Automatizovano popunjavanje web formulara zaobilaženjem pravila istog izvora

Samir Ribić
Elektrotehnički fakultet
Univerzitet u Sarajevu
Sarajevo, Bosna i Hercegovina
Samir.ribic@etf.unsa.ba

Sadržaj—**Popularno je više osnovnih vrsta korisničkih interfejsa za unos podatka, ali se ni za jednu ne može reći da je najbolja u svim slučajevima. Česta je situacija da je potrebno unijeti više podataka automatski u web aplikacijama, a na raspolaganju je samo web formular, a ne i pristup koji omogućava izvršavanje skripte. JavaScript bi mogao takve formulare softverski popuniti i poslati na server, no posebno pravilo da se JavaScript izvršava na istom izvoru to sprečava ili otežava. Ipak, postoje web pregledači koji mogu isključiti ovo pravilo. U radu je prikazan jedan primjer s kodom za automatsko popunjavanje web formulara koji koristi ovo isključivanje pravila istog izvora.**

Ključne riječi- JavaScript, formulari, pravilo istog izvora

I. UVOD

Postoji više tipova interakcije korisnika i računara. Međusobno se razlikuju i svaki od njih ima prednosti i nedostataka u usporedbi s ostalim. Zbog troškova softverskog razvoja, međutim, korisnički softver, nema uvijek implementirane različite načine unosa podataka, pa se ponekad mora koristiti manje efikasan način unosa.

Komunikacija s komandnom linijom odlikuje se težom krivom učenja, većom mogućnosti greške i izrazitom fleksibilnošću. Za unos pojedinačnih podataka, ovaj pristup je sporiji od ostalih, ali za masovni unos velike količine sličnih podataka, komandna linija predstavlja najbrži način. Uz pomoć skriptnih jezika poslovi se mogu automatizovati i tako omogućiti masovni unos podataka prema određenoj situaciji.

Tekstualni unos podataka s maskama je uglavnom zastupljen u starijim aplikacijama, kao što su knjigovodstveni programi za DOS ili mainframe sisteme. Ovakav interfejs je najmanje fleksibilan po pitanju izmjena strukture podataka koji se unosi i interakcije s drugim korisnicima. Za unos pojedinačnih podataka preko tastature, ovakav korisnički interfejs je ubjedljivo najbrži.

Jedan od najčešćih korisničkih interfejsa koristi formulare za unos podataka u grafičkim okruženjima poput Windows, Linux KDE, MacOS-X i slično. Ovaj interfejs se odlikuje velikom prilagođenošću potrebama konkretne aplikacije, praktičnim i relativno brzim unosom pojedinačnih podataka, mogućnošću interakcije s drugim aplikacijama, ali zaostaje za komandnom linijom i skriptnim jezicima u situacijama kada je potreban automatski unos podataka.

Programi za pregled web stranica također mogu izvršavati web aplikacije koristeći programe u klijentskim i serverskim skriptnim jezicima, tako da web pregledač postaje izvršna platforma za različite vrste aplikacija. Ovakve aplikacije se odlikuju lakom izmjenom korisničkog interfejsa u slučaju izmjene aplikacija, nešto slabijim performansama i prilagođenošću interfejsa u odnosu na specijalizovane grafičke aplikacije.

Drugi korisnički interfejsi, kao što su prepoznavanje rukopisa optičkom tehnikom ili tablet uređajima, prepoznavanje glasa, uređaji za ulaz u kompjuterskim igrama, daljinski upravljači i slično, su odlično prilagođeni konkretnoj primjeni, mada su manje praktični za masovni unos podataka.

II. POTREBE ZA AUTOMATIZOVANIM UNOSOM WEB FORMULARA

Web korisnički interfejs je sve više dominantan način unosa podataka, jer je takve aplikacije najlakše održavati na velikom broju klijentskih računara. Ali, vrlo često ga je potrebno dopuniti mogućnostima drugih korisničkih interfejsa, kao što je izvršavanje skripti. Evo nekoliko primjera za to.

U bazu podataka je od više hiljada slogova je dodan podatak o spolu korisnika. Treba sada otvarati slog po slog klikom u web pregledaču, odabrati spol, kliknuti na dugme za potvrdu. S druge strane, već ranije su uneseni jedinstveni matični brojevi građana u bazu. Deseta cifra matičnog broja određuje koji je spol u pitanju i taj podatak bi se mogao dobiti i automatski.

Personalna služba koristi web bazirani program za elektronsku poštu, na primjer Zimbra ili Microsoft Hotmail. Podaci o rođendanima i praznicima koje osoba slavi već postoje u lokalnoj bazi podataka i čestitke koje se za svaku osobu ručno pišu mogle bi se automatski konfigurisati. Slično ovome, pozivi za učešće na konferenciji se mogu personalizovati i slati optimalnom brzinom koju dopuštaju filteri za spam, ali ne presporo.

Korisnikov radni dan počinje ukucavanjem istih ključnih riječi (npr. “vremenska prognoza”) u Google pretraživač i on želi da njegova početna stranica već ima unesene ove podatke.

Dnevno na stotine novih korisnika želi da se prijavi u aplikaciju, ali je softver napisan tako da operater mora za svakog od njih da klikne na ‘Approve’. Podaci kojima bi oni pristupali nisu povjerljivog tipa i operater bi rado zamijenio taj dosadni zadatak pritiskom na jedno dugme.

Neki od navedenih problema, tipično oni koji koriste HTTP GET metod, se mogu riješiti koristeći specifičnu web adresu, na primjer, za pretragu tipičnog web sadržaja na Google može se navesti URL poput:

http://www.google.com/search?q=vremenska+prognoza

Navedeni URL je, međutim, nemoćan za slučaj kada se pretraga obavlja POST metodom, pa bi tada trebalo pribjeći drugom pristupu, na primjer koristeći JavaScript.

III. PRAVILO ISTOG IZVORA U JAVASCRIPT

Programski jezik JavaScript ,ugrađen u web pregledače još od 1995, posjeduje veliki broj funkcija koje bi omogućile automatizaciju unosa web formulara, i realizaciju navedenih primjera. Riječ je o HTML DOM objektima [1], koji omogućavaju pristup do svakog HTML elementa, teksta, unosnih polja na formularima, slika, ekranskim tasterima za unos, čitajući ili mijenjajući njihovu vrijednost. Pored ovoga, mogu se simulirati događaji kao što je klik na dugme mišem ili taster na tastaturi. Na primjer, unosno polje na formularu koji se nalazi unutar istog HTML dokumenta gdje je JavaScript isječak se može napuniti na sljedeći način.

```
var adr=window.document. getElementById("adresa");
adr.value='Ulica Hrastova 30';
```

Za automatsko poretanje skripti, međutim, potrebno je mijenjati sadržaj unosnih polja u drugim HTML dokumentima. Tu se sada javlja problem sigurnosti. Opasnost koju pruža mogućnost HTML DOM se vidi iz sljedeća dva primjera.

Nakon što je korisnik prijavljen na Facebook u drugoj kartici web pregledača otvara privlačnu stranicu sa temom koja ga interesuje, ali na toj stranici se nalazi JavaScript program koji prepoznaje otvorenu karticu sa Facebook, i u korisnikovo ime šalje u javnost poruke takvog sadržaja koje on nikada ne bi slao.

Još opasniji primjer je da spomenuti JavaScript program prepozna stranice za unos kreditne kartice (npr. Paypal, eBay, Amazon), podatke koji se inače ne mogu čitati osluškivanjem mreže zbog kriptografisanja.

Ali, potpuno onemogućiti HTML DOM bi smanjilo upotrebljivost sajtova kao što su Facebook i Amazon, jer ovi sajtovi dinamički mijenjaju sadržaj stranica svojih korisnika.

Kao kompromis je uvedeno pravilo istog izvora (engl. “Same origin policy”)[2]. Pojam “Isti izvor” znači da se izmjene druge stranice koristeći JavaScript tehnike poput HTML DOM i AJAX mogu obaviti samo ako stranica gdje se nalazi skripta i stranica na kojoj su HTML podaci imaju isto ime domene, hosta, protokola i TCP porta.

Na primjer, neka se JavaScript program nalazi unutar sljedećeg dokumenta, na datom URL.

http://www.mediteran.com/dir/index.html

Navedeni JavaScript program će moći da mijenja i čita sadržaj sljedećih web stranica, koje imaju isto ime servera, domene i protokola, bez obzira što stranica ili eventualni direktorij nisu isti.

http://www.mediteran.com/dir/index.html
http://www.mediteran.com/dir/kupovina.html
http://www.mediteran.com/dir2/nabava.html

U narednim primjerima pristup je zabranjen, zbog različitih imena domene, servera, protokola ili porta.

http://www.med.com/dir/index.html
http://server.mediteran.com/dir/kupovina.html
https://www.mediteran.com/dir2/nabava.html
http://v2.www.mediteran.com/dir/index.html
http://www.mediteran.com:81/dir/kupovina.html

Prema tome, eventualni program u JavaScript jeziku koji omogućava automatsko popunjavanje web formulara treba da bude lociran na istom serveru kao i web stranica čiji se sadržaj mijenja. To nije uvijek moguće, pogotovo ako je riječ o automatizovanom unosu na serverima za koje nemamo prava slanja web stranica, nego samo unos podataka kroz ranije pripremljene web formulare.

Kako je pokazano u [3], pravilo istog izvora ima i dosta nedostataka. S jedne strane ono predstavlja programerima ograničenje, a s druge strane sâmo ovo pravilo nije dovoljan uslov za sigurnost JavaScript-a. Na primjer na javnim serverima za smiještanje web stranica, potpuno nevezani korisnici se nalaze na istom izvoru, npr. www.server.com/~petar i www.server.com/~alisa, a postoje i metode hakerske zloupotrebe, kao što su prevare zahtjeva između sajtova, skriptiranje između sajtova i dinamičke farme. Stoga se u [4] predlaže da se provjera autentičnosti obavlja odgovarajućim tokenima, prije upotrebe pravila istog izvora.

Vozač koji savjesno veže sigurnosni pojas radi bezbjednosti u toku vožnje, ipak ponekad želi da ga otkopča, kako bi lakše mogao voziti unazad radi parkiranja na tijesnom parkiralištu. Brzina kojom vozi u tom trenutku je toliko mala da pojas samo smeta i nimalo ne doprinosi bezbjednosti, ali čim izađe sa parkirališta, pojas odmah treba vezati. Slično ovome, korisnik koji je svjestan opasnosti koje pruža isključenje pravila istog izvora, ga nikada neće isključiti kada pregleda razne web stranice, ali će ga isključiti ako je to način da pokrene određenu aplikaciju. Dok je režim pravila istog izvora isključen treba paziti da se ne radi ništa osim te konkretne aplikacije. U konkretnom primjeru automatizacije unosa web stranica, HTML datoteka s JavaScript programom koji automatizuje unos može biti lokalna datoteka, na datotečnom sistemu gdje je web pregledač, otvorena u jednoj njegovoj kartici. U drugoj kartici treba biti otvorena HTML stranica s web formularom koji se popunjava. Radi bezbjednosti, ne otvarati ništa treće za to vrijeme.

Verzije web pregledača Google Chrome između 5 i 20 (mogućnost je ukinuta od verzije 21, ali se starije verzije još

mogu naći na Internetu) omogućavaju isključenje pravila istog izvora. Ova mogućnost je predviđena za razvojnu fazu web stranica, ali u ovom radu se pokazuje njena primjena i u fazi upotrebe. Ako je instalirana verzija Google Chrome koja ima ovu mogućnost (npr. verzija 19) može se pokrenuti tako što se pregledač pokrene iz komandne linije, prelaskom u direktorij gdje se on nalazi, na primjer za Windows Vista to je

cd C:\Users\samir\AppData\Local\Google\Chrome

Nakon ovoga se pokrene web pregledač sljedećom komandom.

chrome --disable-web-security

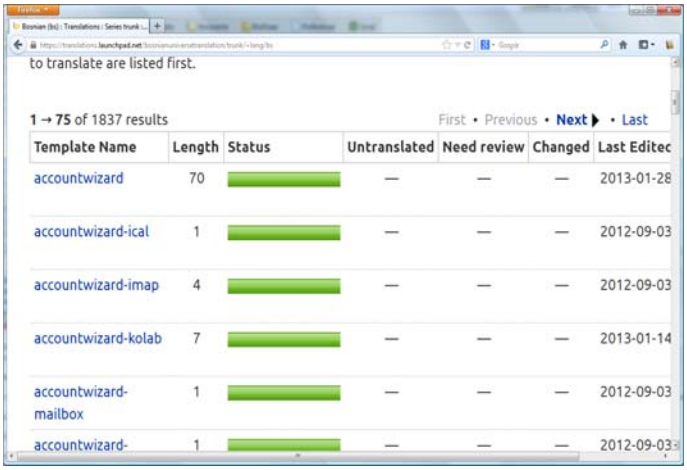
Prilikom instalacije verzije Google Chrome koja ima ovu mogućnost, treba posebno paziti na izvor odakle je ovaj web pregledač preuzet i provjeriti prečice na menijima i radnoj površini da na sebi nemaju ovu opciju permanentno uključenom.

Ova opcija, naravno neće uticati na HTML dokumente na strani servera, nego samo na njihove kopije koje su prebačene na klijentsku stranu.

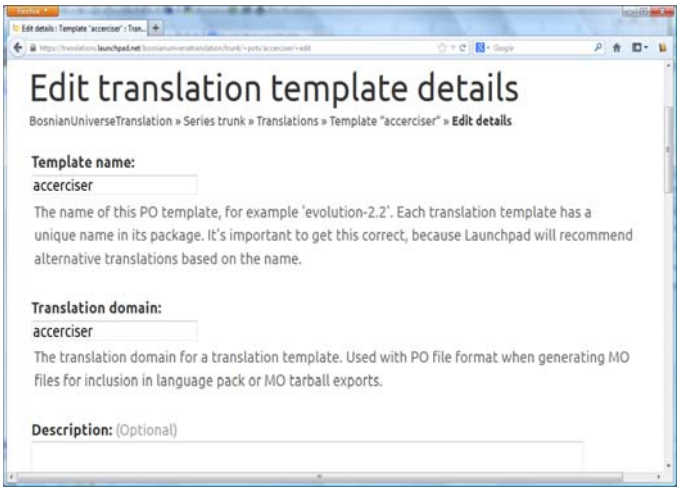
IV. PRIMJER SKRIPTE ZA AUTOMATIZOVANI UNOS

Nakon što je dopušteno JavaScript programima da utiču na lokalni sadržaj web stranica koristeći, na primjer spomenutu opciju Google Chrome, tehnika automatskog popunjavanja web formulara se može ilustrovati primjerom.

Web server http://translations.launchpad.net je namijenjen za prevođenje Linux aplikacija, sa engleskog na više od 200 lokalnih jezika. Redoslijed kojim su prikazane aplikacije koje se mogu prevoditi predstavljen je vrijednošću prioriteta (Sl. 1). Ova vrijednost se može, stoga, koristiti i za grupisanje aplikacija u srodne kategorije, na primjer da se sve KDE aplikacije prikazuju skupa. U administracijskim opcijama moguće je mijenjati parametre svakog prijevoda, a jedan od njih je prioritet (Sl. 2).



Slika 1. Nekategorisana lista URL s predlošcima za prevođenje



Slika 2. Formular za podešavanje parametara prijevoda

Sljedeći primjer koda će obaviti jednostavnu, ali napornu operaciju ako se mora ponoviti veliki broj puta. Zavisno od vrijednosti polja s imenom datoteke i direktorija kome ona pripada, dodjeljuje se odgovarajući prioritet. Na taj način će osoba koja unosi prijevode, vidjeti sve logički povezane predloške za prevođenje kao poredane jedan iza drugog.

Google Chrome ima specifičnu implementaciju JavaScript po tome što se operacije obavljaju asinhrono, što je napomenuto u njegovom programerskom vodiču [5]. Na primjer, metoda "open" objekta "window" otvara web stranicu, ali se vraća u pozivaoca bez čekanja da stranica bude otvorena. Ta osobina ovog pregledača čini ga dosta brzim, ali zahtijeva drugačiji način programiranja, koji koristi tajmere.

U početnom dijelu HTML dokumenta navedene su uobičajene HTML oznake, uz deklaraciju nekoliko globalnih varijabli i niza URL adresa koje će se redom automatski otvarati. Posljednja od njih je prazan string. Ova lista URL-ova se može dobiti i ekstrakcijom iz web stranica pomoću programa kao što su Unix alati sed i grep. Ovdje je prikazana njena skraćena verzija.

```
<!DOCTYPE html>
<html>
<head>
<script>
var myWindow,myInterval,finished,poz;
urls=[
"https://translations.launchpad.net/bosnianuniversetranslation/trunk/+pots/desktop-kdegames-kbreakout/+edit",
"https://translations.launchpad.net/bosnianuniversetranslation/trunk/+pots/desktop-kdegames-kdiamond/+edit",
"https://translations.launchpad.net/bosnianuniversetranslation/trunk/+pots/desktop-kdegames-kfourinline/+edit",
"https://translations.launchpad.net/bosnianuniversetranslation/trunk/+pots/desktop-kdegames-kgoldrunner/+edit",
"https://translations.launchpad.net/bosnianuniversetranslation/trunk/+pots/desktop-kdegames-kigo/+edit",
""
]
```

Sljedećom funkcijom se pokreće otvaranje ovih stranica u nizu. Globalna varijabla poz pokazuje na trenutnu poziciju u ovom nizu. Metodom open objekta window otvara se naredni URL. Kako se ova metoda ponaša asinhrono, postavlja se vremenski interval koji će se okidati svake sekunde i pozivati funkciju doModify, u kojoj će se pokušati popuniti formular.

```
function startScripting()
{
 poz=0;
 if (urls[poz]!='')
 {
  myWindow=window.open(urls[poz], 'scripted');
  poz++;
  myInterval=setInterval(function(){doModify()},1000);
 }
}
```

Funkcija stopScripting isključuje navedeni tajmer.

```
function stopScripting()
{
 poz=0;
 clearInterval(myInterval);
}
```

Pošto JavaScript poznaje dvije vrste praznih varijabli, null i undefined, korisno je uvesti funkciju koja će provjeriti da li je varijabla prazna.

```
function isEmpty( variable )
{
 if ( typeof( variable ) == 'undefined' ) return true;
 if ( variable  == null ) return true;
 return false;
}
```

Funkcija kojom se prosljeđuju događaji miša izgleda ovako.

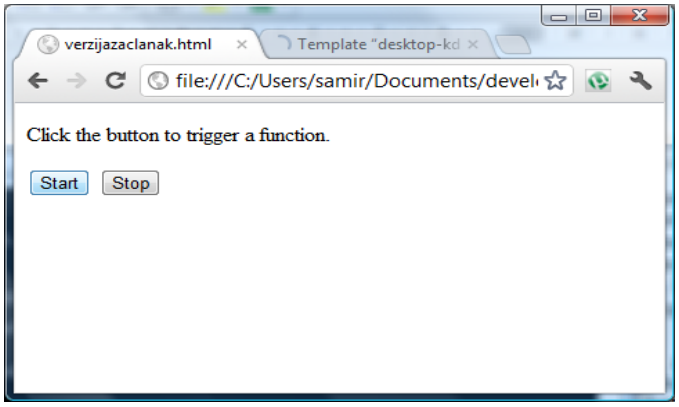
```
var dispatchMouseEvent = function(target, var_args)
{
 var e = document.createEvent("MouseEvents");
 e.initEvent.apply(e,
   Array.prototype.slice.call(arguments, 1));
 target.dispatchEvent(e);
};
```

Nakon što je stranica sa formularom učitana, treba pročitati i popuniti njena polja, te simulirati klik na dugme. Imena polja se mogu prepoznati koristeći opciju web pregledača kojom se pogleda izvorni kod HTML stranice, obično gledajući atribut ID ili atribut NAME elementa INPUT. Do pojedinog elementa se dolazi metodom getElementById objekta window. Bude li bilo koji od ovih elemenata prazan, treba izaći iz funkcije, ali će ona ponovo biti pozvana nakon jedne sekunde iz tajmera. Na ovaj način se obezbjeđuje da je stranica kompletno učitana. Zavisno od vrijednosti polja field.path, puni se polje field.priority drugim vrijednostima. Na primjer, za datoteke iz direktorija 'calligra', dodijeljena vrijednost je 740. Kada je željeno polje popunjeno potraži se element field.actons.change, i pošalje mu se događaj miša 'click'. Fokus se prebacuje na izabranu stranicu i postavlja novi interval od 5 sekundi nakon koga se pokreće funkcija afterSubmit.

```
function doModify()
{
 if (isEmpty(myWindow)) return;
 if (isEmpty(myWindow.document)) return;
 var x=myWindow.document.
  getElementById("field.priority");
 if (isEmpty(x)) return;
 var y=myWindow.document.
  getElementById("field.path");
 if (isEmpty(y)) return;
 if (y.value.substring(0,8)=='kdetempl')
 {
   x.value='750';
 }
 if (y.value.substring(0,8)=='calligra') {
  x.value='740';
 }
 if (y.value.substring(0,9)=='extragear')
 {
  x.value='730';
 }
 var b=myWindow.document.getElementById
  ("field.actions.change");
 if (isEmpty(b)) return;
 b.focus();
 dispatchMouseEvent(b,'click',true,true);
 clearInterval(myInterval);
 myWindow.focus();
 myInterval=setInterval(function(){afterSubmit()},5000);
}
```

Funkcija afterSubmit čeka da se pojavi stranica koja se aktivira nakon što se klikne na dugme za unos formulara. Pokušava se prepoznati njen karakteristični dio. U ovom slučaju, to je odsustvo riječi '+edit' u njenom URL-u. Kada je takva stranica učitana, vraća se fokus na polaznu stranicu, prelazi na naredni URL u nizu, a ako se došlo do kraja, zatvara se pomoćna kartica.

```
function afterSubmit()
{
 if (isEmpty(myWindow)) return;
 var u=myWindow.document.URL;
 if (isEmpty(u)) return;
 if (u.indexOf('+edit') != -1) return;
 window.focus();
 clearInterval(myInterval);
 if (urls[poz]!='')
 {
  myWindow=window.open(urls[poz], 'scripted');
  poz++;
  myInterval=setInterval(function(){doModify()},1000);
 }
 else
 {
  myWindow.close();
 }
}
```



Slika 3. Pokretanje skripte

Preostao je još HTML kôd za prikaz dugmadi koje pokreću funkcije startScripting i stopScripting.

```
</script>
</head>
<body>
<p>Click the button to trigger a function.</p>
<button onclick="startScripting()">Start</button>
<button onclick="stopScripting()">Stop</button>
</body>
</html>
```

Nakon što se klikne na dugme 'Start' (Sl. 3), JavaScript će redom otvarati URL-ove, popunjavati formulare i slati ih na server.

Navedeni primjer je izvršen na oko 2000 URL-ova s prioritetima u Launchpad. Procijenjeno vrijeme da se to odradi ručno je oko 33 sata. Vrijeme potrošeno za programiranje rutine je oko 5 sati, a izvršavanje (zbog postavljenih konstanti tajmera na jednu i pet sekundi) oko 4 sata. Pored direktne kvantitativne uštede vremena, postignuta je i kvalitativna, jer su 33 sata frustrirajućeg ponavljajućeg posla, koji ima potencijalne greške, zamijenjena s 5 sati kreativnog posla, a dobijeni program se, naravno, može kasnije ponovo koristiti.

V. Drugi načini zaobilaženja pravila istog izvora

Pored opisanog pristupa korištenjem opcija u pregledaču i direktne upotrebe HTML DOM, što je praktičan način za automatizaciju popunjavanja formulara, treba spomenuti još neke tehnike zaobilaženja pravila istog izvora u JavaScript, koje se koriste za čitanje podataka s udaljenih servera u JavaScript. Njihova upotreba za popunjavanje formulara, međutim, nije tako jednostavna.

Drugi način zaobilaženja pravila istog izvora se najčešće koristi za omogućavanje AJAX poziva koristeći XMLHttpRequest zahtjeva na udaljeni server. Potrebno je napisati serversku (tzv. proxy) aplikaciju koja preuzima podatke s udaljenog web servera i prikazuje ih u HTML formatu, a zatim se na istom serveru gdje je proxy aplikacija pokreće JavaScript program koji čita taj HTML. Ovakav pristup, međutim zahtijeva više programiranja, a sporiji je i manje efikasan od direktnog pristupa serveru.

JSONP je treći način zaobilaženja pravila istog izvora. On omogućava JavaScript aplikacijama pristup REST serverima bez upotrebe proxy aplikacije. Ako server ima REST web servis koji prosljeđuje podatke navodeći URL sa parametrima u formatima kao što su XML ili JSON, tada se može napisati povratna funkcija fn s jednim parametrom i u HTML napisati kôd sličan narednom koji prosljeđuje rezultate.

```
<script type=”text/javascript” src=
”http://www.c.ba/iz.php?mj=V&format=json&callback=fn”>
</script>
```

Tehnike koje predstavljaju hakerske upade ovdje neće biti opisane, jer je akcenat ovog članka na korisnoj i legalnoj upotrebi zaobilaženja pravila istog izvora.

VI. Zaključak

Pravilo istog izvora predstavlja važnu sigurnosnu mjeru u web aplikacijama. U većini uobičajenih situacija na web-u, ne treba ga isključivati jer predstavlja branu od malicioznih skripti. Sistem nije savršen, jer postoje tehnike koje ga zaobilaze, mada i protiv njih postoje načini borbe. No, i isključivanje ovog pravila može u velikoj mjeri dopuniti upotrebljivost web aplikacija ukoliko se preduzmu odgovarajuće sigurnosne mjere.

Literatura

[1] J. Keith, J. Sambels “DOM Scripting: Web Design with JavaScript and the Document Object Model “, Friendsof, 2010

[2] M. Zalewski, “The Tangled Web: A Guide to Securing Modern Web Applications”, No Starch Press, November 2011, pages 145-149

[3] H. Saiedian, D. Broyle, “Security Vulnerabilities in the Same-Origin Policy: Implications and Alternatives”. IEEE Computer 44(9): 29-36 (2011)

[4] D. Gollmann, Problems with Same Origin Policy - Know Thyself.01/2008; In proceeding of: Security Protocols XVI - 16th International Workshop, Cambridge, UK, April 16-18, 2008..

[5] Google Chrome Developer's Guide, available at site http://developer.chrome.com/extensions/devguide.html.

ABSTRACT

There are many different kinds of user interfaces for data entry. We can not say that any of them is best in all cases. It is quite common to have a situation where mass data entry is required. However, sometimes we have only web form, and not some kind of scripting or command line access which will make the data entry faster. JavaScript could fill up such forms and send them to server. However, there is a special rule that JavaScript accessing the HTML forms, and HTML forms themselves must be executed on the same server, protocol and port. Such rule makes automatic web form data entry more difficult, if not impossible. However, there are some web browsers which can disable this rule. This paper shows one example with code for automatic data entry which uses switching off of the same origin policy.

Automatic Data Entry into Web Forms by Avoidance of the Same Origin Policy

Samir Ribić