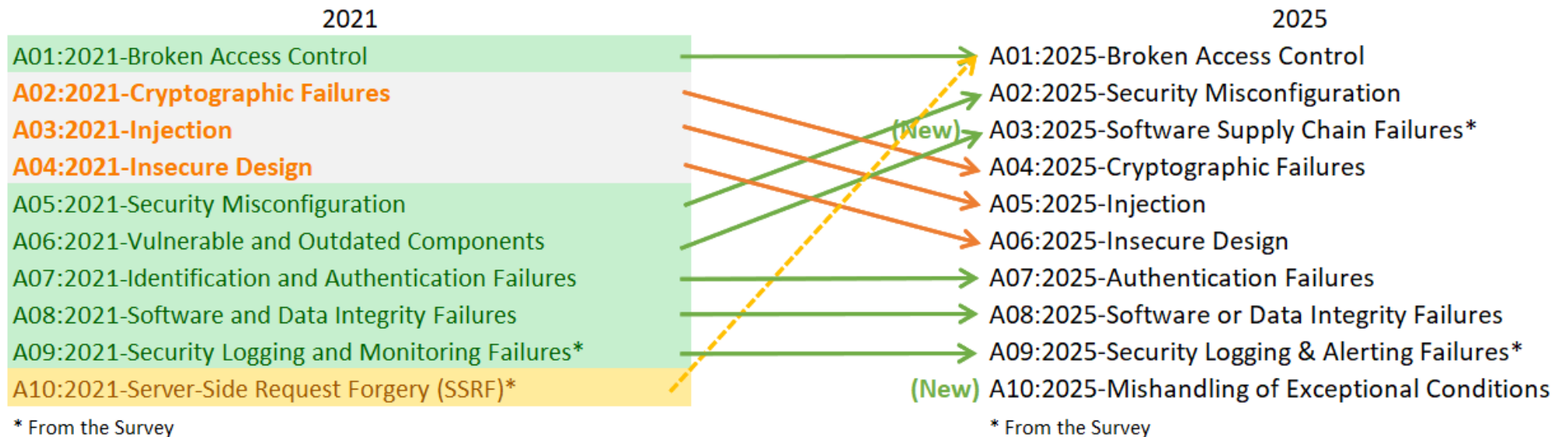


# Web Application Attack

Le applicazioni web sono in grado di fornire risposte (informazioni) alle richieste dei visitatori grazie all'uso dei database. Se l'applicazione risulta essere vulnerabile l'intera base dati sarà esposta a rischio.

## Gli attacchi più frequenti

Top 10 owasp: <https://owasp.org/www-project-top-ten/>

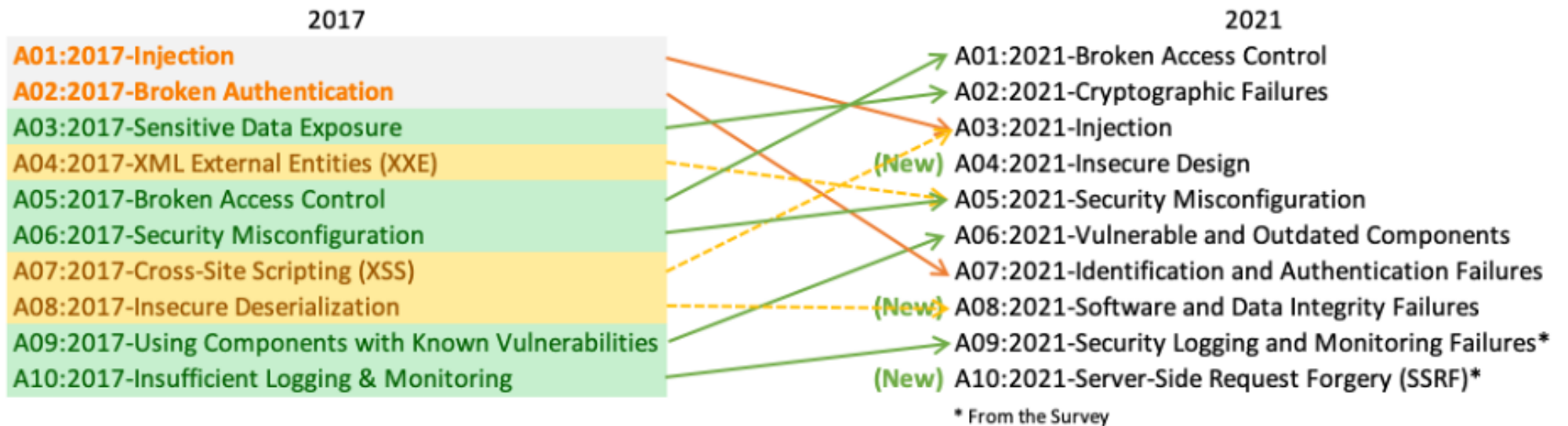


# Web Application Attack

Le applicazioni web sono in grado di fornire risposte (informazioni) alle richieste dei visitatori grazie all'uso dei database. Se l'applicazione risulta essere vulnerabile l'intera base dati sarà esposta a rischio.

## Gli attacchi più frequenti

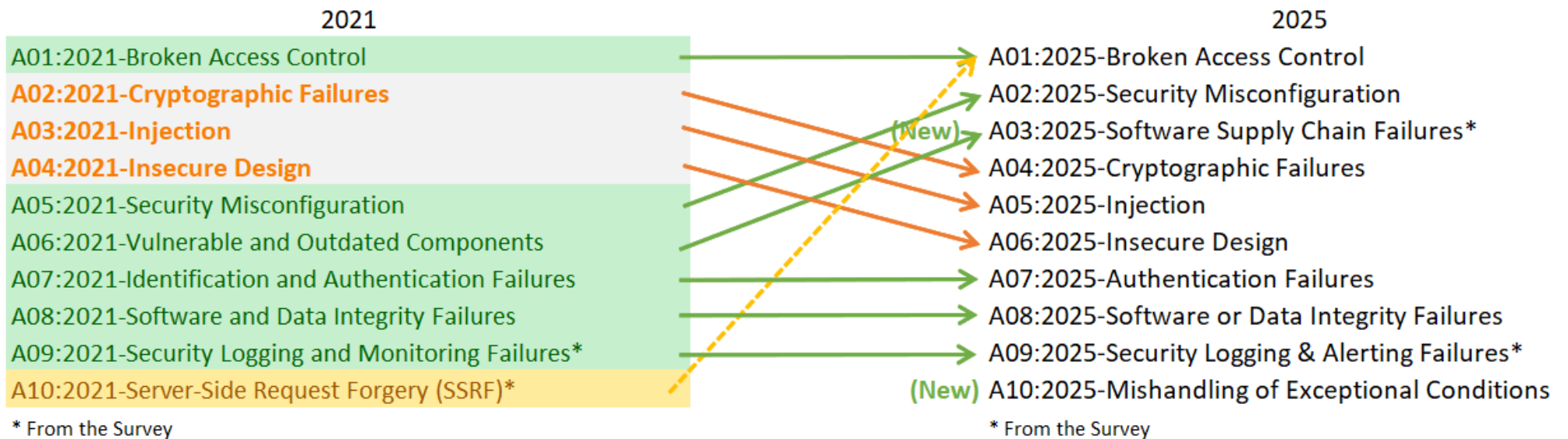
Top 10 owasp: <https://owasp.org/www-project-top-ten/>



# Broken Access Control

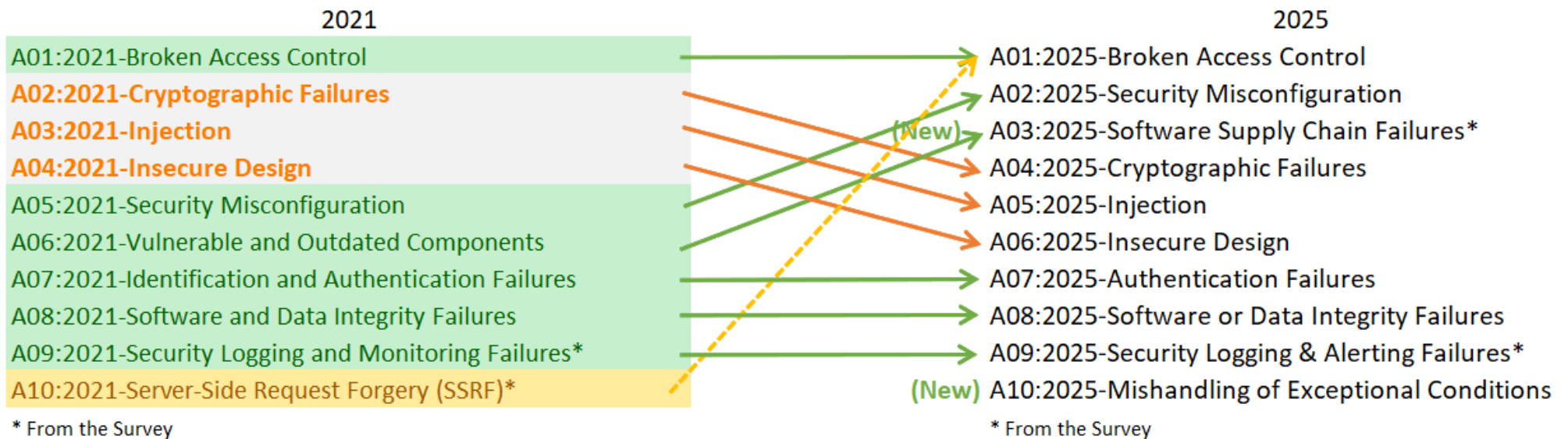
Il controllo degli accessi impone una politica tale che gli utenti non possano agire al di fuori delle autorizzazioni previste.

Gli errori portano alla divulgazione non autorizzata di informazioni, alla modifica o alla distruzione di tutti i dati o all'esecuzione di una funzione al di fuori dei limiti dell'utente.



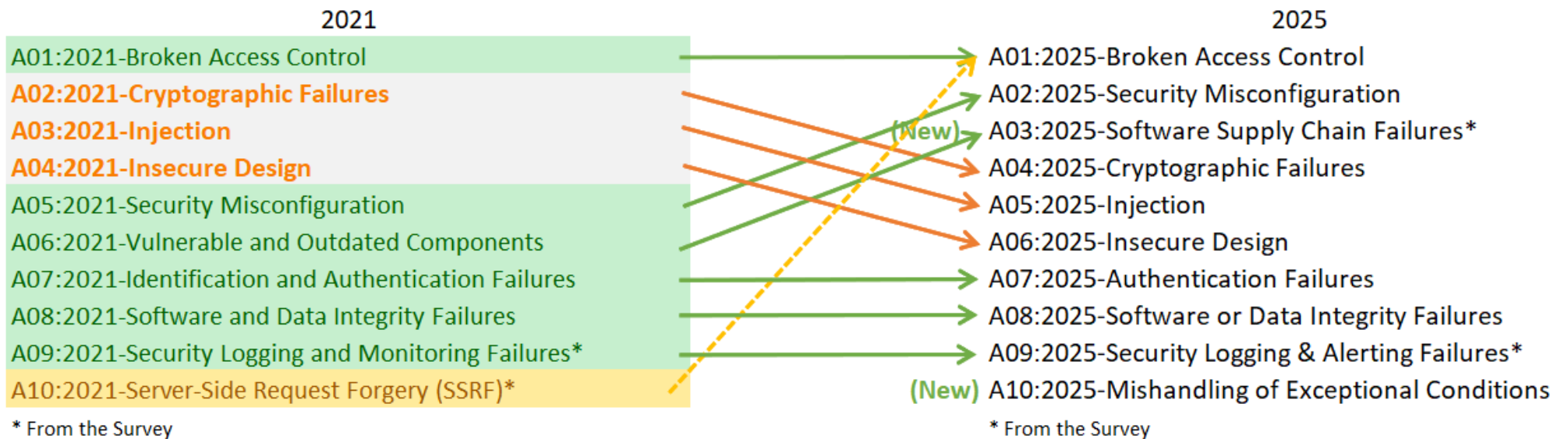
# Security misconfiguration

Molto insidiosa (salita dal quinto al secondo posto) ma anche molto facile da prevenire (ma si trova al secondo posto!) -> FATTORE UMANO!!



# Vulnerable/Outdated components

Molto insidiosa (salita dal sesto al terzo posto) ma anche molto facile da prevenire (ma si trova al terzo posto!) -> FATTORE UMANO!!



# Sensitive Data Exposure

La priorità dello sviluppatore è quella di produrre un'applicazione funzionante, la sicurezza è (quasi) sempre pianificata come step successivo ed alla fine dimenticata, ignorata o fatta male a discapito della protezione dei dati e dei suoi utenti.

## Un esempio

- API token esposti nel codice sorgente
- Informazioni sensibili trasmesse o memorizzate in chiaro
- Credenziali deboli
- Cartelle annidate o sottodomini dimenticati



Index of /admin/backup

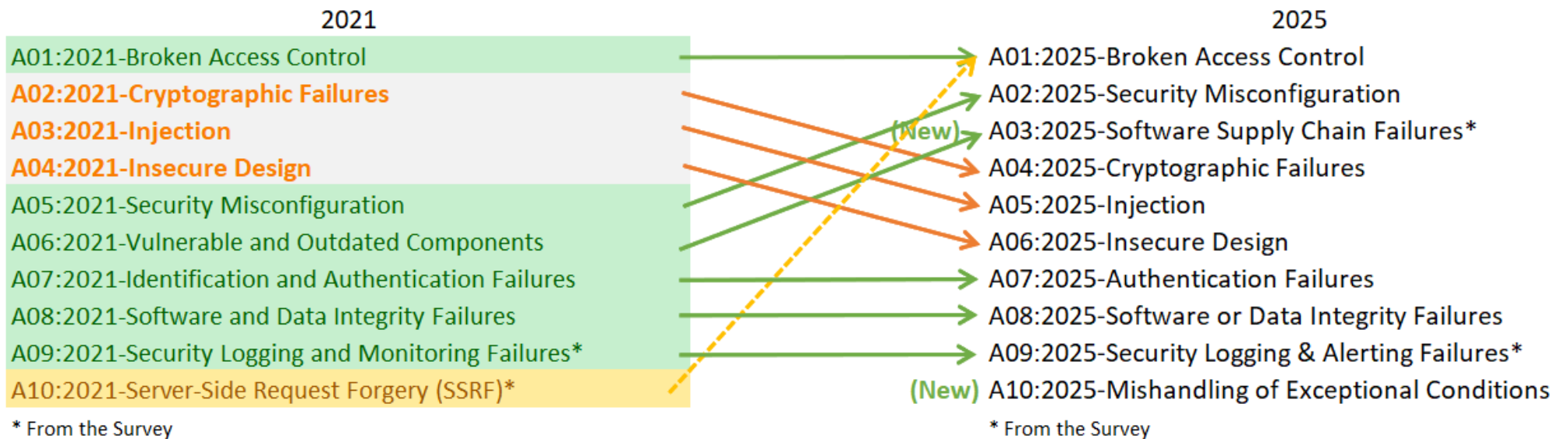
Name	Last modified	Size	Description
<a href="#">Parent Directory</a>			
<a href="#">ETP_la.log</a>	2020-04-27 09:20	635K	
<a href="#">database_connect.php</a>	2020-04-27 09:20	300	
<a href="#">db_dump.sql</a>	2020-04-27 09:21	968K	
<a href="#">old_pass.txt</a>	2020-04-27 09:22	6.3K	

Apache/2.4.43 (Ubuntu) OpenSSL/1.1.1g PHP/7.4.5 Server at 127.0.0.1 Port 80

# Errori di crittografia

Precedentemente noto come «esposizione di dati sensibili», che è più un sintomo generale che una causa principale, l'attenzione è rivolta ai problemi correlati alla crittografia (o alla sua mancanza).

Errori portano all'esposizione di dati sensibili.

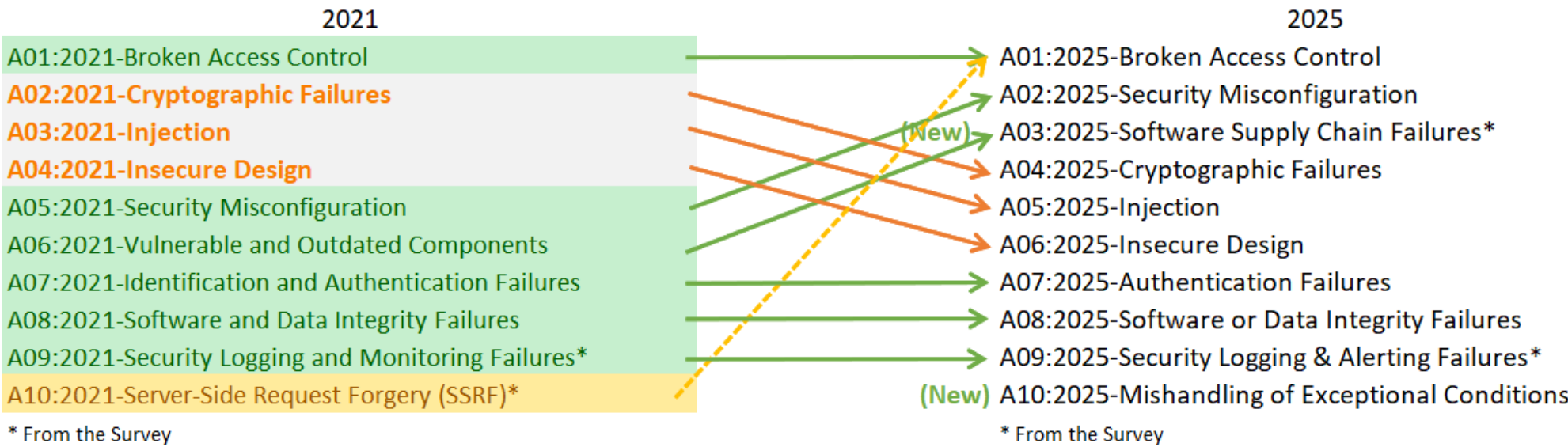




# Command Injection

Attacco il cui obiettivo è l'esecuzione indiscriminata di comandi su un host tramite una applicazione vulnerabile.

Questo attacco risulta possibile quando un'applicazione trasmette dati non sicuri forniti dall'utente.





# SQL injection

I dati passati in input da un utente malintenzionato possono interferire con le query che l'applicativo effettua al proprio database e di conseguenza restituire informazioni senza adeguata autorizzazione.

## Un esempio

GET: <https://insecure-hospital.com/progetti?categoria=covid>

SQL: SELECT \* FROM progetti WHERE categoria = 'covid' AND visibile = 1

**visibile = 1** mostra solo i progetti che possono essere visibili al pubblico

GET: <https://insecure-hospital.com/progetti?categoria=covid'-->

SQL: SELECT \* FROM progetti WHERE categoria = 'covid'--' AND visibile = 1

-- commento in SQL

SQL: SELECT \* FROM progetti WHERE categoria = 'covid'

restituirà in output tutti i progetti, compresi quelli con flag **visibile = 0**

# Cross-Site Scripting (XSS)

Attacchi che sfruttano le debolezze di sicurezza insite nel codice dell'applicazione web per eseguire Javascript lato client.

L'input utente viene incluso nella pagina **senza validarne** il contenuto, il codice arbitrario viene eseguito consentendo all'aggressore di controllare il browser oppure, ove possibile, di utilizzare la sessione della vittima nel contesto dell'applicativo.

## Un esempio

GET: `https://insecure-hospital.com/search?text=covid`

RES: `<p>Search: covid</p>`

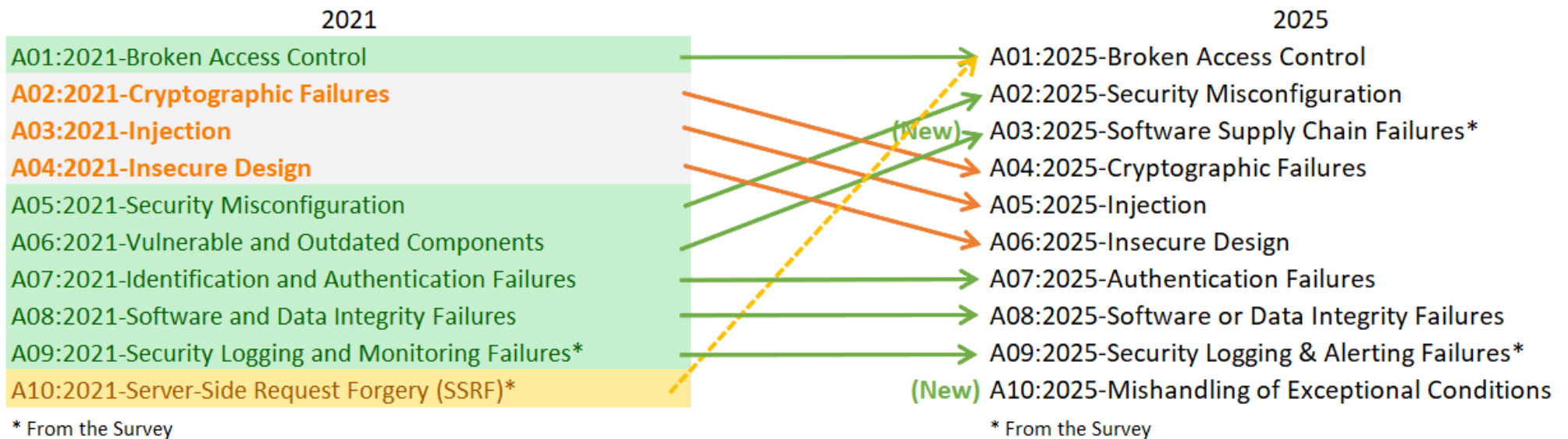
GET: `https://insecure-hospital.com/search?text=<script>alert(document.cookie)</script>`

RES: `PHPSESSID=qd66lO8djbtu823gr82copvbt4`



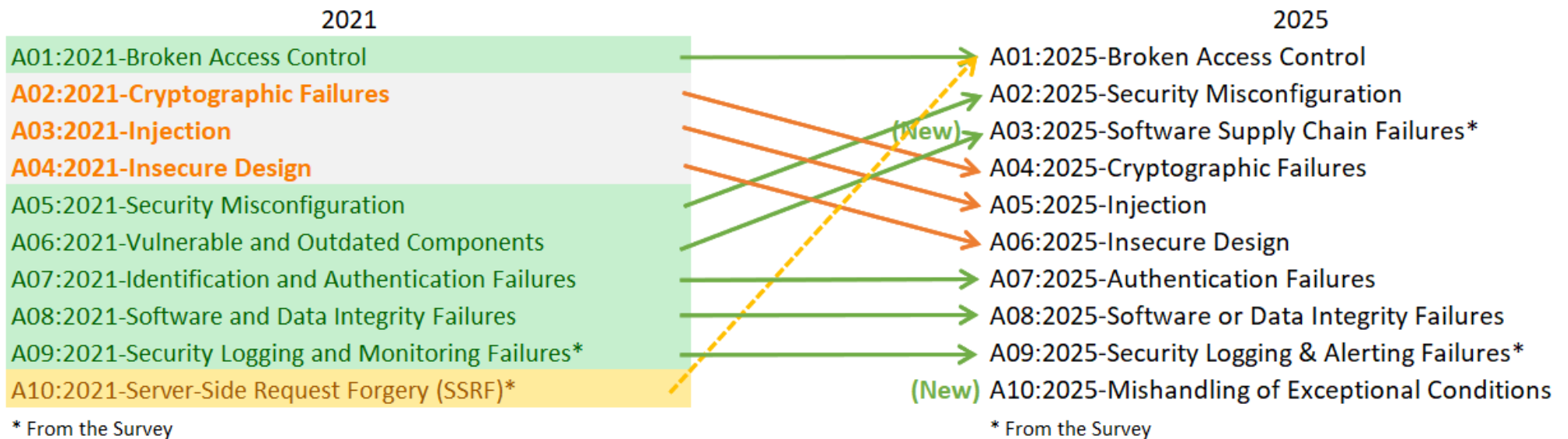
# Insecure Design

Problema legato alla scarsa o mancante profilazione del rischio  
Non si risolve con una implementazione perfetta.



# Errori di autenticazione/identificazione

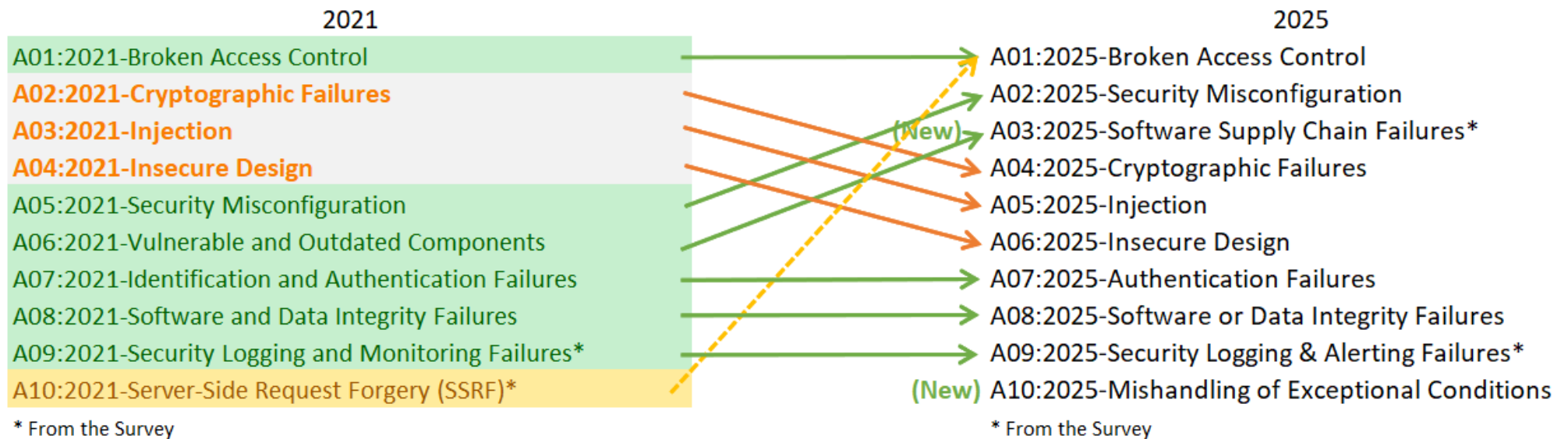
La conferma dell'identità dell'utente, l'autenticazione e la gestione della sessione sono fondamentali per proteggersi dagli attacchi correlati all'autenticazione. Situazione migliorata grazie alla «forzatura» nell'uso di alcune soluzioni.



## Errori di integrità/affidabilità di software e dati

Gli errori di integrità di software e dati sono correlati a codice e infrastrutture che non proteggono dalle violazioni di integrità.

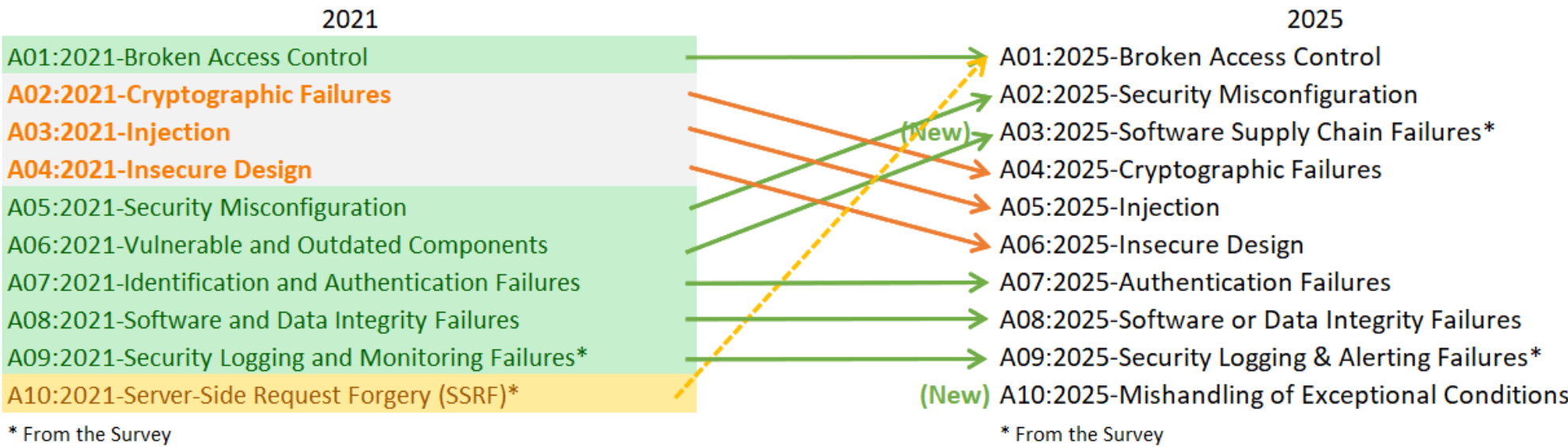
Un esempio di ciò è quando un'applicazione si basa su plugin, librerie o moduli da fonti non attendibili





# Errori di monitoraggio

Sono problemi legati alla scarsa o mancante attività di logging/monitraggio degli eventi  
Costo (percepito) molto elevato



## Gestione errata/mancata di condizioni eccezionali

New entry nella classifica

- si verifica quando il software non previene, rileva o gestisce correttamente situazioni impreviste, causando crash, comportamenti anomali e possibili vulnerabilità.
- può generare problemi di sicurezza (overflow, bug logici, race condition, violazioni di autenticazione) compromettendo riservatezza, integrità e disponibilità del sistema.

