POBOLJŠANJA I PROBLEMI SIGURNOSTI KOD WEB APLIKACIJA UPOTREBOM JAVASCRIPT TEHNOLOGIJE

Ph.D Alem Kozar, email: alem.kozar@iu-travnik.com
Internacionalni univerzitet Travnik u Travniku, Bosna i Hercegovina
Prof. dr Mladen Radivojević, email: radivojevicmladen60@gmail.com
ITEP, Banja Luka

Sažetak: Ovaj naučni rad se temelji na poboljšanjima i problemima sigurnosti kod JavaScript tehnologije odnosno fokus je usmjeren prema skupu postojećih tehnologija i standarda kod kojeg se stavlja naglasak na dinamičnost i asinhronost prijenosa podataka između klijenta i servera koji se naziva Ajax tehnologija. Ova tehnologija je objektno zasnovana tj. koristi objekte bez klasa i naslijeđivanja. Veoma brzo se otkrilo da postoji tehnologija koja predstavlja kombinaciju standardnih tehnologija, koje su već poznate programerima i dizajnerima. Iako Ajax sam za sebe nema vlastitih sigurnosnih slabosti, njegovim integrisanjem u dinamičke sisteme može se u značajnoj mjeri uticati na cjelokupnu sigurnost sistema. Koliko god da je pristup aplikaciji preko web browsera praktičan i krajnjem korisniku olakšava svakodnevni život, u istoj mjeri otežava život timu koji radi na razvoju web aplikacije. Naime, na tržištu postoje tri dominantna web browsera – FireFox, Google Chrome I Microsoft Edge (bivši Internet Explorer) kao i mnoštvo drugih browsera.
Ključne riječi: WEB APLIKACIJA, JAVASCRIPT, AJAX, BROWSER

IMPROVEMENTS AND PROBLEMS OF THE SECURITY IN THE WEB APPLICATIONS BY USE OF THE JAVASCRIPT TECHNOLOGY

Abstract: This scientific work is based on the security improvements and the security issues with JavaScript technology; namely, the focus is on a set of existing technologies and standards that emphasize the dynamics and asynchronism of data transfer between a client and the server called Ajax technology. This technology is objectively designed, ie. uses the objects without class and inheritance. The question arises whether there is some technology that allows the user to perform other activities such as image overview on the site, chatting with friends, reviewing a certain number of notifications received while waiting for the server response. It has quickly discovered that there is a technology that is a combination of the standard technologies, which are already known to the developers and the designers. Although Ajax does not have its own security weaknesses for itself, its integration into the dynamic systems can significantly affect the overall security of the system. As far as accessing the application through a web browser is practical and the end-user makes the everyday life easier, it also makes life difficult for the team working on the development of a web application. Namely, there are three dominant web browsers - FireFox, Google Chrome and Microsoft Edge (formerly Internet Explorer) - as well as many other browsers.
Key words: WEB APPLICATIONS, JAVASCRIPT, AJAX, BROWSER

1. Uvod

U ovom radu je objašnjen skriptni jezik sa svojim poboljšanjima i manama u razvoju interaktivnih web aplikacija. Razvoj web tehnologija ide u pravcu povečavanja učinkovitosti,

osjetljivosti I interakcije web aplikacija. Takav napredak, međutim, povećava i razinu prijetnji sa kojima se kompanije i developeri svakodnevno susreću. Opasnost se može smanjiti neprekidnim provjeravanjem ranjivosti web stranica pouzdanim web skenerima. Kako raste kompleksnost tehnologija slabosti web stranica i njihove ranjivosti postaju sve vidljivije. JavaScript je naziv implementacije ECMAScript norme koju su ostvarili Netscape Communication Corporation, zatim kasnije Mozilla Foundation. Ovaj programski jezik se oslanja na bazi prototipa i najpoznatiji je po korištenju u izradi web stranica (kao jezik na klijentskoj strani), ali ga je moguće koristiti i na strani servera s tim da je to rijedak slučaj, također je objektno zasnovan tj. koristi objekte, bez klasa i naslijeđivanja, ima više namjena, a do sada se posmatrao kao dopuna HTML-a i CSS-a, a HTML5 ga pretvara u osnovni alat programeru za web i mobilne uređaje. U sve čitače weba podrazumjevano su ugrađeni svi interfejsi za programiranje aplikacija (eng. Application programming interfaces, API-s) koji dopunjavaju JavaScript osnovnim mogućnostima. Noviji APi-ji (kao što su Web storage, Canvas i drugi) predstavljaju interfejse za biblioteke ugrađene u čitače weba. Osnovna zamisao je da te moćne osobine budu svuda dostupne preko jednostavnih standardnih metoda za programiranje, proširujući opseg djelovanja jezika i olakšavajući izradu atraktivnog i korisnog softvera za web. Dakle JavaScript se može upotrebljavati za razvoj web stranica front-end i back-end. Front end uključuje manipulisanje DOM-om (Objektni model dokumenta) na webu preglednika (na primjer, animacije, umetanja podataka, asinhrona ažuriranja), a back-end uključuje logiku servera (npr. usmjeravanje, rukovanje podacima, interakcije sa bazama podataka). Neki JavaScript frameworkovi rade oboje, što se zove "full-stack" ili cjelokupni razvoj weba. Danas web aplikacije razvijane kroz skriptnu tehnologiju predstavljaju značajan vektor hakerskog napada na neki website ili server. Najbolnija tačka u funkcionisanju web aplikacije je komunikacija između klijenta i servera. Napadi mogu biti fokusirani na druge korisnike i krađu sesije, pokušaj pregleda nekog file-a sa servera ili napada na bazu podataka. Svaka složenija web aplikacija se sastoji iz dva djela a to su: klijentski i serverski dio pa iz ovoga proizilazi i njihova najveća mana.

2. Web aplikacije

Web aplikacija je softverski program koji se izvršava na web (http) serveru. Za razliku od tradicionalnih, desktop aplikacija koje se pokreću operativnim sistemom, web aplikacije se najčešće pokreću uz pomoć "web browser" programa (Chrome, Firefox, Opera, itd...).

2. 1. Prednosti

Osnovna prednost web aplikacija je njihova nezavisnost od vrste računara, operativnog sistema ili platforme. Zahvaljujući prilagodljivom dizajnu (Responsive design), istu web aplikaciju je moguće koristiti na desktop računaru, tablet računaru ili pametnom telefonu, bez obzira na operativni sistem koji je instaliran na tim uređajima (Windows, Linux, Mac itd). Mogu lako da razmjenjuju podatke sa drugim aplikacijama i, po potrebi, da se integrišu sa različitim web servisima.
Web aplikacija se instalira, ažurira i održava na web serveru, bez potrebe da se distribuiše i/ili

instalira na računarima krajnjih korisnika. To je velika prednost u kompanijama i sistemima sa velikim brojem korisnika.
Web aplikacije se mogu iznajmljivati (SaaS - Software as a Service), po cijeni koja je neuporedivo niža u odnosu na cijenu desktop aplikacije. U ovom slučaju krajnji korisnik nema pristup izvornom kodu aplikacije, što otežava softversko piratstvo ili pokušaje obrnutog inžinjeringa (reverse engineering).
HTML5 omogućava jednostavno kreiranje interaktivnog okruženja u okviru web browsera (video, audio, vizuelni efekti), uz pristup hardverskim elementima korisnikovog računara poput web kamere ili mikrofona.
Po kvalitetu, web aplikacije su postale ravnopravne desktop aplikacijama u tolikoj mjeri da je aktuelni trend da se korisnički interfejs desktop aplikacija dizajnira u stilu web aplikacija.

2. 2. Nedostaci

Ahilova peta web aplikacija je Internet veza. Ukoliko dođe do prekida Internet veze, web aplikacija postaje djelomično ili u potpunosti neupotrebljiva.
U potpunosti su zavisne od web servera i kompanije koja je vlasnik aplikacije (ASP - Application Service Provider), a u slučaju bankrota kompanije ili prestanka funkcionisanja određene usluge, krajnji korisnik skoro da nema nikakva prava. Čak i u slučaju kada se objavi nova verzija aplikacije, korisnicima se rjetko ostavlja mogućnost izbora između nove i stare verzije. Koliko god da je pristup aplikaciji preko web browsera praktičan i krajnjem korisniku olakšava svakodnevni život, u istoj mjeri otežava život timu koji radi na razvoju web aplikacije. Naime, na tržištu postoje tri dominantna web browsera - FireFox, Google Chrome i Microsoft Edge (bivši Internet Explorer) kao i mnoštvo drugih browsera. Pri tome, browseri ne prate u potpunosti zadate standarde, a pojedine elemente različito tretiraju. Prilikom razvoja web aplikacije, mora se obratiti pažnja na ponašanje aplikacije u različitim web browserima, tzv. cross-browser kompatibilnost.

3. Poboljšanja kod JavaScripta

JavaScript je klijentski programski jezik I nije mu dopušteno pisanje ili čitanje file-ova I ova osobina je ugrađena radi sigurnosnih razloga. Ako se budući programer misli baviti sa backend I front-end developmentom onda je ovaj jezik nezaobilazan sa kojim će se suočiti. JS je jako fleksibilan jezik, ali nije static typed što znači da se dosta puta neka greška u programskom kodu zna pojaviti koja se iz prve ne može uočiti, pa kada aplikacija poraste i bude mnogo veća zbog fleksibilnosti JS postaje teži za održavanje zato se preporučuje mnogo unit testiranja i rad sa iskusnijim JS developerima.

Ova tehnologija može da izvršava brojne zadatke na klijentskoj strani aplikacije. Na primjer, može da dodaje potrebnu interaktivnost na web lokaciji kreiranjem padajućih menija, transformisanjem texta na strani, dodavanjem dinamičkih elemanata na stranu i pružanjem pomoći pri unosu forme.

3. 1. Načini uključivanja JavaScript koda u HTML

Predstavljeno je nekoliko načina za uključivanje JavaScripta unutar HTML koda:

- u sekciji <body>...</body>unutar oznaka<script language="JavaScript" type="texr/javascript">...</script>
- u sekciji <head>...</head>unutar oznaka<script>...</script>
- u sekciji <body>...</body> ili <form>...</form>unutar HTML elementa.

Tada se ne moraju koristiti oznake *<script>...</script>*

Pored ova tri načina, najčešće se JavaScript kod smješta u posebnim datotekama sa ekstenzijom .js, npr. funkcije.js, koja se ekasnije uključuje kao poseban file u tekućem HTML dokumentu, kao što je prikazano ispod.

```
<script language="JavaScript" type ="text/JavaScript" src = "funkcije.js">
document.write ("Uključivanje JavaScripta kao externog file-a!");
</script>
```

4. Ajax tehnologija i sigurnosni problemi

Termin AJAX nastao je 2005 godine. Predložio ga je Jesse James Garret, osnivač Adaptive Path-a, preduzeća koje se bavi razvojem informacione arhitekture. Prije pojave Ajaxa najmanja promjena na stranici zahtjevala je učitavanje cijele stranice. Ovo je korisnike prilično iritiralo, jer su morali čekati da se cijela stranica pošalje serveru na obradu i tek po završetku obrade mogli su vidjeti rezultat obrade na svojim preglednicima. Postavljalo se pitanje da li postoji neka tehnologija koja omogućuje korisniku da obavlja ostale aktivnosti kao na primjer pregled slike na sajtu, komuniciranje preko chat-a sa prijateljima, pregled određenog broja pristiglih notifikacija dok čeka na odgovor servera. Veoma brzo se otkrilo da postoji tehnologija koja predstavlja kombinaciju standardnih tehnologija, koje su već poznate programerima i dizajnerima i ona se sastoji od:

- asinhronog JavaScript-a i XML-a (eng. Asynchronous JavaScipt and XML);
- Napredni JavaScript I XML (eng. Advanced JavaScript and XML)

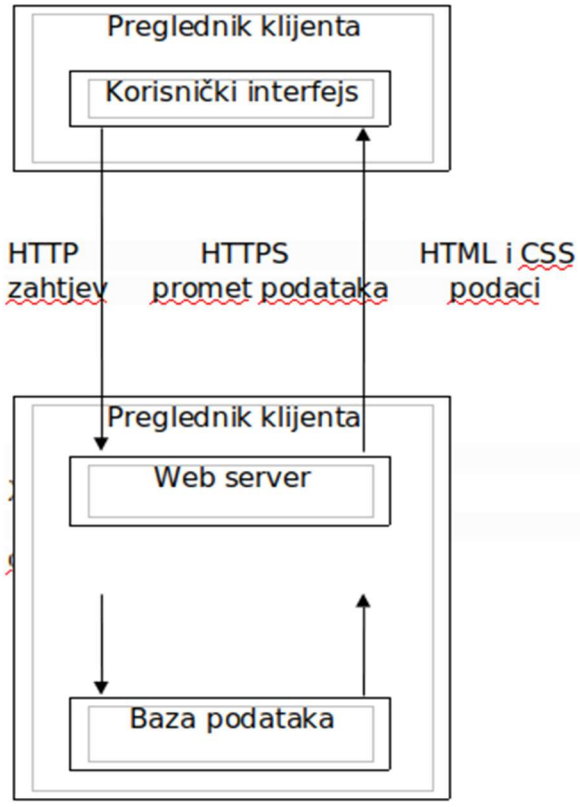
Što se tiće AJAX aplikacija, obzirom da je JavaScript njihova osnova, prve dvije tačke su nešto na šta se pažnja ne obraća, jer sa razvojem tehnologije, JavaScript se uključuje u sve poznate pretraživaće na svim platformama.

4. 1. Novi web model

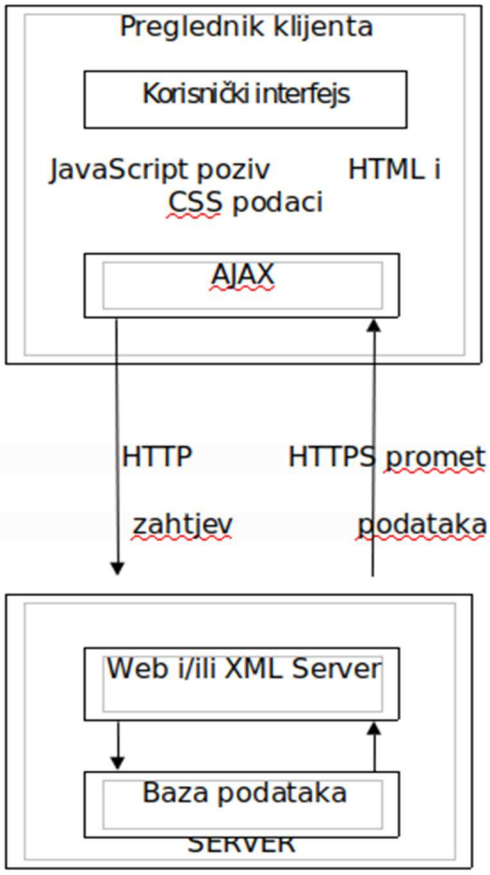
Dosta novih web aplikacija, koje koriste Ajax, vode zaključak da ova grupa aplikacija čini novi web model. Web 2.0 tehnologije obuhvataju Ajax, čineći veoma bogate Internet aplikacije.

Klasični model rada Web aplikacije i AJAX model razlikuju se kako u načinu prihvatanja podataka, tako i po mjestu obrade podataka i to je prikazano na slici ispod.

Slika 1. Klasičan model WEB aplikacije



Slika 2. AJAX Model WEB aplikacije



Ono šta je vrlo značajno u interakciji sa korisnikom je zapravo stvaranje asinhronosti. Naime, stvaranjem asinhronosti postiže se velika osjetljivost aplikacije na korisničku interakciju. Tradicionalna web aplikacija radi na način da klijentski preglednik obrađuje web stranicu, tako što pošalje zahtjev (stranicu) serveru i zatim nakon što primi od njega odgovor, stranicu prikazuje u svom prozoru. Ovo uzrokuje "kreni stani" način rada, tačnije uzrokuje pause pri slanju/primanju podataka u kojima korisnik pred sobom ima prazan prostor preglednika, Ajax nudi asinhronost u slanju/primanju podataka, dodavanjem novog sloja u aplikaciju, kao što je ilustrirano na slici 2. Umjesto da učita samo web stranicu, preglednik na samom početku učita i Ajax mehanizme napisane u JavaScript jeziku. Ovaj mehanizam je odgovoran kako za ispisivanje sadržaja korisniku, tako i za komunikaciju sa serverom pri primanju izmjena, odnosno novih podataka.

4. 2. Sigurnosni problemi

Dinamički web sistemi, koji se danas razvijaju, uglavnom se temelje na korištenju WEB 2.0 tehnologija, koje u sebi sadrže kako AJAX mehanizme tako i mnogo JavaScript kodova. Kroz

evoluciju web-a došlo je i do stvaranja novih vrsta crva i virusa, koji se njime šire. Portali kao što su Google, NetFlix, Yahoo, Facebook, MySpace i ostali, većinom koriste AJAX, te su u nekoliko navrata bili žrtve novih vrsta napada. Iako Ajax sam za sebe nema vlastitih sigurnosnih slabosti, njegovim integrisanjem u dinamičke web sisteme može se u značajnoj mjeri uticati na cjelokupnu sigurnost sistema. Poznato je da je serijalizacija podataka i objekata u prošlosti bila teško izvodljiva kroz umetanje srednjeg sloja, koji je vršio obradu. Danas AJAX može koristiti: XML, HTML, JavaScript polja, JSON i JavaScript objekte, pozivanja funkcija srednjeg sloja, što je dovelo do bitne izmjene različitih tipova podataka između klijenta i servera. Informacije koje pruža server, dinamički se stavljaju u trenutni DOM sadržaj klijenta. Ključni faktori koji utiću na ranjivost AJAX aplikacija su:

- Višestruke razbacane krajnje tačke I skriveni pozivi – Jedna od glavnih razlika između WEB 2.0 i tradicionalnih tehnologija je mehanizam pristupa informacijama. Sistemi bazirani na AJAX-u imaju više krajnjih tačaka za pristup u poređenju sa tradicionalnim sistemima bez AJAX-a. Potencijalni AJAX pozivi su razbacani kroz cijeli sistem, te ih je moguće inicirati pripadajućim događajima. Na taj način se zbog nepreglednosti (AJAX pozivi su skriveni ili slabo vidljivi) otvara mogućnost za napad sa strane;
- Zbunjujuća provjera – Jedan od važnih faktora svakog web sistema predstavlja provjeru ulaza i izlaza. U mnogo slučajeva, pretpostavlja se da je druga strana implementirala provjeru, što dovodi do zablude u kojoj niti jedna strana ne implementira ispravnu provjeru.
- Nepovjerljivi izvori informacija – Uglavnom WEB 2.0 tehnologije prikupljaju informacije iz različitih nepovjerljivih izvora kao što su feed-ovi, blogovi kao i drugi rezultati pretraživaća. Taj se sadržaj gotovo nikad ne provjerava, pa njegovo slanje web pregledniku krajnjeg korisnika dovodi do eksploatacije zaobilaženja istog izvora. Također, moguće je u web preglednik učitati JavaScript kod, koji zahtjeva od njega da izvrši pozive putem zaobilaženja isto izvora, čime se otvaraju sigurnosne rupe. Takav postupak može biti idealan za širenje virusa i crva.
- Serijalizacija podataka – Web preglednici mogu izvršavati AJAX pozive i serijalizaciju podataka, tako što pristupaju: JavaScript poljima i objektima, XML zapisima, HTML blokovima ili JSON zapisima. Ukoliko se bilo koji od tih podataka može presresti i izmjeniti, tada pregledniku može biti podvaljeno izvršavanje zloćudne skripte. Serijalizacija podataka sa nepovjerljivim informacijama može biti smrtonosna kombinacija za sigurnost krajnjeg korisnika;
- Izgradnja i izvođenje dinamičkih skripti – AJAX otvara pozadinski kanal i pristupa informacijama koje nudi server, te ih kasnije prosljeđuje u DOM. Da bi se to ostvarilo, jedan od zahtjeva je mogućnost dinamičkog izvođenja JavaScript skripti, kako bi se osvježilo stanje DOM stable ili memorije preglednika. Posljedice loše provjere sadržaja ili vršenja poziva prema nesigurnim izvorima podataka mogu varirati od krađe podataka o sjednici, pa sve do izvršavanja zloćudnog koda na klijentskoj strani.

Dakle, zbog spomenutih scenarija, web sistemi, koji koriste najnovije tehnologije, mogu postati manje sigurni, što na kraju može dovesti do eksploatacije ranjivosti kako na serverskoj, tako i na klijentskoj strani.

4.3. Izvršavanje koda za iniciranje napada

Izvršavanje napadačkog koda (eng. Cross site scripting, XSS) je najopasnija i najraširenija ranjivost, koja se pojavljuje kod dinamičkih web sistema zasnovanih na WEB 2.0 tehnologijama. Tada se XSS napad može pojaviti svaki put kada system prihvati podatke od korisnika i proslijedi ih dalje na obradu, bez predhodne izvršene provjere ili bez ispravnog kodiranja sadržaja. Na taj način se dopušta napadaču da izvrši proizvoljnu skriptu unutar web preglednika žrtve, čime se može obaviti krađa: korisničkih podataka, podataka o sjednici, izmjeniti sadržaj web stranice, umetnuti nepoželjan sadržaj, izvršiti lažno predstavljanje, ili čak preuzeti potpunu kontrolu nad preglednikom žrtve. Najčešće se radi o zloćudnom JavaScript file-u, iako je ranjivost moguće ostvariti bilo kojim skriptnim jezikom. Postoje tri opasna tipa XSS podatka, kao što su: reflektovani, pohranjeni i DOM ubacivanje.
Reflektovani tip podataka najlakše je eksploatisati, a zasniva se na tome da web stranica reflektuje korisnički unos nazad ka korisniku. Pretpostavka da postoji sljedeći isjećak HTML koda:

```
<html><head><title>XSS napad</title></head><body>
<form action=”obrada.php”method= “post”>
Broj: <input type= “text”name=  “broj”/><input type = “submit”value= “Obradi”/>
</form></body></html>
```

Prilikom klika na dugme Obradi, poziva file (skripta) obrada.php, čiji je sadržaj dat ispod:

```
<?php echo $_REQUEST [‘broj’]; ?>
```

Tada se u prozoru web preglednika ispisuje neprovjereni (sirovi) sadržaj textualnog polja, čime se otvara velika mogućnost da se izvrši napad.

Pohranjeni tip podataka se ostvaruje ubacivanjem nepoželjnog korisničkog sadržaja u file, bazu podataka ili u neki pozadinski sistem, da bi se na kraju bez filtriranja prikazao korisnicima. Ovaj tip XSS-a je vrlo opasan u sistemima, kao što su npr. socijalne mreže, blogovi, forumi I bilo koji drugi sistemi gdje veliki broj korisnika može vidjeti unos drugih korisnika.

XSS napad baziran na DOM ubacivanju nešto se drugačije realizuje sa predhodna dva napada. Kod ovog napada se vrši izmjena JavaScript koda ili promjenljivih, a ne HTML elemenata. Također, moguće je da napadački kod bude mješavina ova tri tipa.

4. 4. Standardne ranjivosti Ajax aplikacija

Iako je Ajax najmočniji skup tehnologija, developeri moraju biti svjesni potencijalnih sigurnosnih rupa koje Ajax aplikacije posjeduju. Povećana interaktivnost sa web aplikacijom dovodi do rasta količine XML texta, kojim se razmjenjuju podaci između klijenta i servera I općenito mrežnog HTTP saobraćaja. To može voditi do otkrivanja pozadinskih aplikacija koje prije nisu bile ranjive, ako je zaštita na strani servera nedovoljna, zlonamjernim korisnicima daje mogućnost upravljanja konfiguracijom korisničkih privilegija.
Još jedna slabost Ajax tehnologije je proces koji formuliše zahtjeve servera. Ajax mehanizam koristi JavaScript za preuzimanje zahtjeva korisnika i transformiše ih u pozive funkcija. Pozivi funkcija šalju se u obliku texta na server i mogu lagano otkriti polja u tabelama baze podataka poput korisničkih imena i slično. Njima se mogu unositi imena varijabli, mijenjati tipovi podataka ili njihove vrijednosti i svaki drugi parametar kojim zlonamjerni korisnici mogu upravljati.
Mnoge web stranice pripisuju svoje interaktivne osobine JavaScriptu, široka upotreba te tehnologije donosi nekoliko značajnih sigurnosnih problema.

4. 4. 1. Trovanje JavaScript polja

Ovo je vrlo čest objekat nad kojim se vrši serijalizacija, jer je jednostavno prenosiv na različite platforme i jednostavan za korištenje u raznim jezicima. JavaScript polje se može zatrovati jednostavnim XSS napadom na klijentu. Dat je primjer JS polja u kojem se, ukoliko nije ispravno provjereno, može zatrovati posljednje mjesto u polju.

```
newArray(“Laptop”, “Thinkpad”, “T60”, “Used”, “900$”, “It is great and I have used it for two years”)
```

Umetanje skripte umjesto vrijednosti posljednjeg mjesta u polju može dovesti do njenog izvođenja.

4. 4. 2. Ubacivanje JSON parova

JSON je jednostavan i efektan format razmjene podataka. Serijalizacija JSON-a je efektan mehanizam u Web 2.0 aplikacijama i često se koristi umjesto XML-a. U sljedećem textu prikazan je jendostavan JSON objekt *bookmarks* sa parom ime-vrijednost.

Primjer JSON objekta “bookmarks”

```
{"bookmarks":[{"Link":"www.example.com","Desc":"Interesting link"}]}
```

Moguće je umetnuti zločudnu skriptu u polje Link ili polje Desc. Ukoliko se ovaj objekt umetne u DOM I izvrši, preglednik će biti ponovo žrtva XSS napada.

4. 4. 3. Manipulacija XML toka

Ajax koristi XML podatke sa raznih lokacija kojr izlaze iz Web usluga baziranih na SOAP, REST, ili XML-RPC tehnologijama. Ove Web usluge često generišu podatke od treće strane. Ukoliko napadač može doći do tih podataka i izmijeniti ih, onda u njih može ubaciti i zločudan kod. Web preglednik koristi XML analizator kojim obrađuje preuzeti XML tok. Analizator može biti ranjiv na različite tipove XML ubacivanja. Prema tome potrebno je vršiti važeću provjeru XML podataka u pregledniku da bi se ponovo smanjila mogućnost XSS napada.

5. Zaključak

Razvoj web tehnologija ide u smjeru povećavanja učinkovitosti, osjetljivosti i interakcije web aplikacija. Korištenjem Ajaxa poboljšava se kvalitet interaktivnosti sa korisnikom, uz želju da što više liči na desktop aplikacije (prema brzini interakcije). Ideja na kojoj se zasniva Ajax jeste da se stranica na kojoj se odvija Web aplikacija učita samo jednom, a da se svaka dalja komunikacija sa serverom izvršava asinhrono bez blokiranja interfejsa i bez ponovnog učitavanja čitave stranice. Asinhrono ponašanje podrazumjeva da nakon interakcije korisnika sa interfejsom, zahtjev ka serveru prihvata JavaScript i XMLHttpRequest objekat, koji u pozadini šalje zahtjeve serveru a prikazujući rezultate kada budu raspoloživi, dok korisnik u međuvremenu može da nastavi sa radom. Javni web priključci 80 (HTTP) I 443 (HTTPS) uvijek su otvoreni za dostavljanje dinamičkog sadržaja i razmjenu podataka. Na taj način dovodi web stranice u neprekidnu opasnost krađe podataka. Opasnost se može smanjiti neprekidnim provjeravanjem ranjivosti web stranica pouzdanim web skenerima.
XSS crvi će postati sve inteligentniji i sposobniji za izvođenje destruktivnih napada poput distribuisanog DoS napada, napada putem elektronske pošte i nepouzdanih web preglednika. Otkriveno je da je moguće koristiti JavaScript za dobivanje pristupa lokalnih i korporativnih mreža, prema tome svaki uređaj koji je povezan na mrežu automatski ga čini ranjivim.
Sigurnost aplikacije se ne smije zanemarivati jer bez sigurnosti korisnici mogu lako, namjerno ili nenamjerno srušiti aplikaciju. Sav korisnički unos se mora provjeravati prije daljnje obrade na serveru i unosa u bazu podataka. Dakle treba provjeravati sadrži li korisnički unos znakove koje bi aplikacija mogla protumačiti kao skriptu ili kod za izvršavanje.