Comenius university in Bratislava
Faculkty of mathemmatics, physics and computer science

Trusted Types and bundlers

(School project for IT security)

2020
Emanuel Tesař

Abstract

Keywords:

Contents

Chapter 1

Cross site scripting

Cross site scripting (abbr. XSS) is one of the most prevalent security vulnerabilities and the most common one when talking about web applications. When we take into the account only the last year (year 2020), XSS has been ranked on 7th place in the OWASP top 10 vulnerabilities ranking. [1]. When, we look at the bounty programs only, and the money rewarded for each security vulnerability, we can see that XSS is the winner. Total bounty just for XSS was more than $4,2$ million USD [3].

Citing very well written introduction from [6].

> XSS attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script (for example encoded in URL), to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it [6].
>
> An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page [6].
>
> *(Be sure to read the cited article for more information and links [6]. The above paragraphs were also largely copied from that very well written article.)*

1.1 Types of XSS

There are many types of XSS, although there is no single finite list of XSS types to rule them all [7]. Most experts distinguish at least between non-persistent *(reflected)* and persistent *(stored)*. There is also a third category *DOM based XSS* which will be explained in more depth as this is the threat model under which Trusted Types operate.

- Stored - The injected script is permanently saved in server database. The client *(user's browser)* will then ask the server for the requested page and the response from the server will contain the malicious script.
- Reflected - Typically delivered via email or a neutral web site. occur when a malicious script is reflected off of a web application to the victim's browser [2].
- DOM based - The vulnerability appears in the DOM - by executing some malicious code. In reflective and stored Cross-site scripting attacks you can see the vulnerability payload in the response page but in DOM based cross-site scripting, the HTML source code and response of the attack will be exactly the same. You can only observe the change at runtime.

1.2 Modern classification of XSS

The classification in the *Section 1.1 - Types of XSS* was created many years back and a lot has since changed. The web got more secure and modern frameworks try to enforce best security practices for developers using them. However, web has since evolved rapidly *and still evolves* while still mostly preserving the backward compatibility with original JS spec - meaning you could still browse the web page created 20 years ago with subtle differences *(Compare that with running Android app created for version 4.1 Jelly Bean created in mid 2012 on Android 11 released 2020 - good luck with that)*.

The previous classification is not ideal, because the categories overlap. Citing from [4]:

> You can have both Stored and Reflected DOM Based XSS. You can also have Stored and Reflected Non-DOM Based XSS too, but that's confusing, so to help clarify things, starting about mid 2012, the research community proposed and started using two new terms to help organize the types of XSS that can occur:

Instead what they propose is just two categories (again, fully citing [4]):

- Server XSS - Server XSS occurs when untrusted user supplied data is included in an HTTP response generated by the server. The source of this data could be from the request, or from a stored location. As such, you can have both Reflected Server XSS and Stored Server XSS. In this case, the entire vulnerability is in server-side code, and the browser is simply rendering the response and executing any valid script embedded in it.

- Client XSS - Client XSS occurs when untrusted user supplied data is used to update the DOM with an unsafe JavaScript call. A JavaScript call is considered unsafe if it can be used to introduce valid JavaScript into the DOM. This source of this data could be from the DOM, or it could have been sent by the server (via an AJAX call, or a page load). The ultimate source of the data could have been from a request, or from a stored location on the client or the server. As such, you can have both Reflected Client XSS and Stored Client XSS.

Just for completeness, DOM based XSS is a subset of client XSS. The source of the data is client side only. And again, study the full article *(together with further references)* for more information [4].

1.3 Common examples of XSS

The basic example of XSS *and probably the easiest to understand* is to take a page which interpolates data from an URL. This is a common practice - you browse a site, find something of value and you want to share it with your friend. Nothing easier, you just copy the URL and they see the same content *or at least similar* to you. This basic example is common for nearly all shopping sites, tourism agencies, accommodation services etc...

This type of attack can be easily demonstrated with the following example page:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>URL XSS</title>
  </head>
  <body>
    <p>Try searching something:
      <i> (for example "<img src=x onerror=alert(1) />")</i>
    </p>
    <input id="query" type="text" />
```

```
    <button id="submit">Search</button>

    <p>You have searched for</p>
    <p id="attack-target"></p>
  </body>
  <script>
    window.addEventListener('DOMContentLoaded', () => {
      const urlParams = new URLSearchParams(location.search)
      document.getElementById('attack-target').innerHTML =
          urlParams.get('query')
    });

    document
      .getElementById('submit')
      .addEventListener('click', () => {
        const query = document.getElementById('query').value
        location.replace(
          `${location.pathname}?query=${encodeURIComponent(query)}`
        );
      })
  </script>
</html>
```

This is very small example and XSS is apparent, but once the project is composed of tens of thousands lines and many dependencies, such mistake can easily sneak through. In this case, if you execute this HTML code in the browser and you try searching for something, the result will be interpolated in the page. This allows the attacker to *prepare an evil URL* which they can then send to benign users to exploit.

1.4 Consequences of XSS

So far, we have talked about many types of XSS, saw a basic example and are aware that there are many more. What we haven't covered are the consequences, which can be caused by these attacks.

TODO: write me

1.5 Protection

TODO: write me

Chapter 2

Introduction to Trusted Types

In this chapter we are going to introduce the concept of Trusted Types (abbreviated as TT). V tejto kapitole si povieme niečo o jadre práce a o jej členení. V zdrojovom kóde v súbore `kapitola.tex` nájdenie ukážky použitých príkazov LaTeXu potrebných na písanie nadpisov a podnadpisov a číslovaných a nečíslovaných zoznamov. Zvyšok textu tejto kapitoly je prebratý zo smernice o záverečných prácach [**?**, článok 5].

Jadro je hlavná časť školského diela a člení sa na kapitoly, podkapitoly, odseky a pod., ktoré sa vzostupne číslujú.

2.1 Členenie

Členenie jadra školského diela je určené typom školského diela. Vo vedeckých a odborných prácach má jadro spravidla tieto hlavné časti:

- súčasný stav riešenej problematiky doma a v zahraničí,
- cieľ práce,
- metodika práce a metódy skúmania,
- výsledky práce,
- diskusia.

2.1.1 Súčasný stav

V časti súčasný stav riešenej problematiky doma a v zahraničí autor uvádza dostupné informácie a poznatky týkajúce sa danej témy. Zdrojom pre spracovanie sú aktuálne publikované práce domácich a zahraničných autorov. Podiel tejto časti práce má tvoriť približne 30 % práce.

2.1.2 Cieľ práce

Časť cieľ práce školského diela jasne, výstižne a presne charakterizuje predmet riešenia. Súčasťou sú aj rozpracované čiastkové ciele, ktoré podmieňujú dosiahnutie cieľa hlavného.

2.1.3 Metodika práce a metódy skúmania

Časť metodika práce a metódy skúmania spravidla obsahuje:

1. charakteristiku objektu skúmania,
2. pracovné postupy,
3. spôsob získavania údajov a ich zdroje,
4. použité metódy vyhodnotenia a interpretácie výsledkov,
5. štatistické metódy.

2.1.4 Výsledky práce a diskusia

Časti výsledky práce a diskusia sú najvýznamnejšími časťami školského diela. Výsledky (vlastné postoje alebo vlastné riešenia), ku ktorým autor dospel, sa musia logicky usporiadať a pri opisovaní sa musia dostatočne zhodnotiť. Zároveň sa komentujú všetky skutočnosti a poznatky v konfrontácii s výsledkami iných autorov. Výsledky práce a diskusia môžu tvoriť aj jednu samostatnú časť a spoločne tvoria spravidla 30 až 40 % školského diela.

Bibliography

[1] Owasp top 10 vulnerabilities. https://snyk.io/learn/owasp-top-10-vulnerabilities/.

[2] Reflected cross site scripting (xss) attacks. https://www.imperva.com/learn/application-security/reflected-xss-attacks/#:~:text=Reflected%20XSS%20attacks%2C%20also%20known,enables%20execution%20of%20malicious%20scripts.

[3] Top 10 most impactful and rewarded vulnerability types. https://snyk.io/learn/owasp-top-10-vulnerabilities/.

[4] Types of cross site scriping. https://owasp.org/www-community/Types_of_Cross-Site_Scripting.

[5] Xss filter evasion cheat sheet. https://owasp.org/www-community/xss-filter-evasion-cheatsheet.

[6] KirstenS. Introduction to cross site scripting. https://owasp.org/www-community/attacks/xss/.

[7] J. R. R. Tolkien. *The Lord of The Rings, The Fellowship of the Ring*. 1954.