**Validacija i validacioni filteri**

Jedinica: 16 od 19

**Rezime**

* Bezbednost je na visokoj poziciji važnosti jedne web aplikacije;
* Validacija podataka može se izvršiti na klijentskoj i serverskoj strani aplikacije;
* Ako se nešto ne vidi na strani, ne znači da ga nema ni u Source HTML kodu;
* Referer je lokacija sa koje je klijent upućen na stranu putem linka;
* Za validaciju na klijentu koristi se JavaScript;
* PHP ima neke ugrađene funkcije za validaciju;
* Ugrađena PHP funkcija za validaciju naziva se *filter\_var* i prima promenljivu koju treba da proveri i tip validacije koju treba da izvrši nad njom.

U ovoj lekciji naučićete osnovne koncepte validacije i njen značaj, ali i obnoviti znanje o tome koliko je bezbednost značajna za jednu web aplikaciju.

Web aplikacije su na internetu dostupne bukvalno svakome ko ima računar i pristup internetu. Zato je bezbednost jedan od bitnih faktora pri izgradnji ozbiljnijih web aplikacija.

Bezbednost nije naročito zanimljiv deo programiranja. Svodi se na to da treba što bolje pretpostaviti šta bi sve korisnik mogao da uradi, i preuprediti to odgovarajućim sigurnosnim merama.

Verujem da do sada to već znamo, ali nije zgoreg ponoviti da podaci koji se ne vide u pretraživaču ne moraju biti uistinu nevidljivi i u izvornom HTML kodu. Na primer, skrivena (hidden) kontrola forme nije vidljiva samo ukoliko ne pristupimo izvornom HTML kodu – što znači da u nju ne smemo da stavimo ništa što korisnik ne sme da vidi.

Zatim, sledi style.display Properties HTML objekta. Ukoliko se vrednost ovog Propertiesa postavi da bude *none*, objekat neće biti vidljiv, ali će i dalje biti u izvornom HTML kodu.

Naročitu pažnju (i to na nekoliko mesta) treba obratiti prilikom rukovanja sa AJAX-om. Pre svega, iako AJAX nije vidljiv u pretraživaču, vidljiv je u HTML Sourceu, kao i informacije koje prosleđuje (osim ako su generisane u realnom vremenu, ali i tada je moguće analizom koda doći do njih). Sa druge strane, AJAX obično koristi serversku aplikaciju za punjenje podacima, pa i ta aplikacija treba da bude zaštićena od direktnog pristupa.

Druga stvar na koju treba obratiti pažnju su mesta gde korisnik ima pravo unosa podataka. Pod ovim se obično podrazumevaju forme, odnosno kontrole formi.

Posebna su (i ogromna) tema načini na koje korisnik može poremetiti aplikaciju. Na primer, zamislimo da naša kontrola prima dva parametra: korisničko ime i šifru, i da ih proveravamo putem upita ka bazi podataka. Korisniku će biti dovoljno da u jedno od ova dva polja unese i kao vrednost, na primer: ' OR 1=1 ' i da ispuni uslov ulaska u zaštićeni deo sajta.

Ili pak ukoliko korisnik ima mogućnost unosa podataka na sajt (npr. forum), ukoliko se ne izvrši validacija tih podataka, korisnik bi mogao bez problema uneti svoju skriptu, koja će, kada bude emitovana na strani, biti i izvršena.

Ova dva najčešća i najjednostavnija načina za neželjeni ulazak na sajtove (a respektivno, se nazivaju SQL injection i Cross Site Scripting), trebalo bi preduprediti određenim merama. Protiv hakovanja se, naravno, ne može, ali bar treba uraditi sve što je u moći programera da svoju aplikaciju što bolje zaštiti.

**Klijentska validacija**

O klijentskoj validaciji nećemo mnogo govoriti. Jednostavno, potrebno je preduprediti unos neželjenih podataka u kontrole na klijentskom delu. Ovo, naravno, moramo uraditi uz pomoć JavaScripta, pri čemu ćemo pre prosleđivanja podataka iz kontrole na server izvršiti validaciju. Na primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | <script type = "text/javascript">  function validationMail(field, alerttxt)  {  with (field)  {  apos = value.indexOf("@");  dotpos = value.lastIndexOf(".");  if (apos < 1 || dotpos-apos < 2)    {alert(alerttxt); return false;}  else {return true;}  }  }    function formValidation(thisform)  {  with (thisform)  {  if (validationMail(email,"It is not e-mail address") == false)    {email.focus();return false;}  }  }  </script>  </head><body>  <form action=""  onsubmit="return formValidation(this);"  method="post">  Email: <input type="text" name="email" size="30">  <input type="submit" value="Submit">  </form> |

Problem je što ovakva validacija nema baš neku suštinu, osim da korisniku pomogne prilikom unosa podataka. Jer, korisnik može celu stranu prebaciti na svoj server, promeniti joj sadržaj, izbaciti validaciju i proslediti podatke koje želi.

Zbog toga, glavna validacija treba da se odvija na serverskom delu aplikacije.

**Serverska validacija**

**Provera referera**

Korisnik na serveru ne može da ostvari takvu kontrolu kao na klijentu. Naime, korisnik podatke do servera mora poslati putem nekog sistema, a na programeru je da taj sistem osmisli tako da ukloni sve moguće pretnje.

U prethodnim lekcijama smo već pominjali „referera”. Ovo je lokacija sa koje je pretraživač došao na trenutnu lokaciju. Dobra je praksa, pre izvršavanja koda na „sigurnoj” stranici, proveriti odakle je došao i nastaviti izvršenje samo ako je u pitanju željena lokacija. Ovo možemo isprobati na najjednostavnijem primeru. Napravimo jedan PHP dokument test.php, sa sledećim sadržajem:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?php     if(isset($\_SERVER['HTTP\_REFERER'])) {        echo $\_SERVER['HTTP\_REFERER'];     }  ?>  <br>  <a href = "test.php">referer check</a> |

Ukoliko otvorimo ovu stranu prvi put, u pretraživaču će biti prikazan samo link ,,provera referera”. Ali, ako kliknemo na link, na strani će biti prikazan i referer, odnosno, strana sa koje je pretraživač došao, što je u ovom slučaju ista strana.

Ostaje samo da odlučite kojom ćete logikom proveriti dolazak (da li će biti samo provera domena ili baš određene lokacije) i uslovno izvršite ostatak strane.

**HTML encode/disarm**

Pomenuti Cross Site Scripting, sa početka lekcije, veoma je jednostavno detektovati i eliminisati u serverskom kodu. Dovoljno je da svaki korisnički unos propustimo kroz funkciju koja će konvertovati opasne karaktere u bezbedniji format (ASCII kod) i u takvom obliku ih pustimo da idu dalje.

Na primer:

|  |  |
| --- | --- |
| 1  2  3  4  5 | function sanitiseText($text)  {  $text =  str\_replace("<", "&lt;", $text);  $text =  str\_replace(">", "&gt;", $text);  } |

**Validacioni filteri**

PHP ima i ugrađene filtere za validaciju koji se mogu ostvariti funkcijom *filter\_var()*. Ova funkcija funkcioniše tako što joj se, kao parametri, proslede string i tip validacije. Zapravo, funkcija filter\_var() ima dva oblika funkcionisanja. Jedan je validacija podataka koja vrši validaciju i daje rezultat o toj validaciji, a druga je sređivanje (Sanitize) podataka, koja jednostavno sređuje podatke; nešto poput malopređašnjeg pasusa, gde smo to uradili ručno. Način na koji će raditi funkcija zavisi od prosleđenog validacionog tipa.

Ovaj primer proverava da li je string validan URL:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?php  $url = "http://www.mysite.com";  if(filter\_var($url, FILTER\_VALIDATE\_URL))   echo("OK");  else   echo("Error");  ?> |

Ovaj primer ispravlja greške (nedozvoljene karaktere) u nazivu URL-a:

|  |  |
| --- | --- |
| 1  2  3 | <?php  echo filter\_var("http://www.myšššsite.com", FILTER\_SANITIZE\_URL);  ?> |

PHP prepoznaje sledeće validacione filtere:

* + FILTER\_CALLBACK  
    Filter koji poziva korisnički definisanu funkciju za obradu sadržaja,
  + FILTER\_SANITIZE\_STRING  
    Izbacuje tagove i specijalne karaktere (HTML encode),
  + FILTER\_SANITIZE\_STRIPPED  
    Izbacuje tagove i specijalne karaktere (HTML encode),
  + FILTER\_SANITIZE\_ENCODED  
    URL enkodiranje stringa,
  + FILTER\_SANITIZE\_SPECIAL\_CHARS  
    Izbacuje specijalne karaktere,
  + FILTER\_SANITIZE\_EMAIL  
    Priprema string u e-mail formatu,
  + FILTER\_SANITIZE\_URL  
    Priprema string u URL formatu,
  + FILTER\_SANITIZE\_NUMBER\_INT  
    Izbacuje slova, ostavlja samo brojeve i aritmetičke operatore,
  + FILTER\_SANITIZE\_NUMBER\_FLOAT  
    Izbacuje slova, ostavlja samo brojeve, aritmetičke operatore, tačku i znak E za eksponent,
  + FILTER\_SANITIZE\_MAGIC\_QUOTES  
    Dodaje backslash ispred dvostrukih navodnika,
  + FILTER\_VALIDATE\_INT  
    Proverava da li je vrednost integer,
  + FILTER\_VALIDATE\_BOOLEAN  
    Proverava da li je vrednost boolean,
  + FILTER\_VALIDATE\_FLOAT  
    Proverava da li je vrednost float,
  + FILTER\_VALIDATE\_REGEXP  
    Proverava vrednost po regularnom izrazu,
  + FILTER\_VALIDATE\_URL  
    Proverava da li je vrednost zapisana kao validan URL string,
  + FILTER\_VALIDATE\_EMAIL  
    Proverava da li je vrednost zapisana kao validan e-mail,
  + FILTER\_VALIDATE\_IP  
    Proverava da li je vrednost zapisana u formatu validnog IP-ja.

Moguće je izvršiti čitav niz validacionih uslova nad nizom podataka naredbom *filter\_var\_array().*

Enkripcija na web serveru je bez smisla ukoliko se skriptovi koji koriste šeme za enkripciju  ne nalaze na serveru koji dozvoljava SSL (Secure Sockets Layer – protokol koji se koristi za enkripciju informacija preko interneta). SSL omogućava da sve podatke koje klijent unese na svojoj strani, browser kriptuje pre prosleđivanja serveru, a zatim da iste te podatke server dekriptuje kada dođu do njega.

**hash – Kriptografska ekstenzija**

Omogućava kriptovanje poruka provlačenjem kroz veliki broj ponuđenih hash algoritama. Za korišćenje ove ekstenzije nije potrebna nikakva dodatna instalacija, jer je po standardu ugrađena u PHP.

Funkcija hash() generiše hash vrednost od prosleđene poruke. Za pravilan rad funkcije, neophodno je proslediti joj dva obavezna parametra i po potrebi treći parametar koji je opcioni.

Pregled funkcije bismo mogli predstaviti ovako:

hash ( $algo , $data [,bool $raw\_output = false ] )

***$algo*–**Prvi parametar. Predstavlja izbor algoritma koji će biti korišćen. Pregled dostupnih algoritama možete dobiti upotrebom funkcije **hash\_algos()** na sledeći način:

|  |  |
| --- | --- |
| 1 | print\_r(hash\_algos()); |

Na strani će biti prikazani algoritmi. Neki od često korišćenih su: md5, sha256, haval160,4 itd.

***$data*–**Poruka koja će biti hashovana.

***$raw\_output*–**Podrazumevana vrednost ovog parametra je FALSE. Ukoliko njegovu vrednost eksplicitno podesimo na TRUE, biće vraćen sirov binarni podatak.

**Primer:**

Ukoliko bismo želeli da pomoću md5 algoritma kreiramo hash od reči **admin**, napisali bismo to ovako:

|  |  |
| --- | --- |
| 1 | echo hash("md5", "admin"); |

Rezulat na strani je:

21232f297a57a5a743894a0e4a801fc3

**Da li je moguće dekriptovati šifru kriptovanu uz pomoć md5 enkripcije?**

 Nije moguće

 Moguće je

**Na koji način možemo sprečiti unos neželjenih podataka?**

 Validacijom na serverskoj strani

 Porukom o uspešnom unosu

 Uvođenjem dodatne promenljive

 Ništa od ponuđenog nije tačno