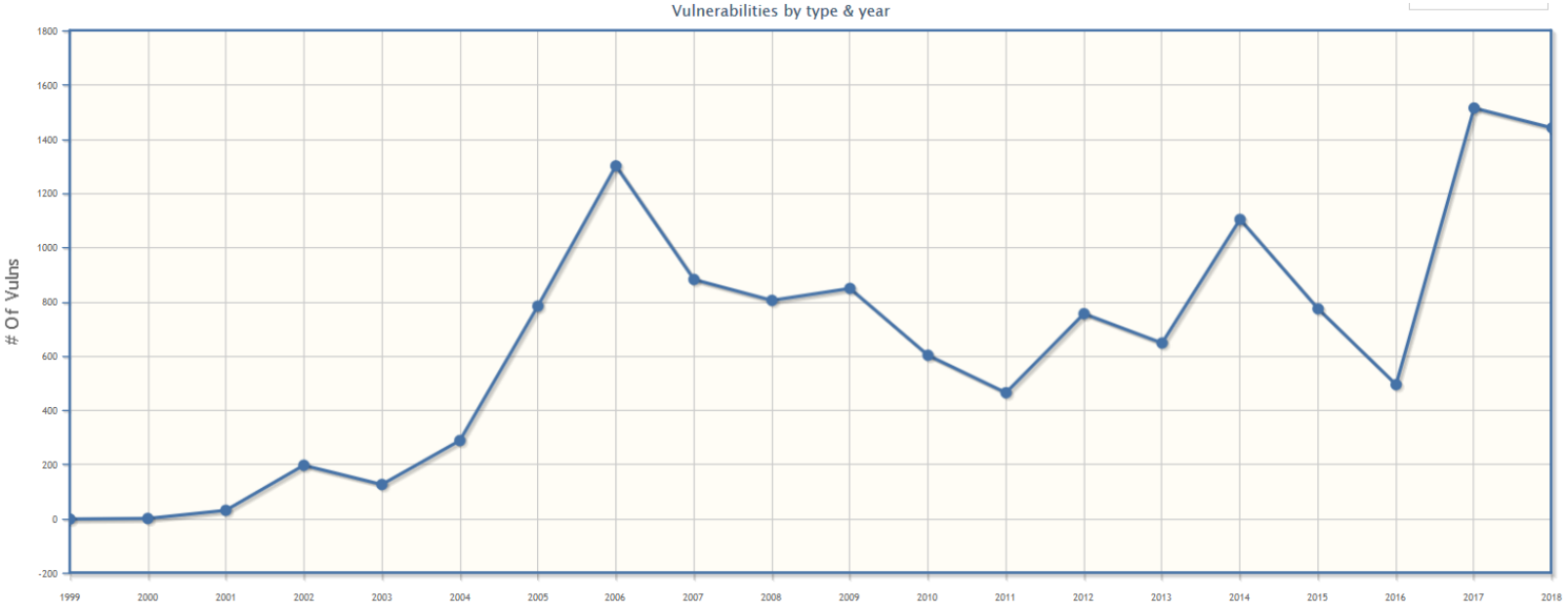
IT/Dev CONNECTIONS

WEB APPLICATION SECURITY TOMORROW

Christian Wenz
CEO
Actition GmbH

Delivered by KNect365 TMT an Informa business

Cross-Site Scripting (XSS)

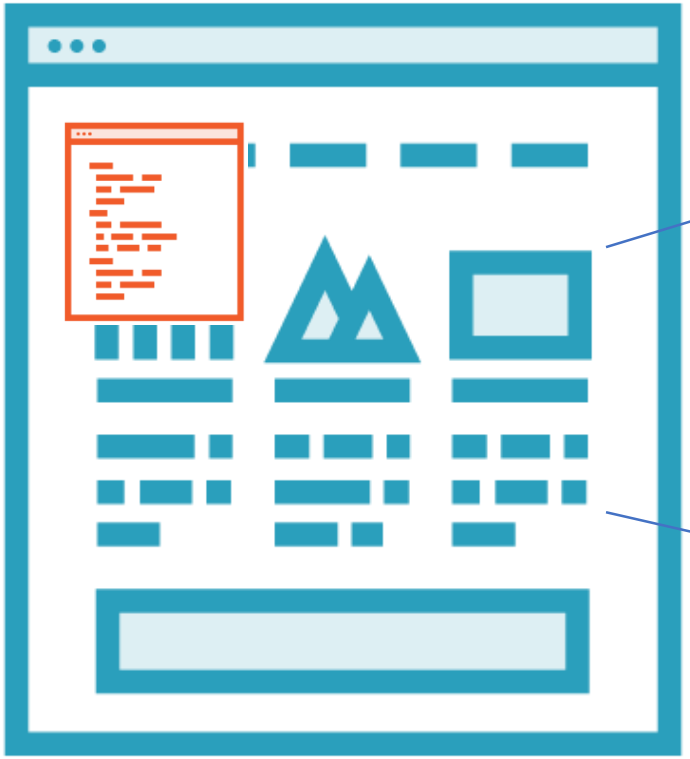


https://www.cvedetails.com/vulnerabilities-by-types.php

Types of XSS

- DOM-based (type 0): Everything happens in the client
- Persistent (type 1): Server stores malicious input
- Non-persistent (type 2): Server echoes back malicious input

Same-Origin Policy



http://example.com:80/page.php

http://example.com/script.js

https://code.jquery.com/
jquery-x.y.z.min.js

Protocol

Domain

Port

Content Security Policy

- Content-Security-Policy: default-src 'self';

Template Injection

- Many „SPA“ libraries evaluate code based on templates in the DOM
- This might enabe injecting attacks

AngularJS Template Injection

- Angular < 1.6 has a sandbox that checks expressions and transform them
- 1 + 2 turns into:

```
var fn = function(s, l, a, i) {
    return plus(1, 2);
};
return fn;
```

- Many, many sandbox escapes: https://portswigger.net/blog/xss-without-html-client-side-template-injection-with-angularjs
- In Angular 1.6, the sandbox was removed

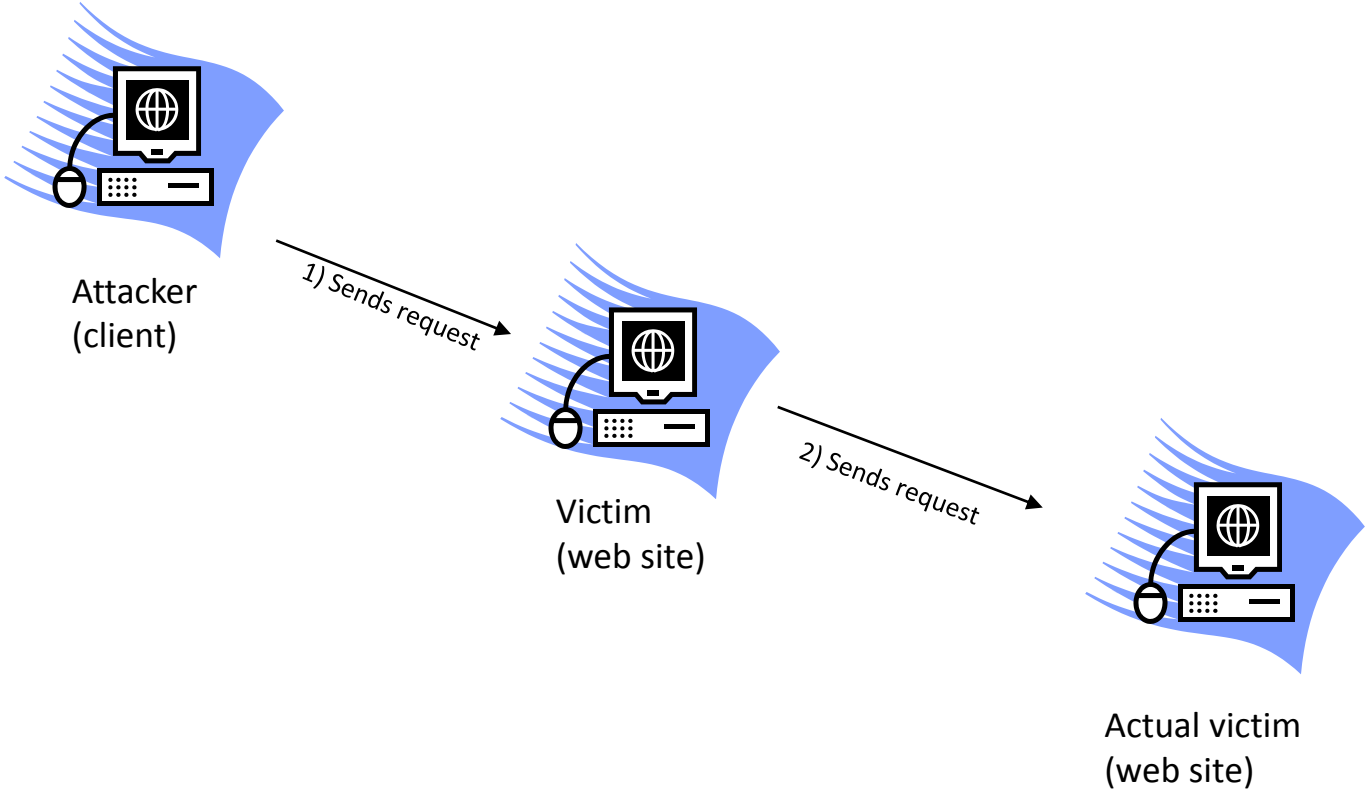
XXE (XML External Entities)

- XML supports entities
- Entities may reference external files
- If an attacker may supply the XML for the application, this can be problematic

XXE Defense

- Java: depends on the API
  - Example: Spring MVC – https://github.com/spring-projects/spring-framework/pull/317/commits/2843b7d2ee12e3f9c458f6f816befd21b402e3b9
- .NET: secure by default since 4.5.2
- OWASP Cheat Sheet: https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet

Server-Side Request Forgery



Insecure Deserialization

- Rule #1: Validate input
- If possible, limit the expected deserialization types
- https://www.owasp.org/index.php/Deserialization_Cheat_Sheet
- Paper based on .NET: https://speakerdeck.com/pwntester/attacking-net-serialization
- Payload generator: https://github.com/pwntester/ysoserial.net

Abusing CORS

- Cross-Origin Resource Sharing
- A „legal“ way to circumvent the Same-Origin Policy (SOP)
- Lazy developers might use *, or null
- Crazy developers use the Origin header for authorization

Mass Assignment

- The curse of model binding

ASP.NET vs. Mass Assignment

- Allow lists/deny lists
  - [Bind(Include="FirstName,LastName")] only binds the specified properties
  - [Bind(Exclude="DateOfBirth")] does not bind the specified properties

Phishing via External Links

- When opening a new browser window or tab, JavaScript provides access to the opening tab/window

- Same Origin Policy prevents DOM access, but window access is possible

Clickjacking

- Users click where they do not want to click
- Generator: http://samy.pl/quickjack/
- Use X-FRAME-OPTIONS header (and, soon, frame-ancestors CSP directive)

Regex DoS

- ASP.NET MVC validation of email addresses and phone numbers

```
[AttributeUsage(AttributeTargets.Property | AttributeTargets.Field | AttributeTargets.Parameter, AllowM
public sealed class EmailAddressAttribute : DataTypeAttribute {

    // This attribute provides server-side email validation equivalent to jquery validate,
    // and therefore shares the same regular expression.  See unit tests for examples.
    private static Regex _regex = new Regex(@"^((([a-z]|\d|[!#\$%&'\*\+\-\/=\?\^_`{\|}~]|[\u00A0-\uD7FF

    public EmailAddressAttribute()
        : base(DataType.EmailAddress) {

        // DevDiv 468241: set DefaultErrorMessage not ErrorMessage, allowing user to set
        // ErrorMessageResourceType and ErrorMessageResourceName to use localized messages.
        DefaultErrorMessage = DataAnnotationsResources.EmailAddressAttribute_Invalid;
    }

    public override bool IsValid(object value) {
        if (value == null) {
            return true;
        }

        string valueAsString = value as string;
        return valueAsString != null && _regex.Match(valueAsString).Length > 0;
    }
}
```

- Fixed ☺

Thank you!

- Thank you!
- info@christianwenz.de
- @chwenz