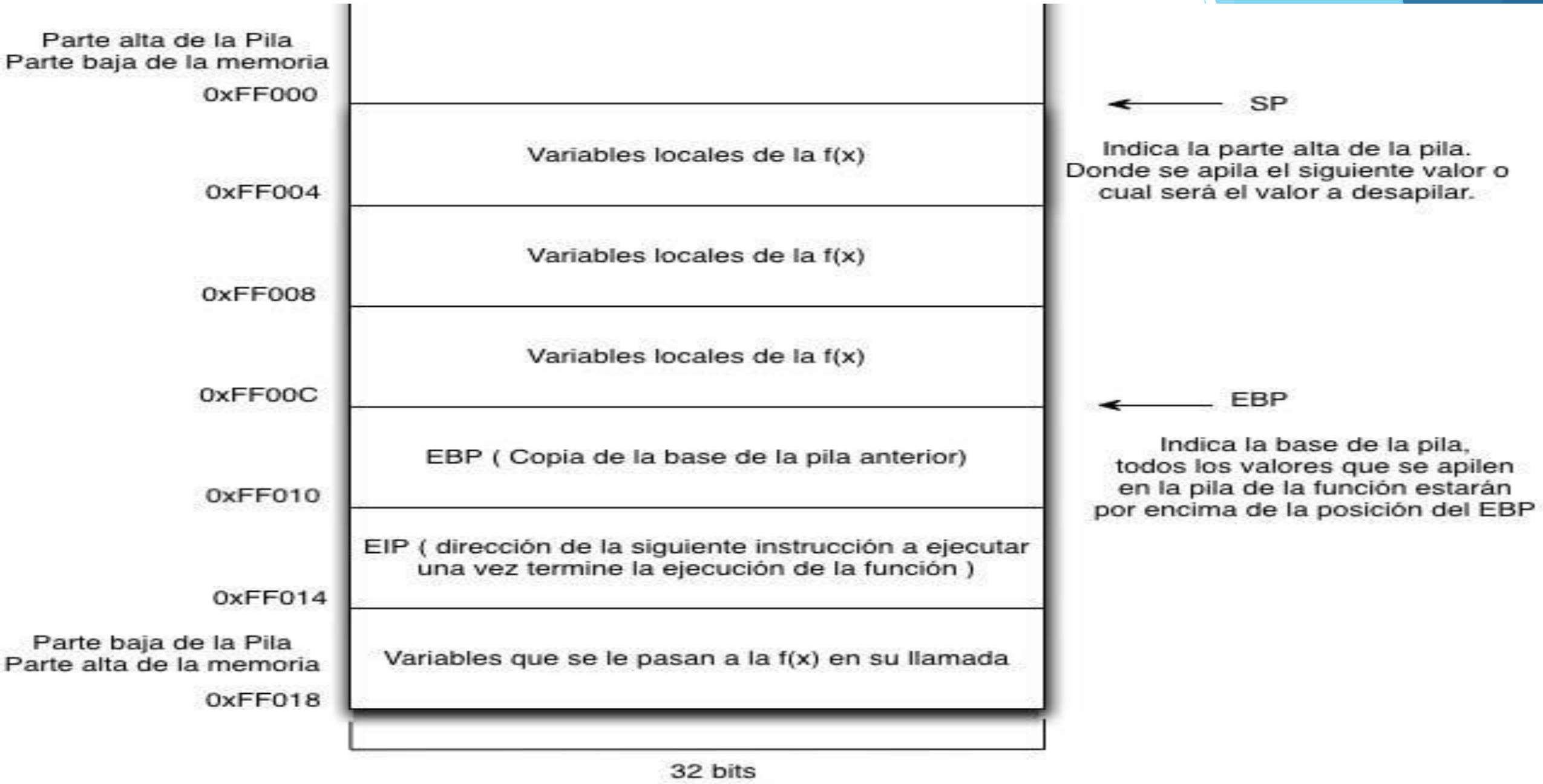
Programación de código seguro

GESI Almería

Distinción de vulnerabilidades

- Stack OverFlow y Heap OverFlow
  - Comunes en lenguajes y aplicaciones de escritorio
- TOP 10 OWASP
- Muchas otras

Stack OverFlow



Stack OverFlow: Ejemplo

```c
#include <string.h>
#include <stdio.h>
void func(char *arg)
{
        char nombre[32];
        strcpy(nombre, arg);
        printf("\nBienvenido a Linux Exploiting %s\n\n", nombre);
}
int main(int argc, char *argv[])
{
        if ( argc != 2 ) {
                printf("Uso: %s NOMBRE\n", argv[0]);
                exit(0);
        }
        func(argv[1]);
        printf("Fin del programa\n\n");
        return 0;
}
```

Heap OverFlow

- Evolución de la técnica Stack OverFlow
- Mayor complejidad
- Consiste en desbordar el Heap (Zona de memoria dinámica) -> (Datos del programa en tiempo de ejecución)

TOP 10 OWASP

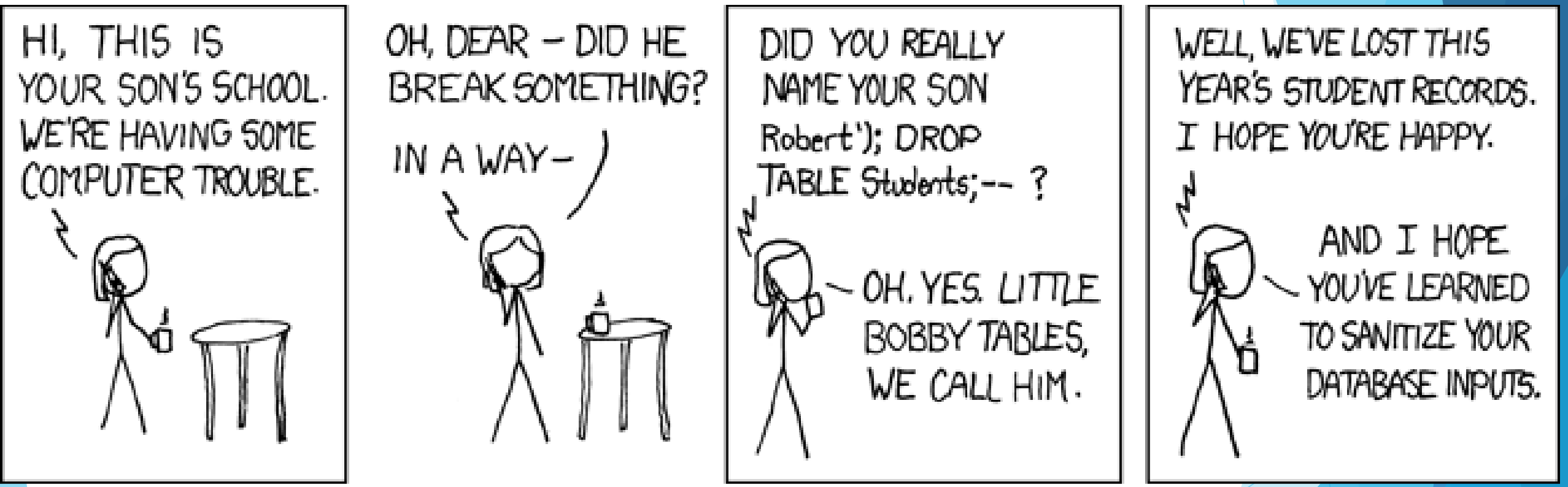
- Las 10 vulnerabilidades más comunes que afectan a aplicaciones web
- También puede afectar a un programa de escritorio clásico
- Se están extendiendo gracias a las app móviles y multiplataforma

Inyección

- Se manda información no fiable a un interprete
- No solo SQL: LDAP, SO, SMTP, etc…
- Fáciles de descubrir examinando el código
- Impacto severo

```
consulta := "SELECT * FROM usuarios WHERE nombre = '" + nombreUsuario + "';"
```

```
"Alicia'; DROP TABLE usuarios; SELECT * FROM datos WHERE nombre LIKE '%"
```


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Pérdida de autenticación

- Problemas de lógica en la implementación de gestión de sesiones (Vulnerabilidades en el cierre de sesión, gestión de contraseñas, pregunta secreta, etc…)
- Impacto severo
- Ejemplos:
  - hvp://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii
  - Las sesiones no expiran…

XSS (Secuencia de comandos en sitios cruzados)

- Vulnerabilidad más común
- Se envía una página con datos suministrados por usuarios sin validarse correctamente
- <script>alert(1)</script>

Referencia directa insegura a objetos

- Se accede directamente a un recurso
- Para acceder a ese recurso, no se controla el usuario de la aplicación, sus permisos, etc…
- **hvp://example.com/app/accountInfo?acct=notmyacct**

Inexistente control de acceso

- ¿Tienen todas las funcionalidades su autenticación correspondiente en el lado del servidor?
- ¿Los controles de autenticación se basan solamente en información solicitada u obtenida del usuario?

CSRF

- Obligar a un usuario a cometer una acción sin que se de cuenta
- Ej:
- **Utilizar un token aleatorio único por formulario y enlace, o utilizar chaptas que verifiquen la intervención de un usuario**

Componentes con vulnerabilidades conocidas

- ¿Eres consciente de las vulnerabilidades de los componentes y librerías que utilizas?
- ¿Te aseguras de que no existen backdoors en los componentes que estás utilizando? ¿Son seguros?

¡Conclusión!

- Punto común: Datos no controlados en las entradas de usuario
- NO SE PUEDE CONFIAR EN EL USUARIO. Si algo se puede romper, lo romperá.
- ¡¡El usuario es el hacker más peligroso!!

Consejos para hacer código seguro

- Michael Howard
  - Responsable de programas de seguridad en Microsoft
  - Libro: Writing Secure Code

1- Asumir la responsabilidad

- Las herramientas de análisis no detectan todas las vulnerabilidades
- Somos responsables de hacer código seguro
- Si alguien puede arreglar su código, es quien lo conoce: Su programador
- Producto seguro = Diseño seguro + Código seguro
- ¡Security By Design!

2- No confiar en los datos

- Toda entrada es nociva hasta que se demuestre lo contrario
- Según CVE: El 47% de las vulnerabilidades viene de confiar en los datos de entrada
- No bloquear lo malo: Solo dejar entrar lo esperado. Rechazar todo lo demás, en especial lo sospechoso. Ayudarse de expresiones regulares.

3- Análisis del modelo de amenazas

- TMA o Análisis del modelo de amenazas
- Ayuda a determinar los riesgos de seguridad de nuestra aplicación
- Indica como y por donde puede aparecer un posible ataque

4- Ir un paso por delante

- Vigilar siempre la aparición de nuevas técnicas y nuevas amenazas
- Listas de correo como: Bugtraq

5- Pruebas aleatorias

- Probar aleatoriamente datos en las entradas
- MS prueba su software con 100.000 iteraciones de baterías de entradas incorrectas

6- No escribir código no seguro

- Prohibido el uso de más de 500 funciones en C y C++ por no ser seguras. (Ej: Print)
- Reutilizar código que funciona y es seguro
- PHP: Explode, Split, StreamScandir, parse de los prepared staments

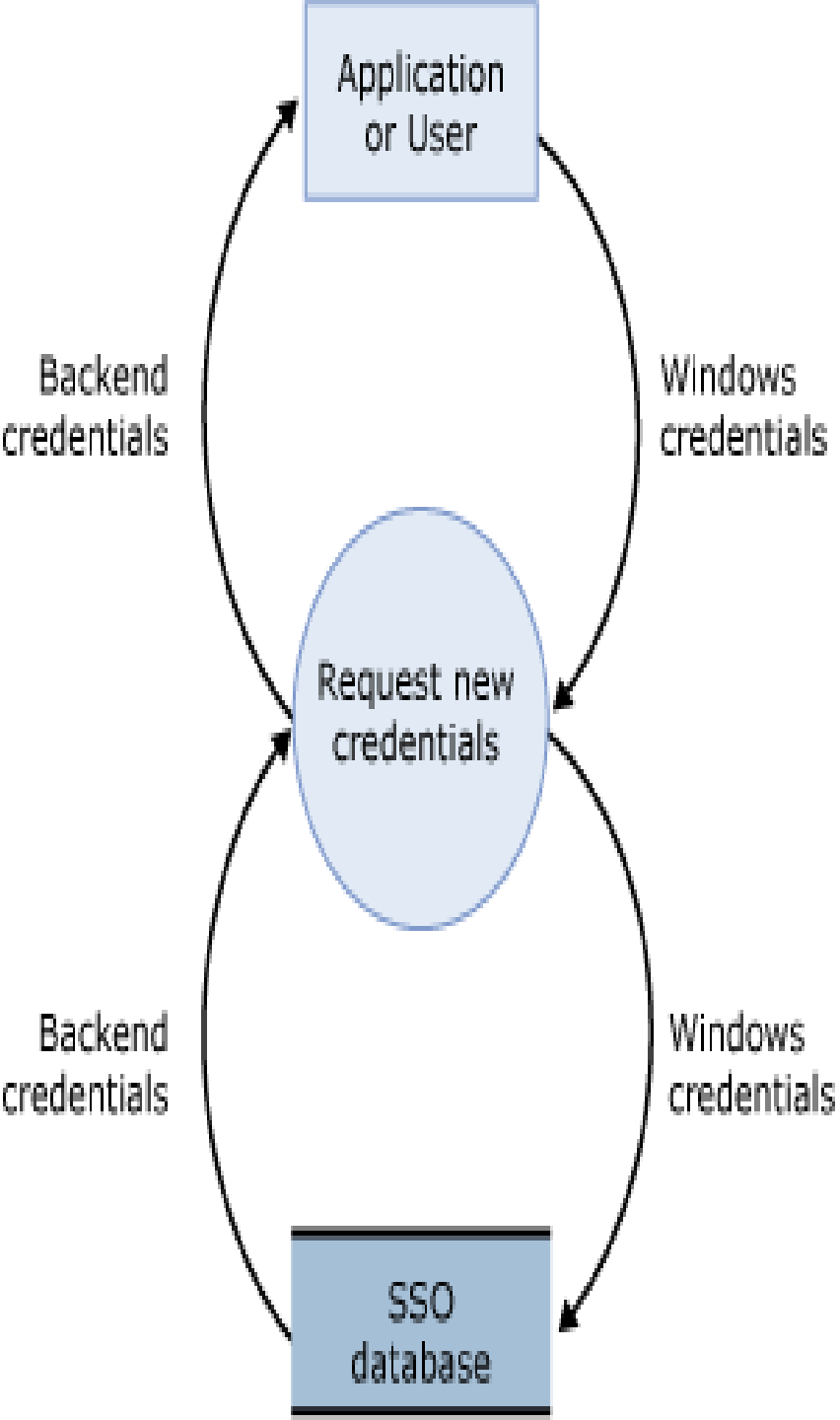
7- Asimetría estratégica

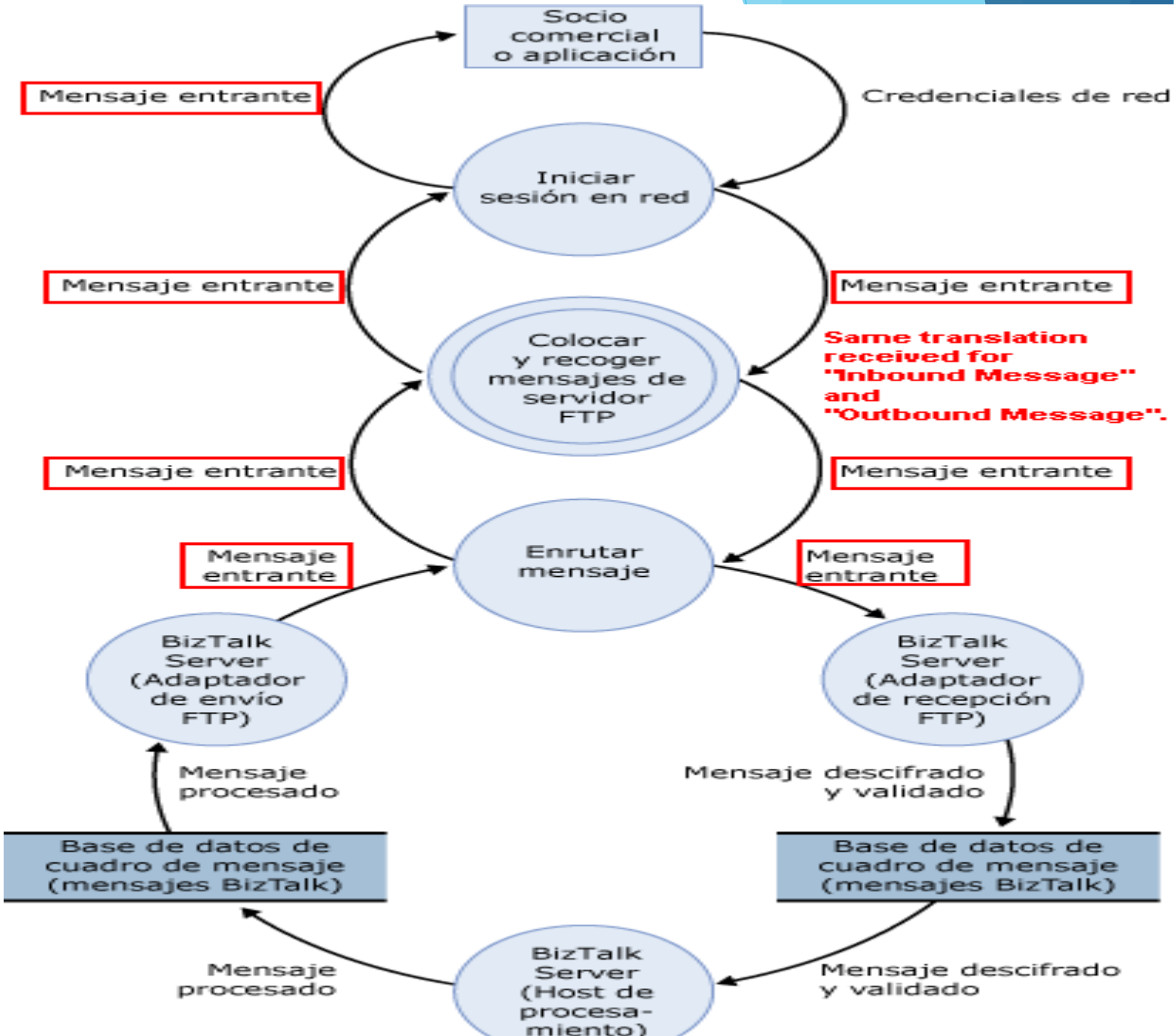
- El atacante no está limitado en el tiempo:
- Hay que repasar el código antiguo y mantenerlo en los niveles de calidad y seguridad actuales

TMA

- 1- Recopilar información
  - Usuarios y casos de uso
  - Crear DFD: Diagramas de flujo de datos

- 2- Analizar el modelo de amenazas
  - Identificar entradas, limites de confianza, y flujo de una entrada desde su primera hasta su última aparición.
  - Anotar activos implicados.


Application or User
Backend credentials
Windows credentials
Request new credentials
Backend credentials
Windows credentials
SSO database


Socio comercial o aplicación
Mensaje entrante
Credenciales de red
Iniciar sesión en red
Mensaje entrante
Mensaje entrante
Colocar y recoger mensajes de servidor FTP
Same translation received for "Inbound Message" and "Outbound Message".
Mensaje entrante
Mensaje entrante
Mensaje entrante
Enrutar mensaje
Mensaje entrante
BizTalk Server (Adaptador de envío FTP)
BizTalk Server (Adaptador de recepción FTP)
Mensaje procesado
Mensaje descifrado y validado
Base de datos de cuadro de mensaje (mensajes BizTalk)
Base de datos de cuadro de mensaje (mensajes BizTalk)
Mensaje procesado
BizTalk Server (Host de procesa-miento)
Mensaje descifrado y validado

TMA

- 3- Catalogar las entradas
  - Suplantación de identidad
  - Manipulación de datos
  - Rechazo
  - Revelación de información
  - Denegación de servicio
  - Elevación de privilegios

TMA

- 4- Listar las amenazas

Amenaza	Descripción	Activo	Impacto
Un usuario malintencionado podría pueden manipular los mensajes en su tránsito del servidor de BizTalk Server al servidor SQL Server y viceversa.	Para la comunicación entre el adaptador de SQL y el servidor SQL Server se emplea texto sin cifrar.	Cuerpo del mensaje	Suplantación de identidad Manipulación de datos

TMA

- 5- Analizar las amenazas, documentarse sobre las técnicas de explotación y mitigaciones. Implementar mitigaciones.
- 6- Repetir iteración del análisis con cierta continuidad

¿Qué más podemos encontrar?

- OWASP:
  - Code Review Guide
  - Development Guide
  - Testing Guide
  - Standard ASVS (Application Security Verification Standard)