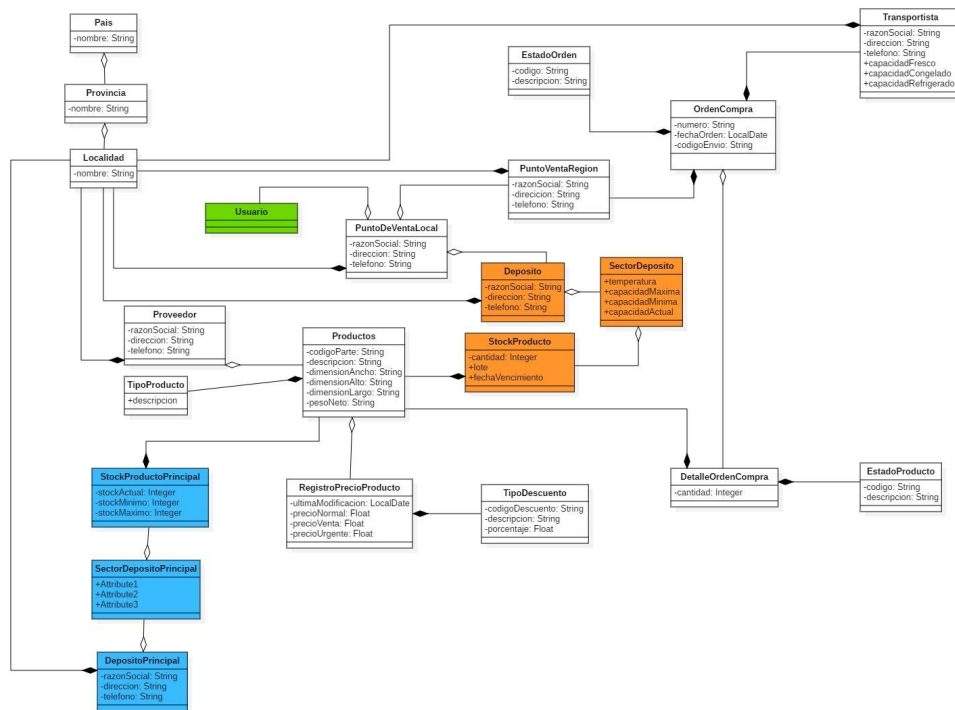
Threat Model - FRESCOS

Meli desea ampliar su negocio para trabajar con la industria alimentaria (**FRESCOS**), donde todos los productos para que puedan ser almacenables, transportables y comercializables deben de poseer información común como la **fecha de caducidad** y **número de lote**, cada tipo de producto además posee alguna información específica.

Los productos agroalimentarios se dividen en tres categorías:

- **Productos frescos:** deben de estar limpios y refrigerados dentro de un rango de temperatura expresadas en grados Celsius, los mismos tienen una caducidad más rápida, ejemplo: productos de la huerta, panificados
- **Productos refrigerados:** deben llevar la temperatura de refrigeración recomendada expresada en grados Celsius, ejemplo: lácteos,
- **Productos congelados:** deben llevar la temperatura de congelación recomendada expresada en grados Celsius, a su vez existen tres tipos de productos congelados
 - **Producto congelado por aire:** se congelan de manera progresiva
 - **Producto congelado por nitrógeno:** se congelan de manera inmediata

Threat Model - FRESCOS



Threat Model - FRESCOS

Consideraciones de arquitectura (integration):

- **Autenticación**

- Teniendo en cuenta la necesidad de que una API externa a Meli consuma nuestros endpoints ¿Cuales consideran sería la mejor forma de autenticarlos?

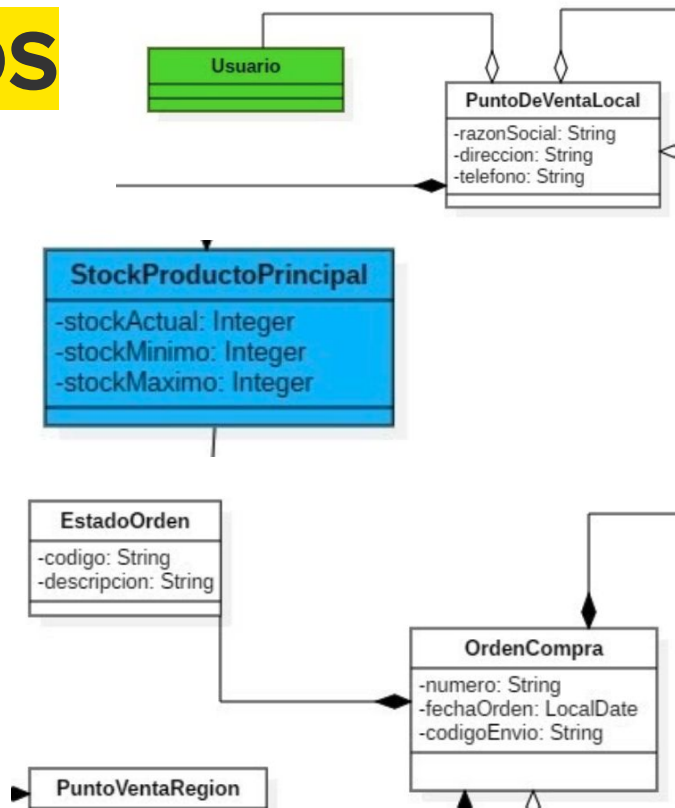
- **Autorización**

- ¿Puedo ver las órdenes de compra de cualquier persona?
- ¿Puedo alterar los estados de las ordenes solo conociendo su id?

Threat Model - FRESCOS

Consideraciones de arquitectura:

- **Validacion sintáctica:**
 - `fechaOrden` es una fecha?
 - `stockActual` es un entero?
- **Validacion semántica:**
 - `codigoOrden` es un enumerado['A','T']?
 - `pesoNeto` puede ser negativo?
 - `stockMaximo` puede ser menor que `stockMinimo`?



Threat Model - FRESCOS

Consideraciones de arquitectura:

- **Input Validation (SQLi Example)**

- Que pasa si me ingresan como `codigoEnvio`, el siguiente string:

```
String codigoEnvio := ‘; DROP TABLE OrdenCompra;’
```

- Y la consulta que termina haciendo mi aplicación a la base de datos es la siguiente:

```
func buildSql(dealerNumber string) string {  
    return fmt.Sprintf("SELECT * FROM CE_table WHERE CE_number = '%s';", dealerNumber)  
}
```



DANGER

```
SELECT * FROM CE_table WHERE CE_Number=''; DROP TABLE CE_table;'';
```

Threat Model - FRESCOS

Consideraciones generales:

- **Logs, auditoría**
 - ¿La casa matriz tiene logs de los pedidos que recibe?
 - ¿Cómo me defiendo en caso de incidentes?
- **Rate limiting, ataques de DOS**
 - ¿Qué pasa si un usuario solicita 1k de ordenes en 1 hora?
- **Privacidad de Datos**
 - ¿Qué pasa si un usuario se da de baja?
 - ¿Necesito mantener los datos en el tiempo?

Threat Model - DELUXE

Consideraciones generales:

- **Logs, auditoría**
 - ¿Se tienen logs de los pedidos que se reciben?
 - ¿Cómo me defiendo en caso de incidentes?
- **Rate limiting, ataques de DOS**
 - ¿Qué pasa si una filial solicita 1k de repuestos en 1 hora?
- **Privacidad de Datos**
 - ¿Qué pasa si una concesionaria o filial se da de baja?
 - ¿Necesito mantener los datos en el tiempo?

Threat Model - DELUXE

Se requiere para la casa matriz **(warehouse)** ubicada en Brasil desarrollar el módulo “Gestión de Repuestos”, el cual permite administrar los pedidos de repuestos provenientes de las casas centrales de cada país **(filial)**



Threat Model - DELUXE

Consideraciones de arquitectura:

- **Autenticación**

- ¿Existe algún riesgo de enviar username y password por parámetro en cada solicitud?

- **Autorización**

- ¿Puedo ver el estado del pedido de otras casas centrales?

`GET /api/parts/orders/{orderNumberCM}`

- ¿Puedo solicitar un repuesto a otra Concesionaria(CE) si conozco su número?

`POST /api/parts/orders/order-request`

Threat Model - DELUXE

Consideraciones de arquitectura:

- **Validacion sintáctica:**
 - `orderDate` es una fecha?
 - `dealerNumber` es un entero?
- **Validacion semántica:**
 - `carrier` es alfanumérico?
 - `orderDate` mayor a la fecha actual?
 - `shippingWay` es un enumerado['A','T']?

Enviar pedido de repuesto - **POST** /api/parts/orders/

Parámetros de Entrada		
Nombre	Descripción	Obligatorio
username	Usuario suministrado por la casa matriz 8 caracteres alfanuméricos	SI
password	Contraseña suministrada por la casa matriz 8 caracteres alfanuméricos	SI
orderDate	Fecha de pedido yyyy-MM-dd	
dealerNumber	Número del CE (Concesionario) que lo solicita Formato: 0000 (4 caracteres numéricos)	SI
typeOrder	Código de tipo de pedido, N: Normal, U: Urgente	SI
shippingWay	Medio por el cual se enviará el pedido A: Aéreo T: Terrestre	NO
carrier	Nombre del transportista, 100 caracteres alfanuméricos	NO
serialNumber	Numero de VIN del vehículo para el cual son los materiales Formato VIN: 17 caracteres alfanuméricos	SI

Threat Model - DELUXE

Consideraciones de arquitectura:

- **Input Validation (SQLi Example)**

- Que pasa si me ingresan como `dealerNumber`, (CE) el siguiente string:

```
String dealerNumber := ' ; DROP TABLE CE_table ;'
```

- Y la consulta que termina haciendo mi aplicación a la base de datos es la siguiente:

```
func buildSql(dealerNumber string) string {  
    return fmt.Sprintf("SELECT * FROM CE_table WHERE CE_number = '%s';", dealerNumber)  
}
```



DANGER

```
SELECT * FROM CE_table WHERE CE_Number=' ; DROP TABLE CE_table ;';
```

Threat Model - DELUXE

Consideraciones generales:

- **Logs, auditoría**
 - ¿La casa matriz tiene logs de los pedidos que recibe?
 - ¿Cómo me defiendo en caso de incidentes?
- **Rate limiting, ataques de DOS**
 - ¿Qué pasa si una filial solicita 1k de repuestos en 1 hora?
- **Privacidad de Datos**
 - ¿Qué pasa si una concesionaria o filial se da de baja?
 - ¿Necesito mantener los datos en el tiempo?

// Security Tips

Para el diseño e implementación de una app

IT BOARDING

BOOTCAMP



Security Tips

- ¿Conocemos los endpoints que disponibilizamos de forma pública?
- ¿Toda la información que devolvemos es indispensable?
- ¿Sólo le disponibilizamos la información a su dueño?
- ¿Sabemos cómo actúa la app en caso de error?
- ¿Tenemos los suficientes logs/metricas para alertarnos en caso de abuso?
- ¿Almacenamos datos que podrían identificar a un usuario?



Security Tips

- **Protección de Datos**
 - [Site Data Privacy](#)
 - [¿Cómo cuidamos los datos personales?](#)
- **Threat Modeling** ([formulario de pedido](#) o email a websec@)
- **Training**

Elearning seguridad web (secure code warrior)

 - [Presentación](#)
- **Estándares de desarrollo seguro**
 - [Security Cheatsheets](#)
- [WebSec Site](#)





Gracias.

IT BOARDING

BOOTCAMP

