Aplicaciones Web, Privacidad y Vulnerabilidades

Ariel Waissbein

Ariel Futoransky

CORE SECURITY TECHNOLOGIES

Agenda

1. Introducción
2. Catalogo de Ataques
3. Una propuesta nueva

Motivación

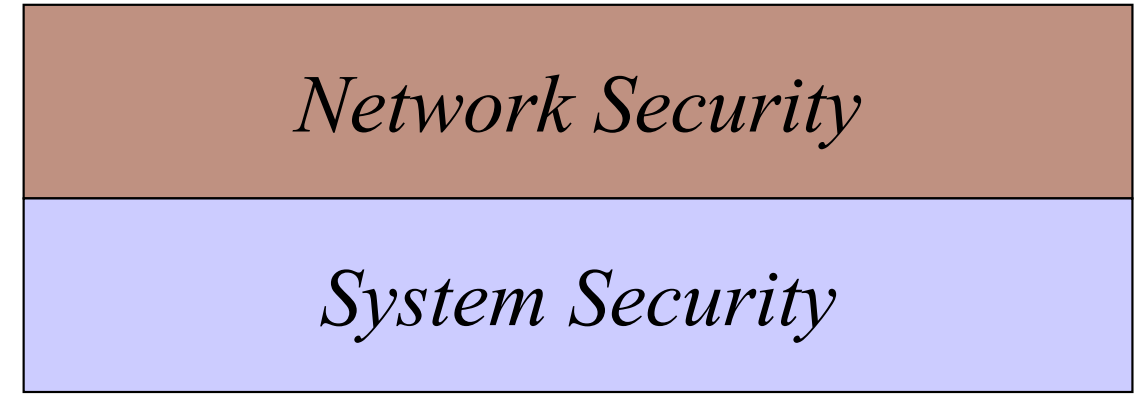
- Con la explosión de usuarios, la ubicuidad de Internet, y la pluralidad de servicios ofrecidos, han aparecido los atacantes.
- Hoy día, cualquier usuario de un sistema informático *sabe* que debe protegerse.
- … la "seguridad informática" es una disciplina; pero no una ciencia.
- Más precisamente, no hay modelos formales que permitan definir y resolver problemas. (Salvo en sub-campos.)
- Tenemos soluciones a (ciertos) problemas bien especificados
  - …y recetas, que llamamos "best practices", para los otros problemas.

Elementos

Estratégicos
- Jugadores
- Recursos
- Objetivos

Tácticos
- Vulnerabilidades
- Exploits

Seguridad de aplicaciones en contexto



Firewall
IDS

Patch Management
IPS

Source Auditing
Analysis tools
Filters

Vulnerabilidades en system/network security

- Vulnerabilidades Binarias
  - Buffer-overflows
  - Format-Strings
  - Heap-Management (dbl-free)

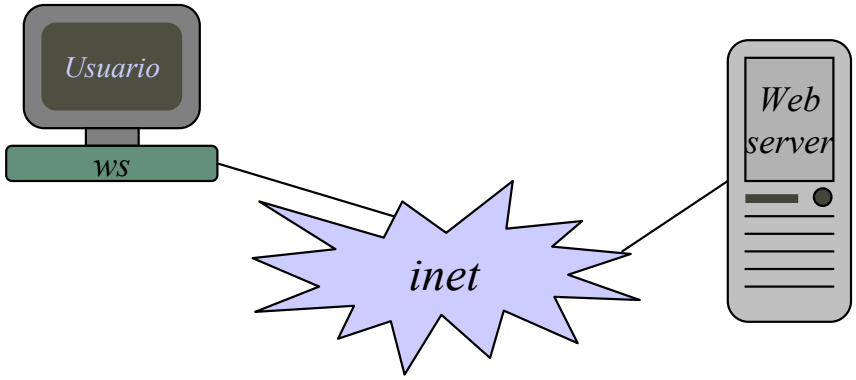
- Otras
  - Race-Conditions
  - Covert-Channels
  - Information Leaks
  - Protocol-errors

Soluciones disponibles

- Mejoras del compilador
  - Stack Protections: Canaries
  - Parameter reordering
  - Argument Copying
- Execution Domains
  - w^x
  - Ret Range-checks
  - Random Image addressing
- Sandboxing & Typesafety

En lo que se refiere a web applications…

- Actores:
  - Usuarios (e.g., internautas)
  - "Dueños" de la página
  - Atacantes
  - Desarrolladores (gráfica y lógica)
  - Sysadmins
  - Editores de Contenido



- Problemas asociados:
  - Los webapps manejan info. crítica de los usuarios
    - No hay garantias de seguridad y privacidad sobre esa info.
  - Los usuarios "corren código" en sus browsers.
    - No hay garantías sobre la integridad de ese código (e.g., no puedo distinguir código malicioso de código benigno).

Motivación (2)

- "Web applications" ha crecido de manera que hoy tenemos lenguajes en los que es trivial desarrollar páginas complejas (e inseguras).

- Día a día el número de ataques aumenta, y así su impacto en la población (e.g., incidentes con tarjetas de crédito, bancos,…)

- Costos:
  - Las empresas pueden perder credibilidad
  - Los usuarios recursos (tiempo, dinero)

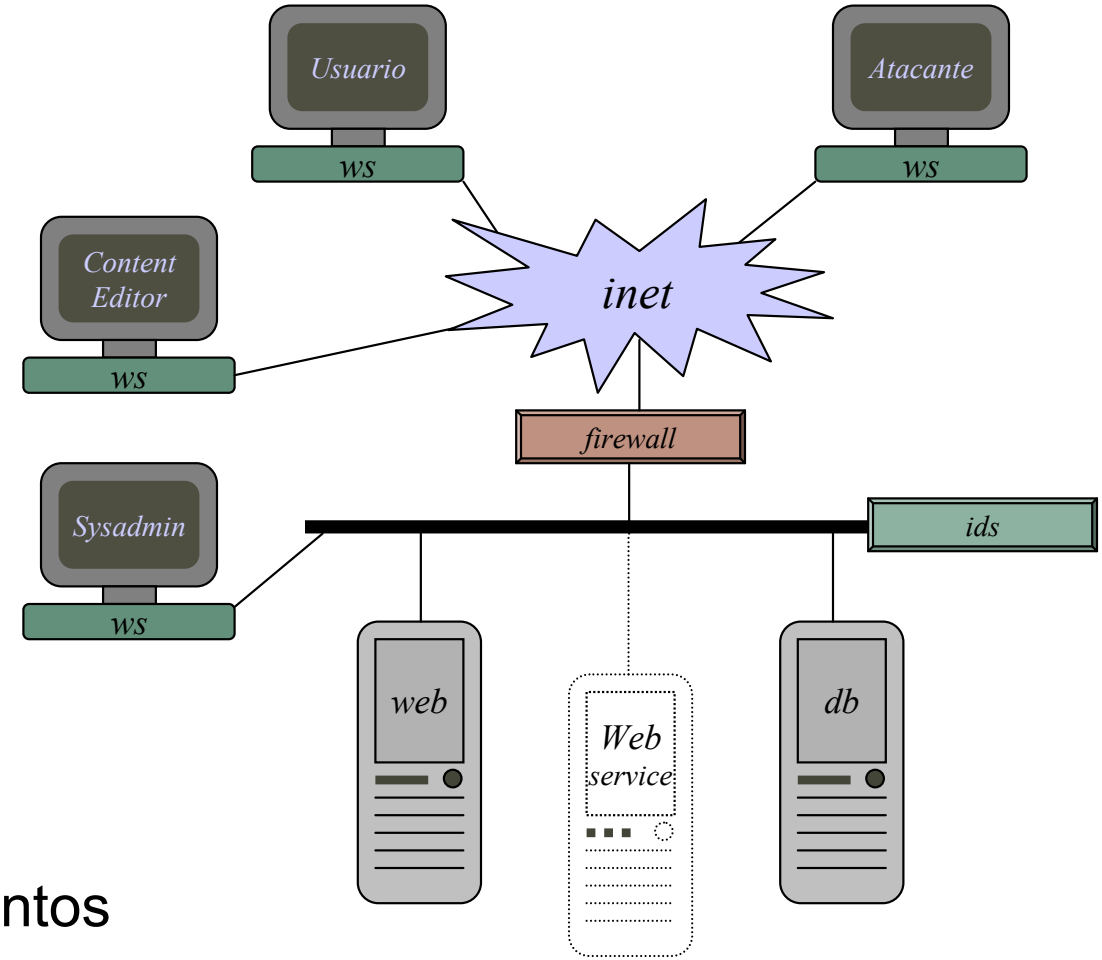
OWASP Top Ten Webapp Security Vulns: This "ten-most-wanted" list acuately scratches at the tip of an enormous iceberg. The underlying reality is shameful: most we application software is written oblivious to security principles, software engineering, operational implications, and indeed common sense.

-**Dr. Peter Neumann** (Científico ppal. De SRI Intl. Comp Sec. Lab.)

Catalogo de vulnerabilidades web

Entendiendo al problema…

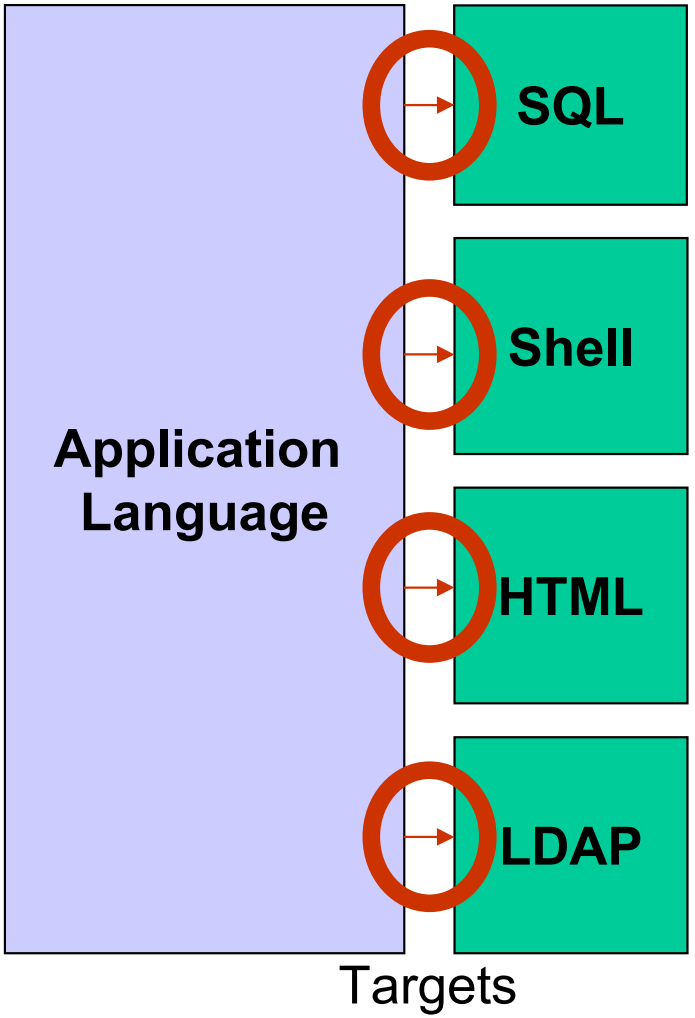
- Componentes:
  - Browser
  - Web Server
  - Database Server
  - Aplicación
  - [Web service]



- Las aplicaciones admiten distintos roles y son dinámicas.

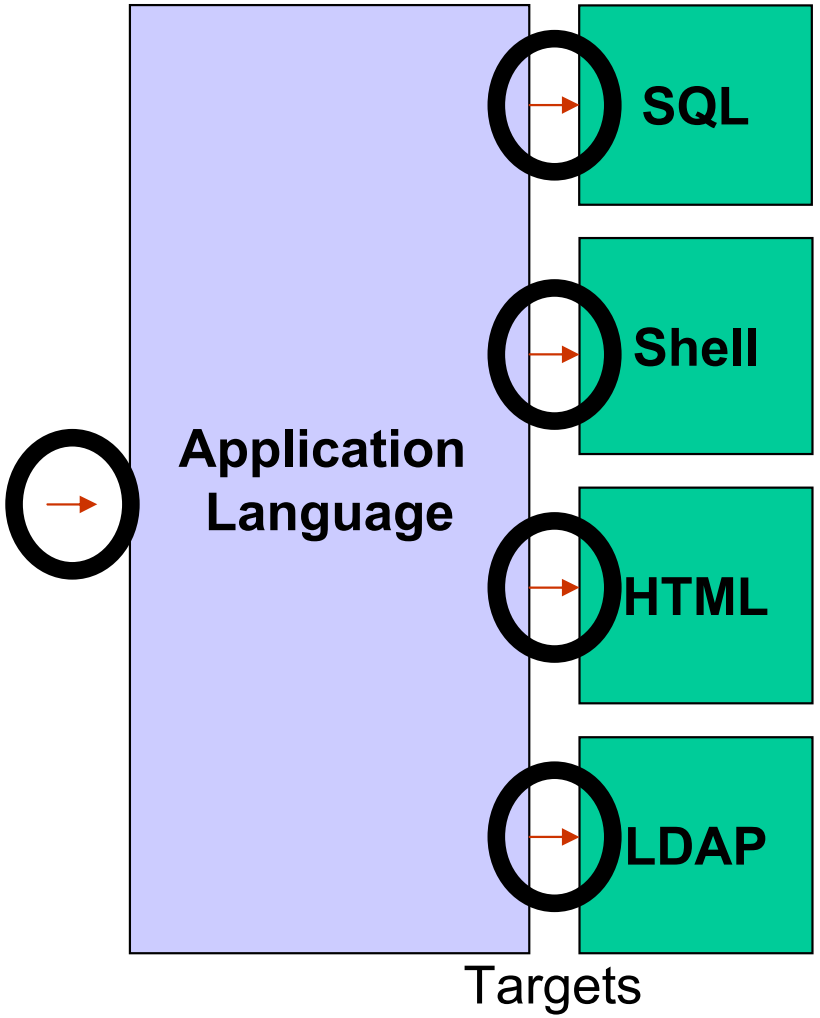
Anatomía de un ataque

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

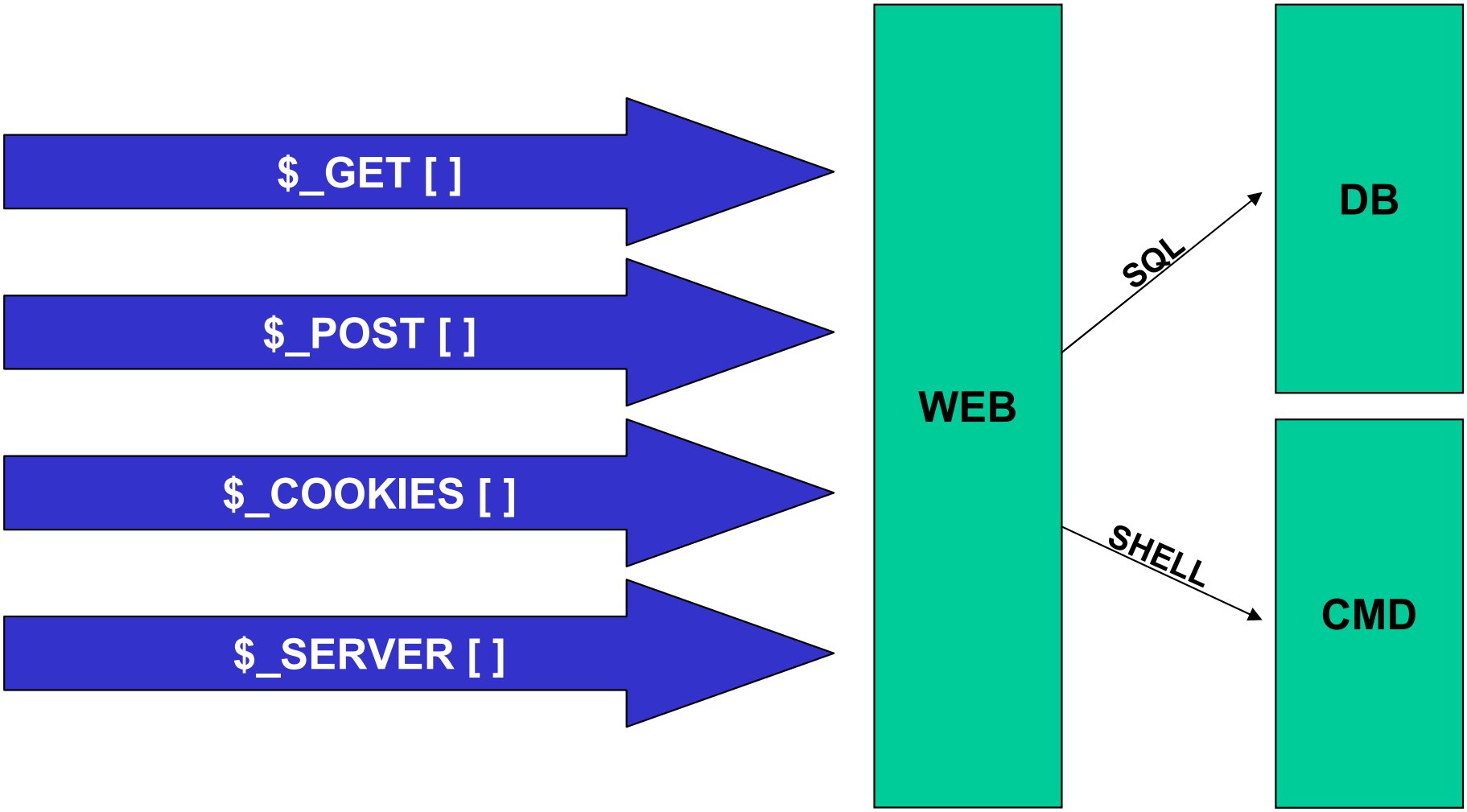


Herramientas de defensa

- Filtrar
- Normalizar/Escape
- Bloquear
- Mejorando la especificacion
- Tecnologias avanzadas



Vectores de ataque



Mas Vectores de Ataque

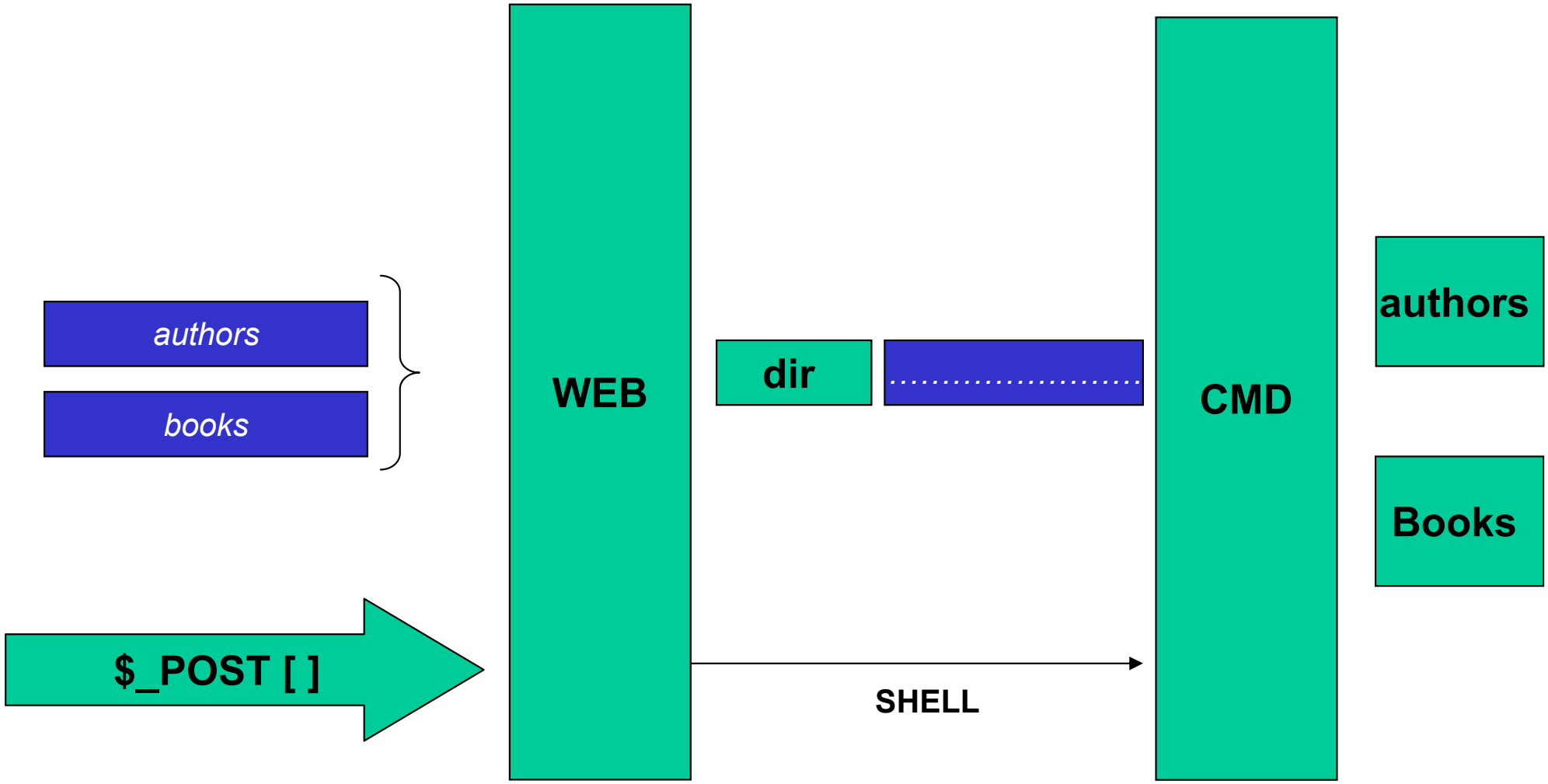
- HTTP_REFERER
- SERVER_NAME
- HTTP_HOST
- REMOTE_HOST
- REMOTE_ADDR

....

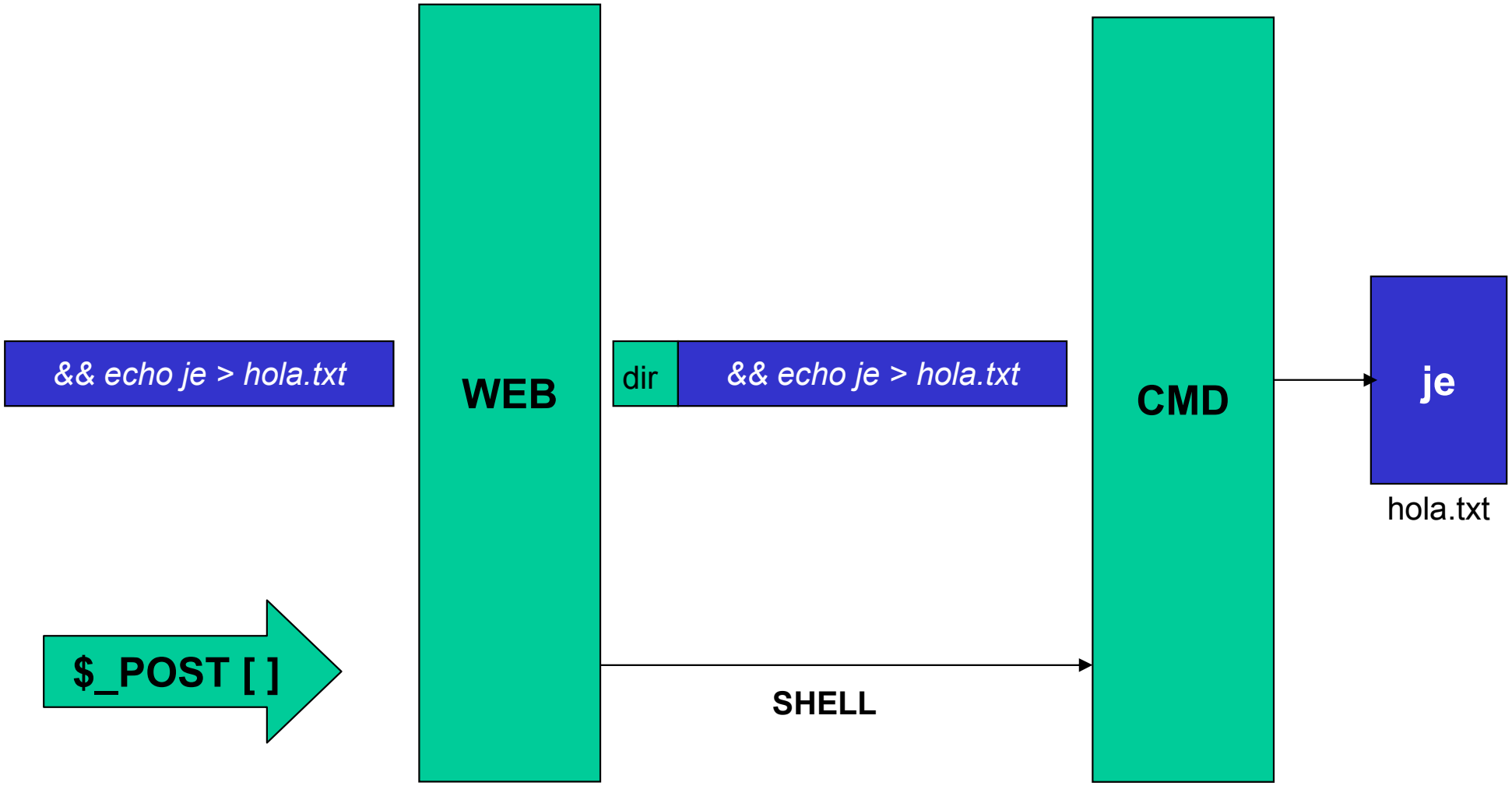
Todavía Mas Vectores de Ataque

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

Shell-Command-Injection



Shell-Command-Injection(2)



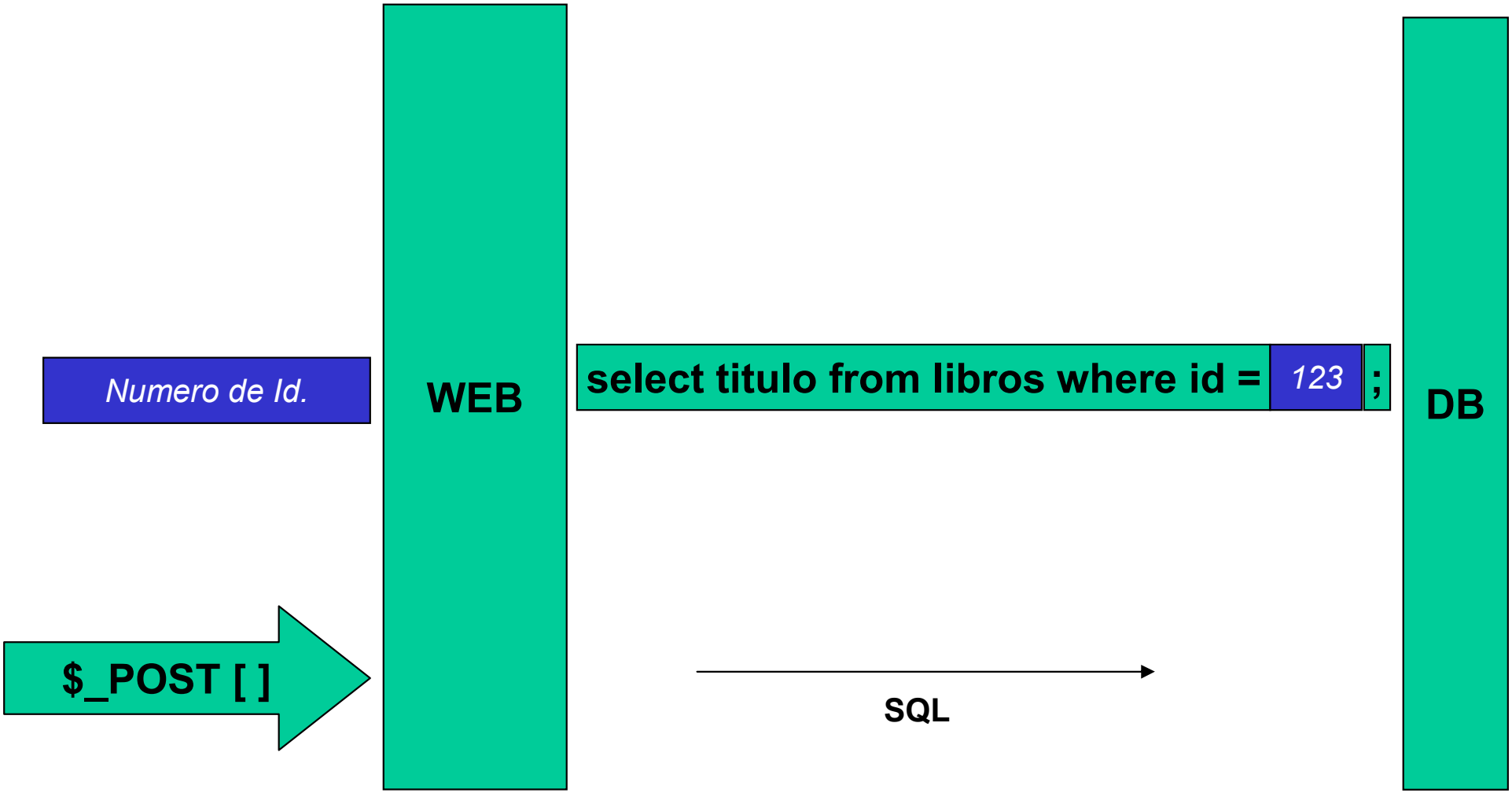
Protegiendose

- En PHP:
  - EscapeShellCmd()
  - EscapeShellArgs()

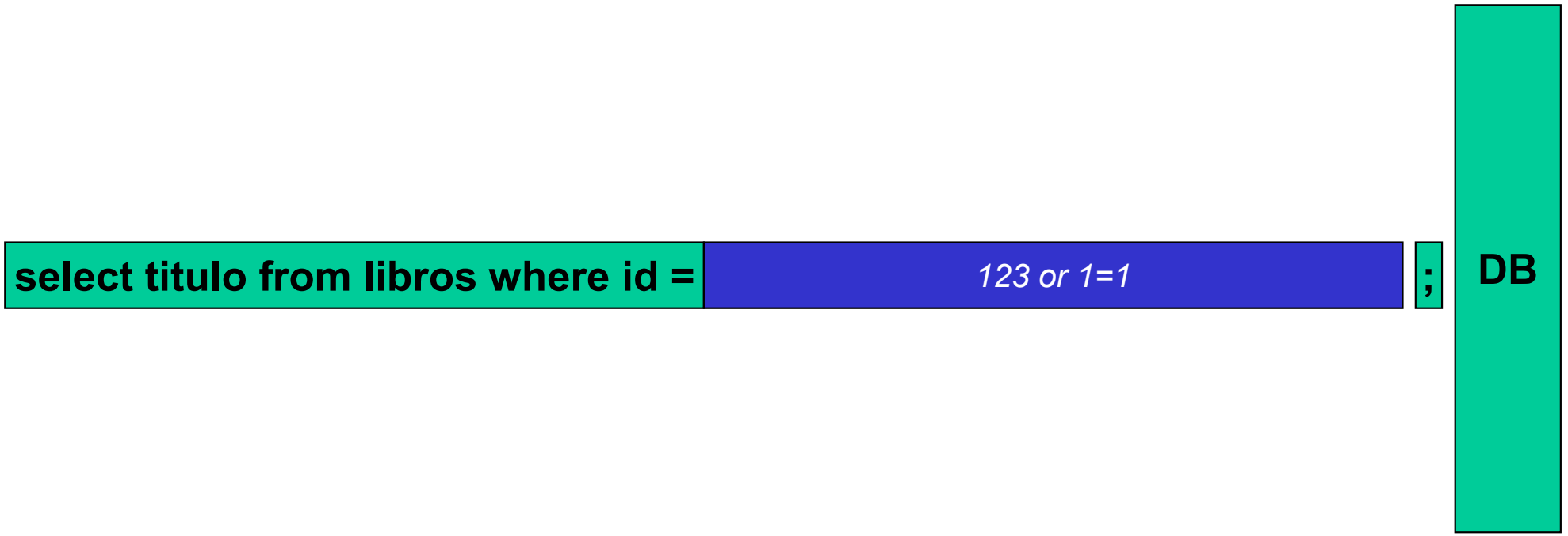
- Manualmente
  - Validando parametros
  - RegEx
  - Escapando metacaracteres

- Y además
  - Prohibiendo escribir en directorios browseables
  - Registrando eventos sospechosos para poder enterarse de un ataque

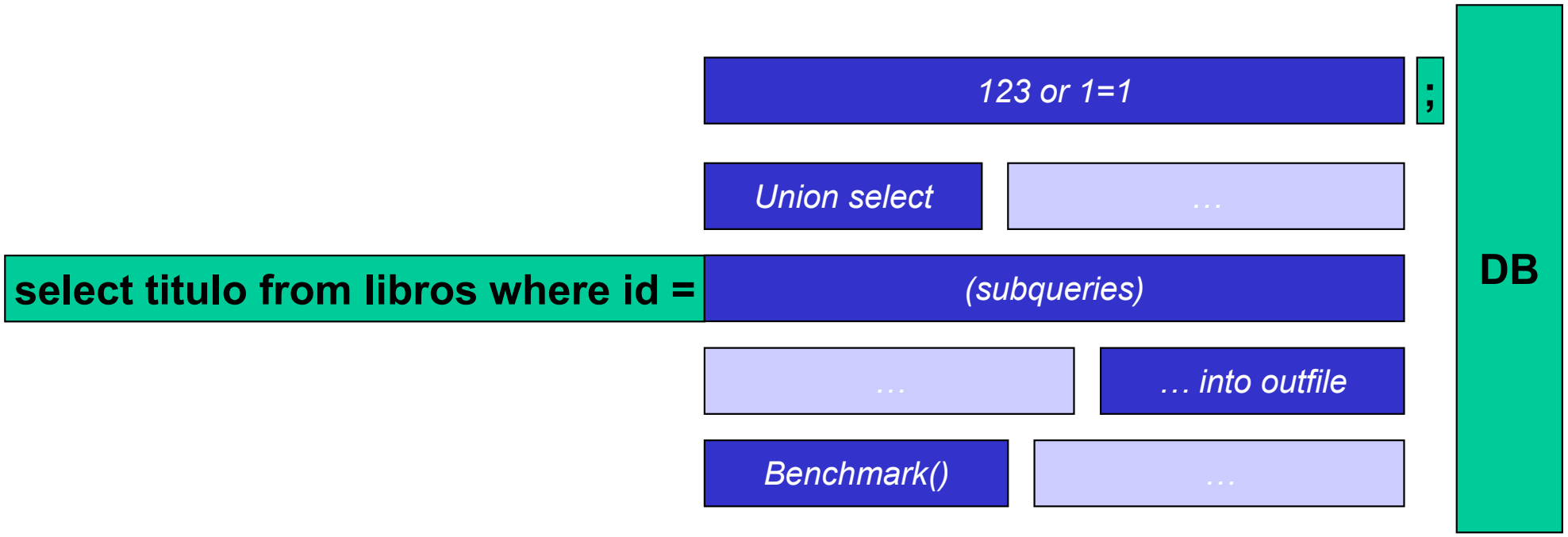
SQL Injection



SQL Injection (2)



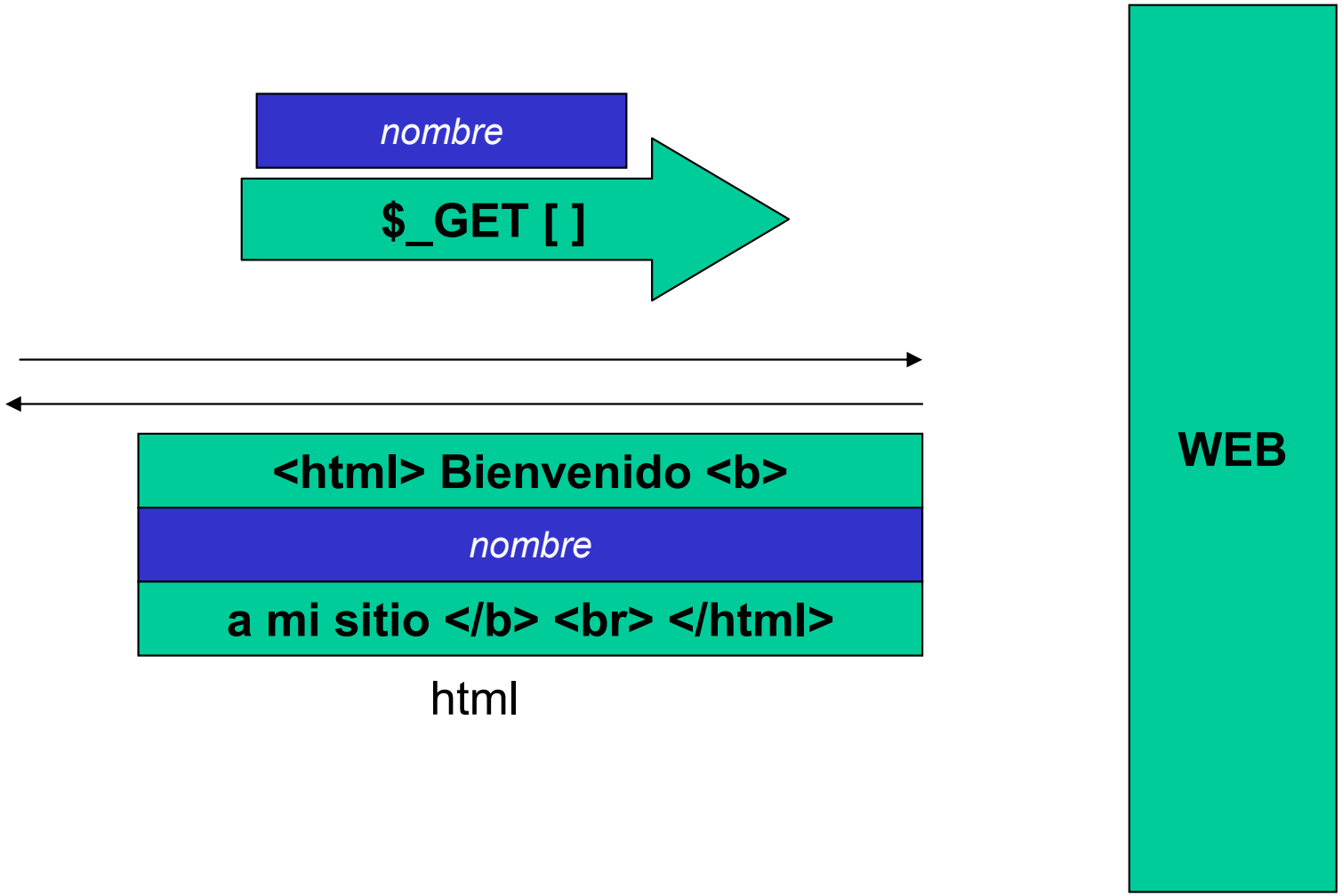
SqlInjection(3)



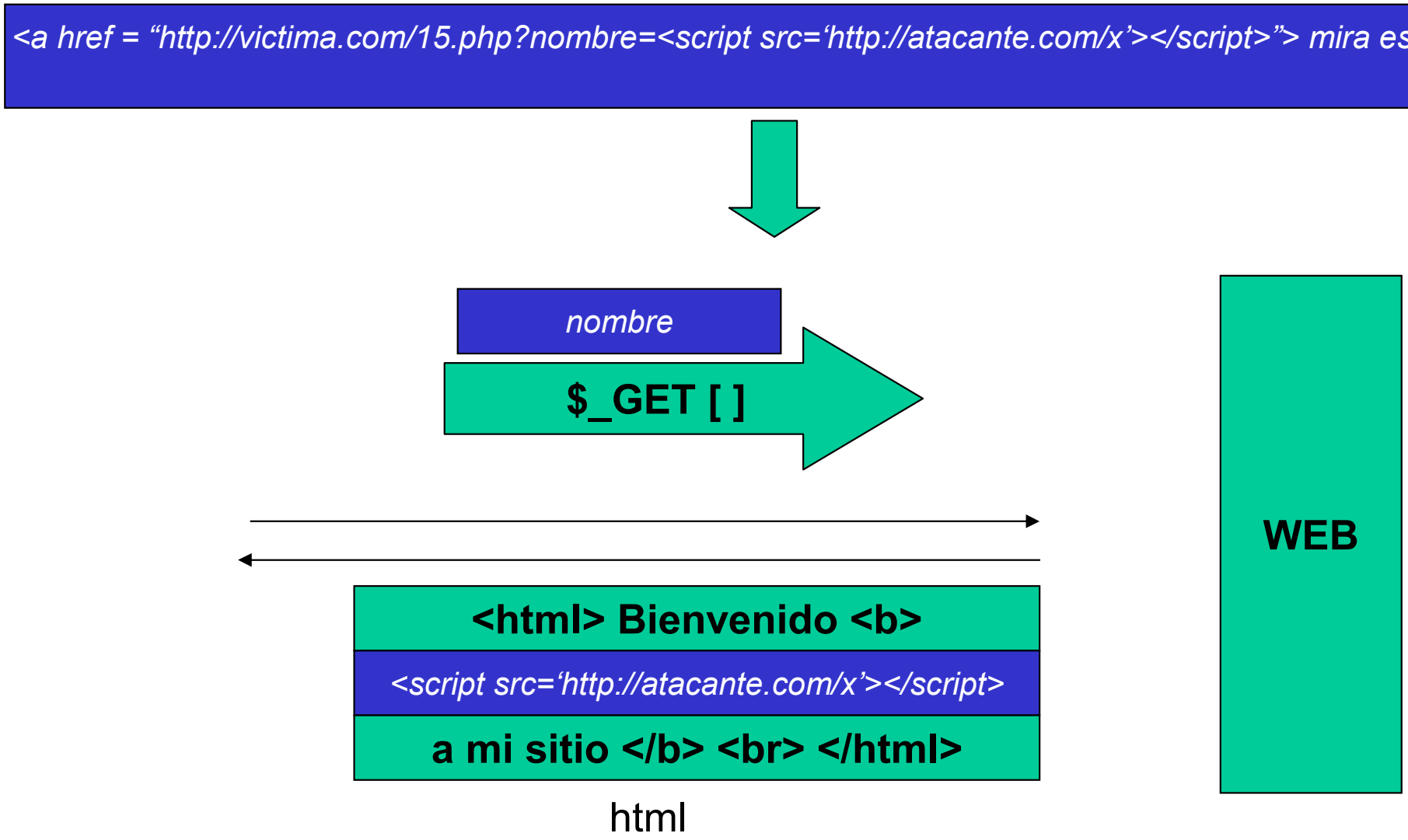
Protegiendose

- En PHP:
  - addslashes()
  - mysql_real_escape_string()
  - mysql_escape_string()
  - magic_quotes_gpc
  - Magic_quotes_runtime
- SQL-Parameters
- Manualmente
  - Validando parametros
  - RegEx
  - Escapando metacaracteres
- Y además
  - Prohibiendo escribir en directorios browseables
  - Registrando eventos sospechosos para poder enterarse de un ataque

XSS (cross-site-scripting)



XSS (cross-site-scripting)

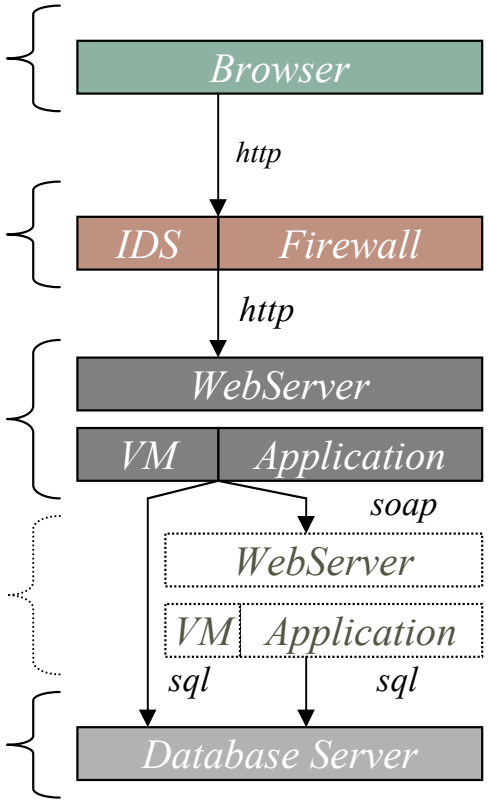


Protegiéndose

- Eliminando html-tags
- Encodeando html-tags
- Ojo con los canales disponibles!
- Validando http-referer?

Ataques: aquel top ten de OWASP

1. Parámeter tampering
2. Acceso de control inseguro
3. Manejo de sesión inseguro
4. Cross-site scripting
5. Buffer overflows
6. Command injection flaws
    1. E.g., database & Shell
7. Error handling
8. Mal uso de Crypto
9. Fallas de admin. Remota
10. Mala config. de server



Arquitectura de 2 o 3 capas

Y Privacidad?

- Las políticas de privacidad son objetos estáticos
- No hay herramientas para implementar
- El usuario no recibe garantías

Claves para la seguridad en WebApps

- Conocer las vulnerabilidades
  - Sql-injection, shell-command-injection, cross-site-scripting, directory-traversal, ldap-injection
- Conocer los vectores de ataque!
  - Desconfiar de todo el input del usuario
- Programar defensivamente
  - Validar, Validar, Validar
  - Usar las APIs de seguridad
  - Loggear cualquier evento medianamente sospechoso
- Instalar y configurar en modo paranoico
  - Restringiendo el impacto de una vulnerabilidad

Una modelo de protección

Nuestro Enfoque

- Es necesario aumentar el entorno de ejecución para incluir información que facilite diferenciar distintas estructuras de ataque
- La caracterización mencionada basta para identificar y detener las principales familias de ataques
- Adicionalmente puede incluirse información sobre la política de seguridad vigente para ser validada en el entorno.

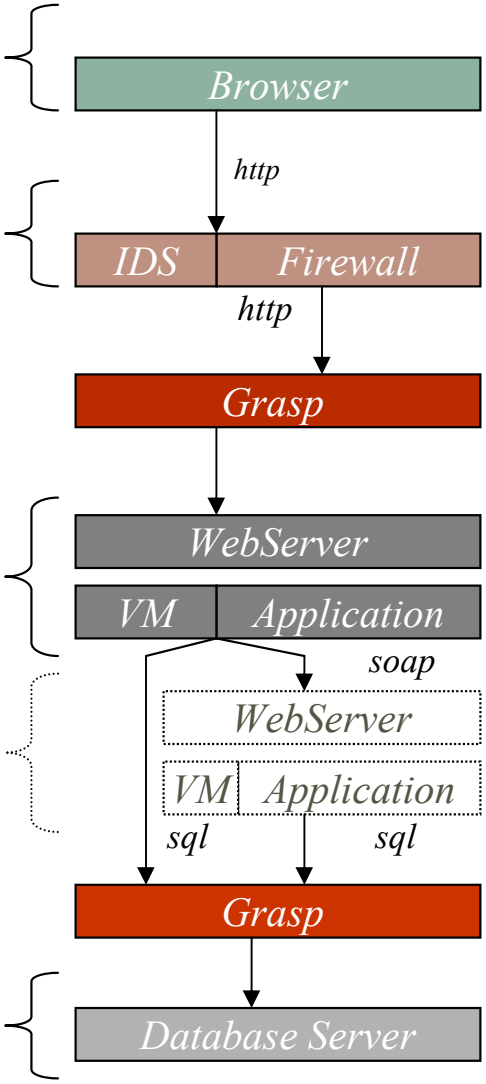
La técnica: modificar el execution environment

- Los objetos tienen asignados tags *user-controlled (y/o private)*.

- Estos tags se manejan con granularidad de caracacteres.

select * from users where uid = john;

e	r	e		u	i	d	=	j	o	h	n	;	String original
								x	x	x	x		Tags asociados

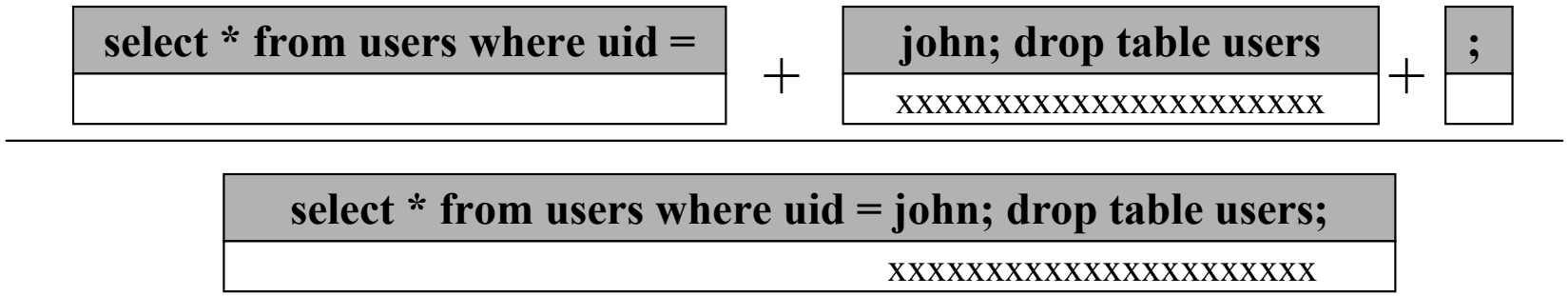


Agregando marcas

- Las marcas de seguridad son agregadas para cada string controlada por un potencial atacante
- Son considerados los distintos vectores de ataque descriptos

Propagando

- Las marcas son propagadas a nuevos objetos de forma de preservar la información de origen y permitir validaciones.



Identificando ataques

- Las parametros de operaciones protegidadas son inspeccionados buscando patrones de ataque.

- Puntos de inspección
  - Acceso a la base de datos
  - Acceso al file-system
  - Ejecución de comandos externos
  - Operación de salida
  - APIs

- Métodos de validación
  - Un alfabeto extendido (automatas, context-free-grammars)
  - ¨Escapando¨ Metacaracteres
  - Bloqueando y loggeando

Performance & Resultados obtenidos

- Tenemos un prototipo funcionando sobre PHP
- La solución protege determinística y automáticamente contra ataques conocidos y desconocidos
- No es necesario modificar las aplicaciones existentes
- La penalidad de performance es tolerable.

Privacidad

- Un lenguaje para describir políticas de seguridad
  - Asignando marcas en función del origen de la información
  - Validando en función del destino

- La misma herramienta implementa y valida en tiempo real la política de privacidad especificada.

Reflexiones

Gracias

Contacto:

ariel.waissbein@coresecurity.com

ariel.futoransky@coresecurity.com