TP2 Arquitectura Web ASP.NET MVC 5

—

SEGURIDAD

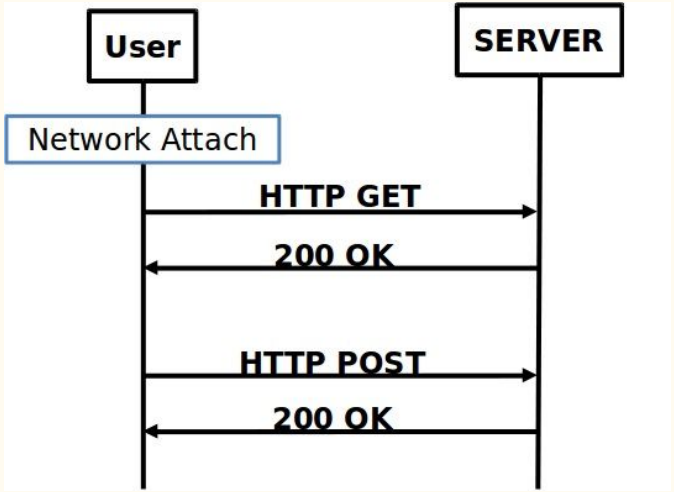
Julián Len - Matías Cadaval - Nicolás Len

Vulnerabilidades encontradas:

- HTTP: Mensajes no encriptados
- CSRF
- No era posible el cambio de password
- HTML Injection
- Upload infected files
- Acceso de bots
- Passwords inseguras
- Imágenes públicas
- Conversaciones (no) privadas

HTTP

Descripción: HTTP es el protocolo de comunicación utilizado entre el cliente y el servidor. El problema es que, la información enviada a partir de este protocolo no es encriptada.



HTTPS

Se le agrega a HTTP, el protocolo SSL para la comunicación entre el servidor y el cliente.

- Se negocian los algoritmos a utilizar durante la conexión
- Se autentican ambos entes (A través de certificados)
- Se genera un canal seguro para definir una Master Key
- Se derivan las claves necesarias a partir de la Master Key
- Se constata la integridad de todos los mensajes de intercambio de claves

A partir de ahora los mensajes estarán encriptados por estas claves.

HTTPS: Cómo lo usamos en nuestro proyecto.

Creamos nuestro propio certificado SSL y lo vinculamos con el proyecto, forzando al IIS a montar el sitio en la URL https://localhost apuntando a un puerto habilitado para SSL.

Al subirlo a un servidor remoto, se debería contratar un certificado de confianza.

Cross Site Request Forgery (CSRF)

Descripción:

Este ataque fuerza al navegador web de su víctima, validado en algún servicio, a enviar una petición indeseada.

Created by :- Saineshwar Bageri

USER LOGS IN TO BANK ACCOUNT

1

When User Click On Malicious Link It get Connected Bank Session

3

4

The Account is hacked Amount from User Account is Transfer to Hacker Account.

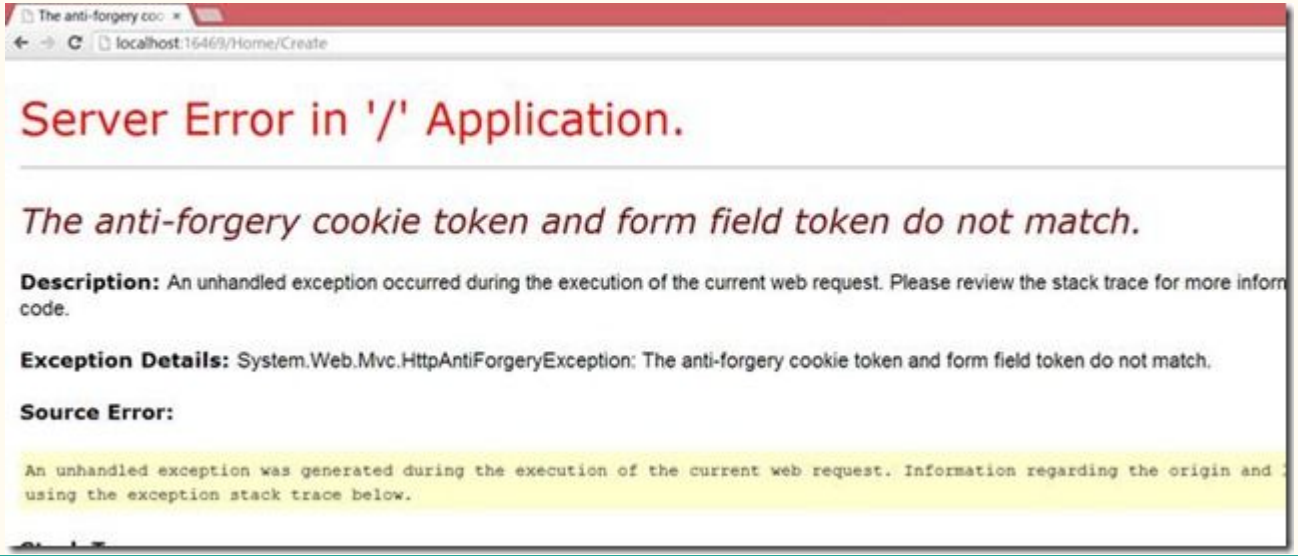
Mean while hacker will send an Email which contains Malicious link

2

CSRF: Solución

Solución:

Para esto hicimos uso de la propiedad AntiForgeryToken, que agrega una entrada en el formulario con una clave que el servidor provee. De esta manera, al recibir la petición, el servidor valida autenticidad verificando que recibió la clave que proveyó anteriormente.



Imágenes públicas

Descripción:

-Cuando un usuario envía una imagen, esta se guarda en el servidor y es accesible mediante una url. Esta url es pública, y cualquiera puede acceder a la imagen.

Solución:

-Se solicita la imagen mediante un request al servidor, en el que se verifica si el usuario solicitante tiene acceso a la misma. Si no tiene acceso, se arroja un error.

Cross-site scripting (XSS)

-Es un tipo de vulnerabilidad informática típica de páginas web. Permite a los atacantes inyectar código del lado del cliente, en las páginas vistas por otros usuarios. Si bien no hay una clasificación estándar de este tipo de vulnerabilidades, se pueden separar en dos grandes grupos:

Directa (persistentes): los datos provistos por el atacante son guardados por el servidor, y se reproducen permanentemente.

Indirectas (no persistentes): en general estas vulnerabilidades se deben a faltas de sanitización de parámetros. El código inyectado no persiste en el servidor, sino que en general se engaña a la víctima mediante algún link que cree que es seguro.

A continuación analizaremos algunas vulnerabilidades XSS que encontramos.

HTML Injection

Descripción:

-Los mensajes enviados entre usuarios se imprimen sin previamente verificar el uso de caracteres reservados de HTML. Esto permite inyectar código HTML en el chat del usuario que recibe el mensaje. Ejemplo: un link oculto en un mensaje

Solución:

-Codificamos los mensajes correctamente antes de imprimirlos (se reemplazan por HTML entities), previniendo cualquier tipo de inyección.

Enviar mensajes a conversaciones ajenas

Descripción:

-Editando el código JavaScript del lado del cliente, un usuario puede editar el id de la conversación y enviar un mensaje a una conversación ajena. El mensaje se guarda en la base de datos y queda de manera permanente en dicha conversación.

Solución:

-Se agregó la validación correspondiente del lado del servidor, que verifica si el usuario está enviando un mensaje a una conversación de la que forma parte.

Otras posibles vulnerabilidades resueltas

- Cambio de Contraseña
- Expiración de sesión
- Captcha de Google al registrar usuarios
- Strong passwords

ASP NET Identity + Entity Framework

—

Manejo de usuarios y base de datos

Entity Framework

Microsoft provee un framework ORM (Object-Relational Mapping) con el cual automatiza las actividades relacionadas a base de datos. Esto permite al desarrollador trabajar con datos relacionales usando objetos de dominio específico en vez de tener que escribir código de acceso a datos.

Si se usa con responsabilidad, se garantiza la protección frente a ataques SQL Inyection.

ASP.NET Identity

Sistema de miembros para todo tipo de aplicación ASP.NET.

Por defecto, ASP.NET Identity se encarga de manejar los usuarios y roles, donde se ocupa de almacenar la información en una base de datos. Garantiza seguridad de distintas formas. Por ejemplo, encripta las contraseñas y permite fácilmente restringir el acceso a Actions de Controllers, o a un Controller entero, a ciertos usuarios.

Trabajos a futuro

- Envío de mensajes por SignalR del lado del servidor (y no del cliente).
- Pruebas frente a DoS.
- Hacer test online de seguridad con la aplicación deployada.
- Chequeo de virus en archivos enviados.
- Verificación por mail.

Muchas gracias!
PREGUNTAS?