VELEUČILIŠTE U BJELOVARU

SEMINARSKI RAD

Cross Site Scripting (XSS attacks)

Domagoj Maček

Bjelovar, Svibanj 2022.

U ovom seminarskom radu obrađena je jedna vrsta hakiranja ili ranjivosti računalne sigurnosti koja se zove cross site scripting (XSS). Navedena tema je obrađena u 5 poglavlja, a to su:

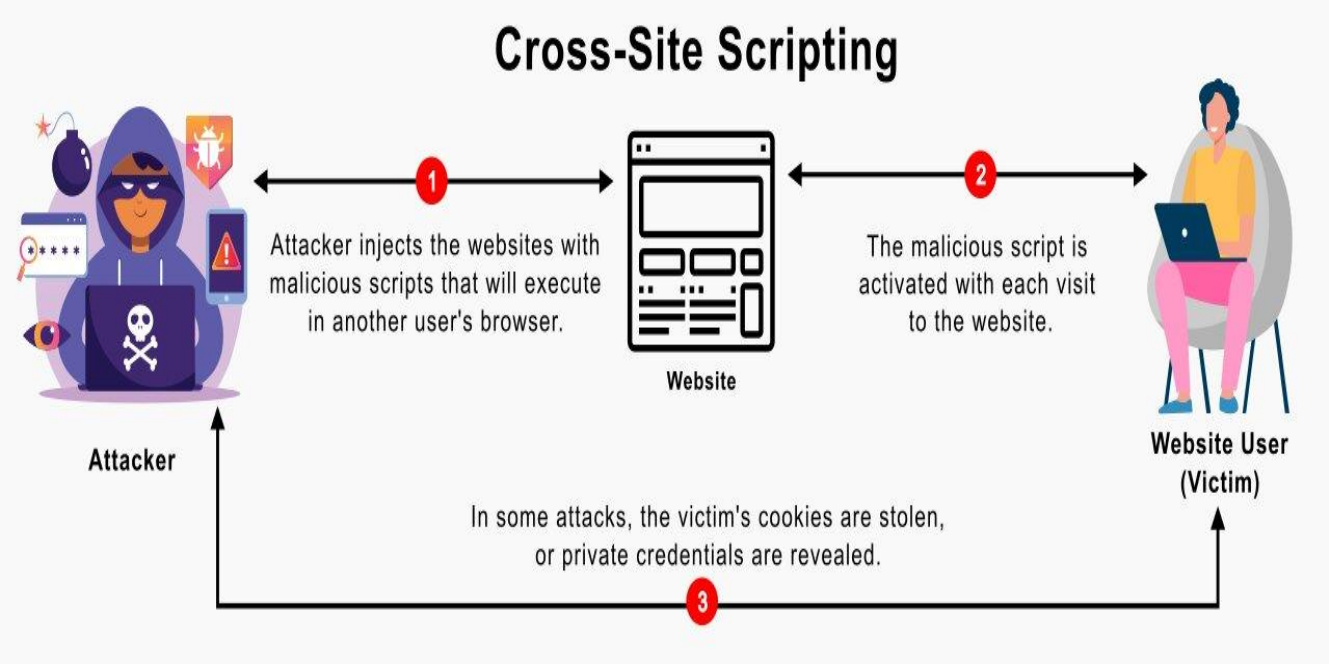
- Definicija XSS-a
- Klasifikacija i način izvođenja napada
- Primjeri nekih od XSS napada
- Zaključak
- Literatura

Sadržaj

1. Definicija XSS-a

Cross site scripting ili XSS najčešća je vrsta napada na web aplikacije. To je vrsta napada koji ubrizgava, umeće zlonamjeran kod u pouzdani kod (HTML, CSS, JavaScript) na inače sigurne web stranice i web aplikacije ili one za koje se misli da su sigurne. Napadač koristi nekakav propust u ciljanoj web aplikaciji da pošalje nekakvu vrstu zlonamjernog JavaScript koda.

Napadačev cilj nije zapravo stranica, ili web aplikacija, već sam korisnik koji koristi određene web stranice ili web aplikacije. Naječešće se koristi za krađu korisničkih kolačića ili za usmjeravanje povjerljivih informacija na neku drugu, preusmjerenu web aplikaciju, napadač je čak i u mogućnosti da može manipulirati sa prozorom web preglednika u kojem je otvorena web aplikacija. Po ovome bi netko mogao zaključiti da ova vrsta napada i nije toliko opasna te da napadač ne može napraviti neku ozbiljniju štetu, ali treba imati na umu da je JavaScript moćan programski jezik koji podržava rad sa kolačićima, slikama, DOM (Document Object Manipulation) manipulaciju ...



Slika 1. *Cross Site Scripting*

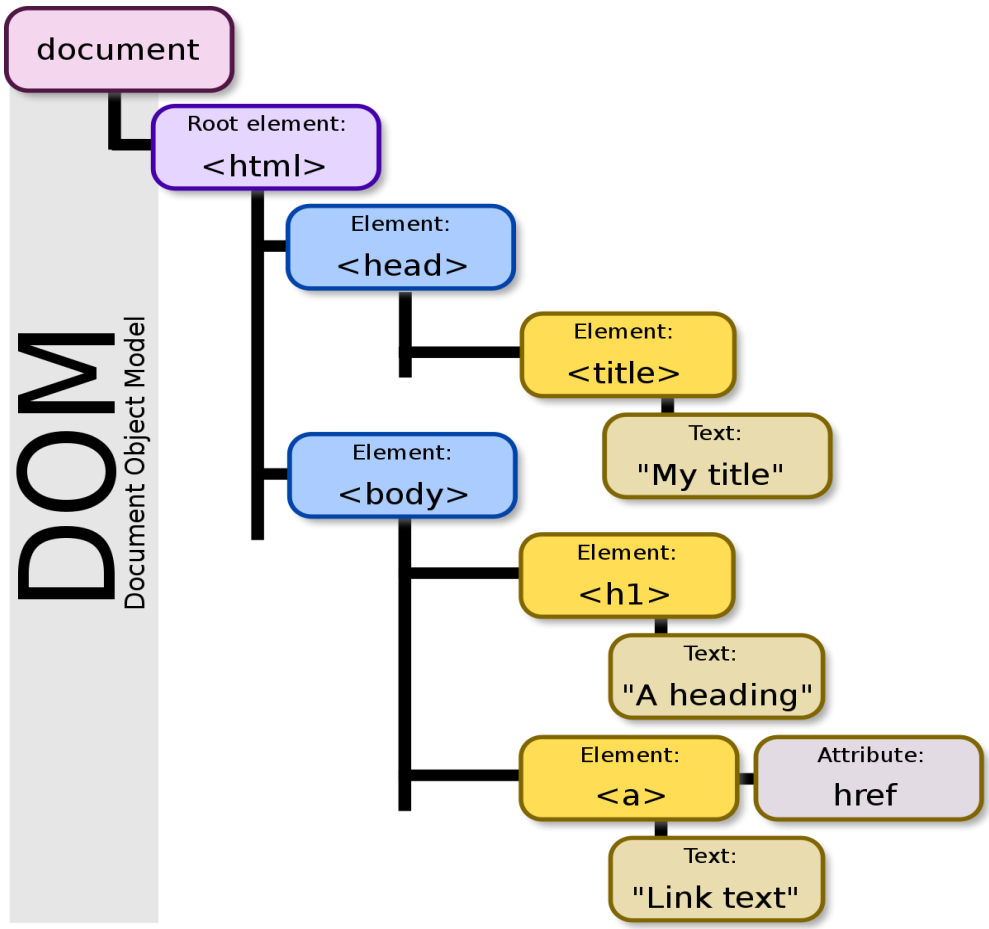
2. Klasifikacija i načini izvođenja napada

Postoje 3 vrste XSS-a koje se koriste za napade i koji imaju sličan cilj na metu koja se napada, tj. mogu se klasificirati kao:

- DOM-Based (engl. *Document Object Model)* (Type 0)
- Reflektirani (engl. *Reflected)* (Type 1)
- Pohranjeni (engl. *Stored)* (Type 2)

2.1 DOM-Based XSS

Kod ove vrste napada važno je reći što je to DOM (engl. *Document Object Model)*. To je API za HTML i XML dokumente koji definira logičku strukturu kreiranog dokumenta u obliku logičkog stabla te način na koji se tom dokumentu pristupa i manipulira. Svaka grana tog logičkog stabla završava u određenom čvoru koji u sebi sadržava objekt. Ukratko DOM je samo model koji nam pokazuje izgled našeg HTML koda od vrha do dna logičkog stabla.



Slika 2. *DOM (Document Object Model)*

Dakle ovaj napad služi kako bi se dohvatio ciljani DOM element kreiranog dokumenta te bi se potom izmjenio prema napadačevom kodu. Napadač pošalje poveznicu neke web

stranice sa implementiranim malicioznim kodom. Preglednik žrtve prima poveznicu te šalje HTTP zahtjev na poveznicu i potom prima statičku HTML stranicu. Preglednik zatim počinje graditi DOM stranice te tako dohvaća i malicioznu skriptu koja se onda pokreće.

2.2 Reflektirani XSS (engl. *Reflected)*

Ovo je vrsta XSS napada u kojoj se napadačeva maliciozna skripta reflektira kroz HTTP odgovor web preglednika. Npr. napadač šalje poveznicu neke stranice sa umetnutim malicioznim kodom žrtvi za koju žrtva misli da je ispravna, tj. stranica nije ˝sumnjiva˝. Potom žrtva otvara poveznicu i pronalazi npr. nekakve HTML okvire za pretragu podataka na stranici. Pošto je umetnuta maliciozna skripta u poveznicu stranice i takva stranica je podložna ovoj vrsti napada, svaka žrtvina pretraga će se najnormalnije izvršiti ali također će pokrenuti i malicioznu skriptu (najćešće JavaScript kod). Ovom metodom napadač najčešće krade kolačiće žrtve i jedna je od najčešćih vrsta napada na internetu.

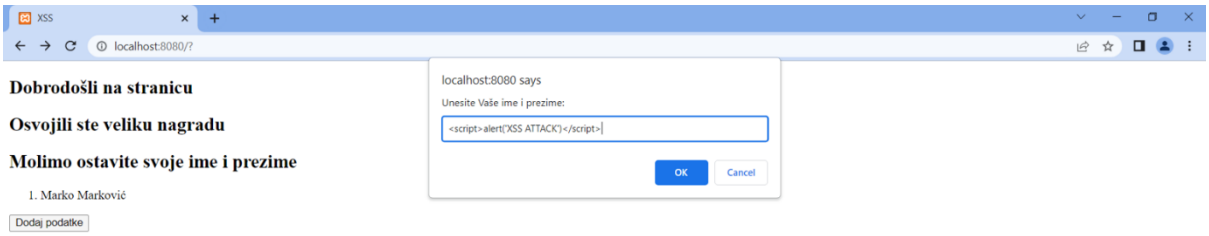
2.3 Pohranjeni XSS (engl. *Stored)*

Ova vrsta XSS napada je najštetnija za žrtvu. Napadačeva zlonamjerna skripta sa malicioznim kodom ostaje trajno pohranjena na nekoj web aplikaciji, npr. unutar baze podataka. Događa se slična stvar kao i kod reflektiranog XSS napada. Napadač locira ranjivost aplikacije ili već unaprijed zna za nju ukoliko ju je sam izradio. Najčešće su to blogovi, društvene mreže, platforme za dijeljenje videa, oglasne ploče itd. Svaki put kada se neka od tih web aplikacija i web stranica ili samo određen dio gdje se nalazi zlonamjerna skripta na njima pogleda, zlonamjerna skripta sa malicioznim kodom se prenosi u žrtvin preglednik koji ju vidi kao legitimni dio web aplikacije ili web stranice te se potom izvršava.

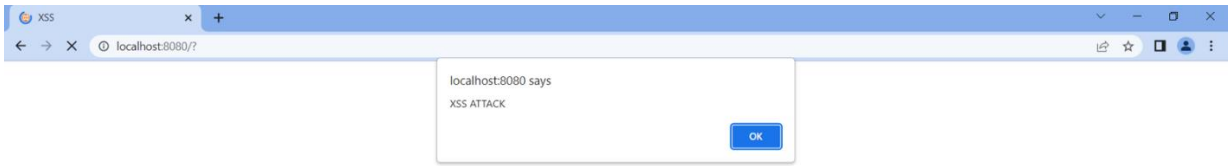
3. Primjeri nekih od XSS napada

Primjer1: Pohranjeni XSS napad

U ovom jednostavnom primjeru prikazan je pohranjeni XSS napad. Kreirana web stranica je ranjiva na ovu vrstu napada te se zbog toga može vrlo lako i izvesti.



Slika 3. *Pohranjeni XSS*

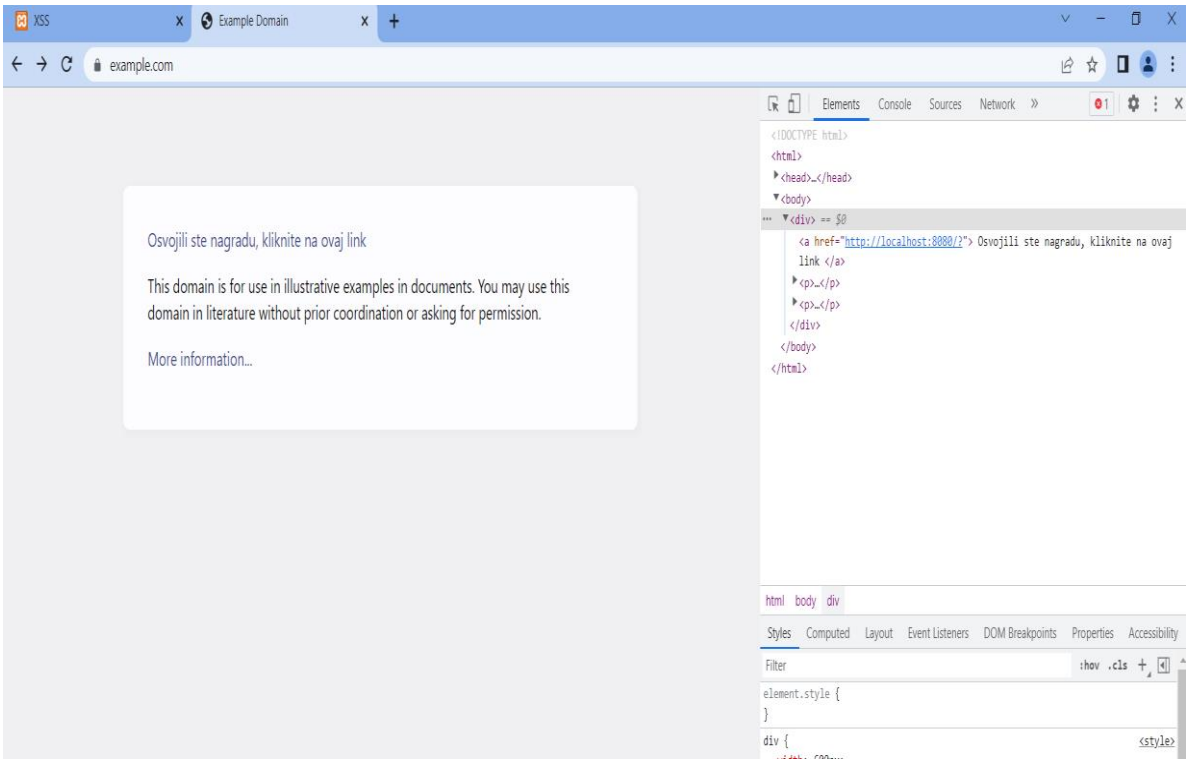


Slika 4. *Pohranjeni XSS*

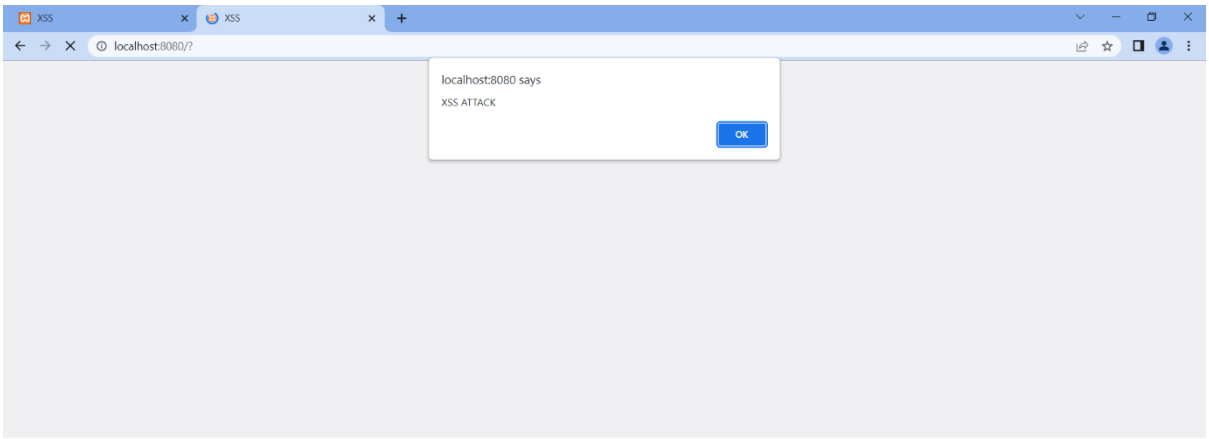
Na slici 3 vidimo upis maliciozne skripte koja se na slici 4 izvršila. Svaki put prilikom osvježavanja stranice ponovno će se pokretati maliciozna skripta jer je trajno pohranjena na lokalnom NodeJS serveru.

Primjer1: Reflektirani XSS napad

U ovom jednostavnom primjeru prikazan je reflektirani XSS napad. Kreirana web stranica je sada dostupna na poveznicu koju napadač šalje na drugu web stranicu koja je također podložna XSS napadima i preusmjerava žrtvu na željenu stranicu ukoliko žrtva pritisne klikom miša na poveznicu koja u sebi sadrži zlonamjernu skriptu.



Slika 5. *Reflektirani XSS*



Slika 6. *Reflektirani XSS*

Na slici 5 se može vidjeti poveznica na stranicu na koju napadač želi da se žrtva preusmjeri. Poveznica se također može poslati i putem neke poruke. Potom na slici 6 se vidi izvršavanje zlonamjerne skripte prilikom preusmjeravanja na stranicu.

U nastavku slijedi kod koji je omogućio rad stranice pomoću NodeJS servera:

Site.js

```
const express = require("express");
const fs = require("fs")
const app = express(); //kreira se objekt app

const username = ["Marko Marković"] //ime koje se već nalazi na listi
app.use(express.json());
//<!-- SEARCH -->

app.get("/search", (req, res) => { //pruža na m mogućnost da pretražimo stranicu po zadanom
upitu
    let ind = fs.readFileSync(__dirname + "/site.html")

    const s = "Could not find user " + req.query.q;
    ind = ind.toString().replace("<!-- SEARCH -->", s);
    res.send(ind);
})

app.get ("/js", (req, res )=> { //pokrećemo skriptu source.js
    res.sendFile(__dirname + "/source.js")
});

app.get("/", (req, res) => {  //u ovom dijelu se stvara lista korisnika na stranici
    let ind = fs.readFileSync(__dirname + "/site.html") //čita stranicu i vraća njen sadržaj

    const s = username.reduce((a, c) => { //vraća vrijednost na temelju izračuna prethodnog
elementa
        return `${a}<li>${c}</li>`
    }, "")
    ind = ind.toString().replace("<!-- LIST -->", s);
    res.send(ind);
})

app.get("/username", (req, res) => {

   res.send(username)
})


app.post("/username", (req, res) => {

    username.push(req.body.name);
    res.send({"success":true}) //poruka da li je naš željeni unos uspio
})

app.listen(8080); //sluša konekciju na portu 8080
```

```
console.log("Listen to 8080")
```

Source.js

```
const btnAdd = document.getElementById("btnAdd") //dohvaća button iz HTML-a
btnAdd.addEventListener("click", e => postUsername(prompt("Unesite Vaše ime i prezime: ")))
//poruka prilikom pritiska na gumb

async function postUsername(name) { //funkcija za ispis imena
    const olData = document.getElementById("olData")
    const res = await fetch("http://localhost:8080/username", {"method": "post", "headers":
{"content-type": "application/json"}, "body": JSON.stringify({"name": name   }) });
    const a = await res.json();
    alert(JSON.stringify(a))
}
```

4. Zaključak

Cross Site Scripting (XSS) napadi su prema literaturama najčešći napadi na internetu. Naizgled ne izgledaju opasno i žrtve tog napada mogu misliti da nisu ozbiljno ili nisu uopće oštećene, ali to nije tako. Ozbiljniji XSS napadi sa ozbiljnijim zlonamjernim skriptama sa malicioznim JavaScript kodovima mogu napadaču omogućiti krađu kolačića koji u sebi sadrže ID sesije te tako doći do korisničkih podataka, moguće je pustiti virus prema žrtvi te preoteti kontrolu nad radom računala žrtve, moguće je preusmjeravati žrtvu na željenu web stranicu ili web aplikaciju ili čak blokirati pristup sa slanjem velikog broja skočnih prozora, moguće je također i uvjeravati žrtvu da ne radi ništa opasno te od nje tražiti informacije za nekakvu prijavu gdje bi žrtva napadaču dala svoje osobne podatke, a napadač bi ih potencijalno iskoristio na nekakav negativan način. Sa svim navedenim se gotovo svaki korisnik interneta i sam suočio. Postoje razni filteri ili zaglavlja u JavaScript kodu Najbolji način za obranu bi bio oprez korisnika i neposjećivanje stranica koje su sumnjive te također ne treba vjerovati svim ljudima na internetu koji od nas traže osobne podatke jer ne žele nam svi uvijek dobro iako današnje novije verzije preglednika u sebi imaju uključenu stavku blokiranja JavaScript koda na svim stranicama.

5. Literatura

[1] https://medium.com/@ryoberfelder/describing-xss-the-story-hidden-in-time-80c3600ffe81

[2] D. Sever https://repozitorij.fpz.unizg.hr/islandora/object/fpz%3A2280/datastream/PDF/view, 2021

[3] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

[4] https://en.wikipedia.org/wiki/Cross-site_scripting

[5] https://www.veracode.com/security/xss

[6] https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/