WEB saugumas

Marius Gžegoževskis

Turinys

- Cross-Site Request Forgery (CSRF).
- SQL Injection.
- CAPTCHA testai.

CSRF(Cross Site Request Forgery)

- CSRF tam tikra prasme yra priešingybė XSS (Cross Site Scripting).
- XSS – Pasinaudoti vartotojo patiklumu jog WEB aplikacija, kurioje jis yra prisijungęs/autentifikuotas serverio, yra saugi.
- CSRF – Pasinaudoti WEB aplikacijos/Web serverio „patiklumu“, jog vartotojo aliekami veiksmai yra teisingi, kadangi vartotojas yra autentifikuotas šioje svetainėje.
- Kitaip tariant CSRF yra vadinamas XSS atakos plėtiniu.
- Idėja: Apgauti vartotoją, kuris yra prisijungęs arba kitaip autentifikuotas vartotojas WEB aplikacijoje/svetainėje, priverstinai atlikti tam tikrus veiksmus:
  - Pakeisti elektroninio pašto adresą.
  - Pakeisti pradinį puslapio adresą.
  - Pakeisti slaptažodį.
  - Išsiųsti jautrią informaciją kam nors.
  - Įsigyti kokį nors daiktą.
  - Cookie nėra pasisavinamas – vartotojui yra leidžiama naudoti savo cookie informaciją, nurodant jam kokius veiksmus jis turi atlikti.

CSRF apžvalga

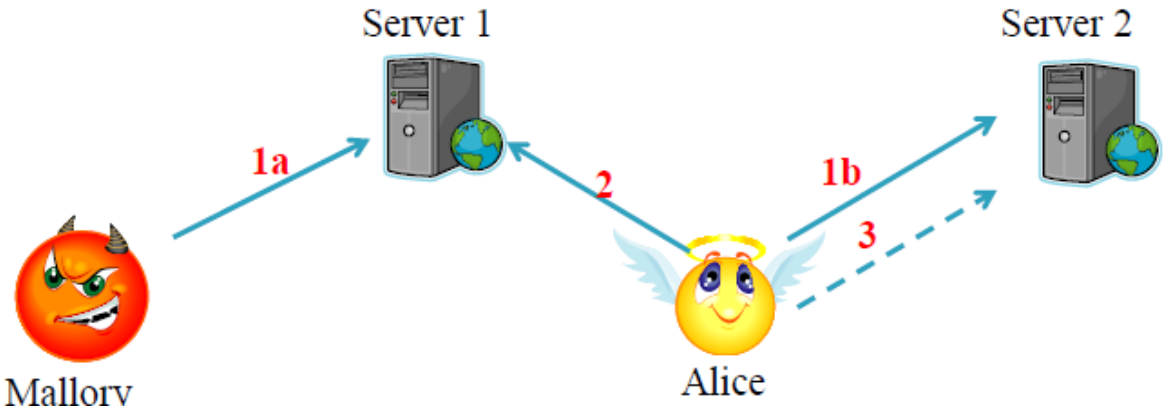
1a. Mallory įterpia kodą serveryje „Server 1“.

1b. Alice prisijungia serveryje „Server 2“, bei gauna savo „session cookie“.

2. Alice apsilanko svetainėje patalpintoje serveryje „Server 1“.

3. Kodas patalpintas serveryje „Server 1“ nurodo Alice atlikti tam tikrus veiksmus serveryje „Server 2“.

Pastaba: 1a ir 1b tvarka nesvarbi.



CSRF kodo įkėlimas „Server 1“

- Mallory atlieka nuodugnų tyrimą, kaip ir kokios užklausos yra atliekamos serveryje „Server 2“.
- Pavyzdžiui: Server 2 yra bankas.

```
POST /action.php HTTP/1.1
Host: www.server2.com
...
Cookie: PHPSESSID=gdfkeh4jkfbg...
Content-length: 42

toClearing=6352&toAcc=46718259&amount=1000
```

- Užklausa apačioje „toClearing“ persiųs 1000 piniginių vienetų į vartotojo paskyrą su tam tikru „clearing number“ ir vartotojo paskyros numeriu.
- Kintamieji action.php puslapyje yra gaunami naudojant **$_REQUEST[‘ ’]** užklausą.

CSRF kodo įkėlimas „Server 1“

```
POST /action.php HTTP/1.1
Host: www.server2.com
...
Cookie: PHPSESSID=gdfkeh4jkfbg...
Content-length: 42

toClearing=6352&toAcc=46718259&amount=1000
```

- Taigi tą patį veiksmą galime atlikti naudojant **GET** užklausą.

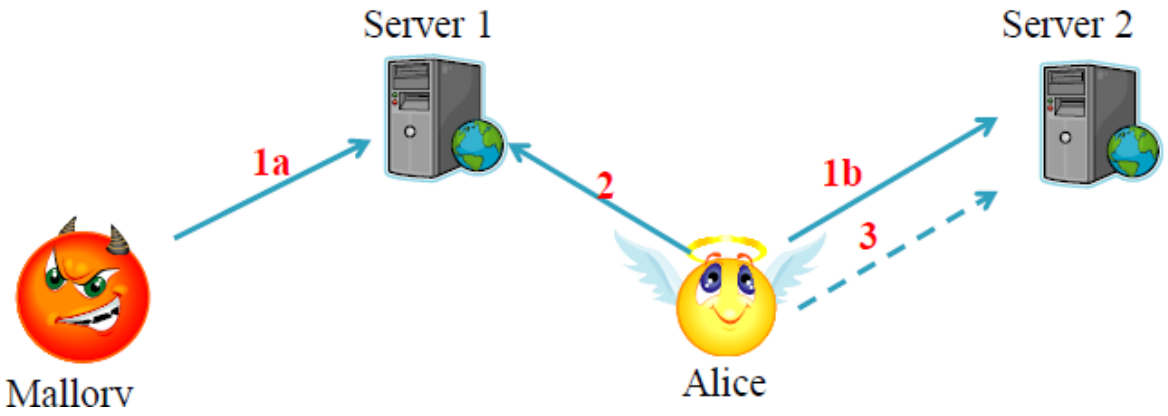
```
GET /action.php?toClearing=6352&toAcc=46718259&amount=1000 HTTP/1.1
Host: www.server2.com
...
```

Kaip įvykdyti CSRF ataką „Alice“?

- **Pirmas būdas.** Nusiųsti elektroninį laišką adresuotą „Alice“.

```
<a href=...action.php?toClearing=8362&toAcc=83712539&amount=1000>
Look at this!
</a>
```

- Bet ji žinos (matys nuorodą) ir tai nebus CSRF ataka.



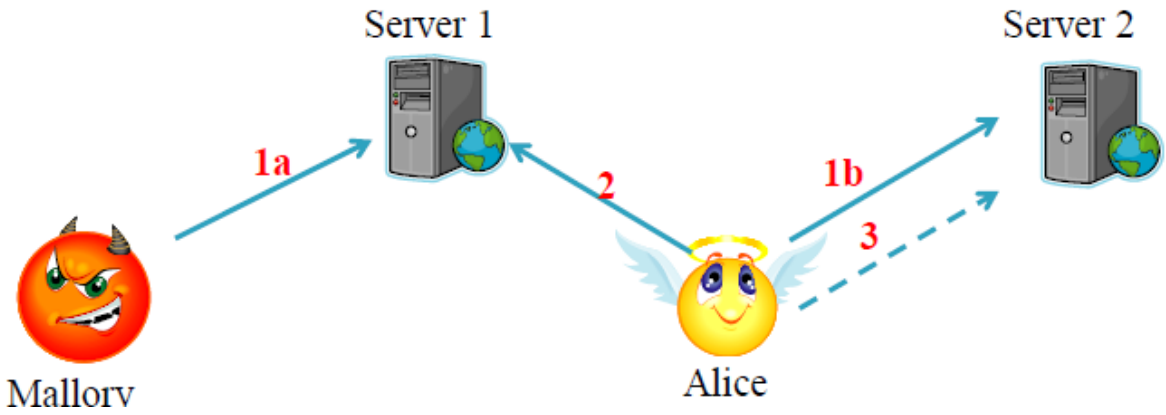
Kaip įvykdyti CSRF ataką „Alice“?

- **Antras būdas.** Įterpti paveikslėlį prieš tai pateikto pavyzdžio kode, ir jį patalpinti pavyzdžiui forume ar kitoje vietoje.

```
<img src=”...action.php?toClearing=8362&toAcc=83712539&amount=1000”>
```

- Dabar „Alice“ matys tiktai sugadinto paveikslėlio nuorodą, bet užklausa vistiek bus vykdoma.

- Reikia manyti jog „Alice“ yra prisijungus prie banko WEB aplikacijos/internetinės svetainės, tada „Mallory“ gaus pinigus. Taigi CSRF ataka sėkmingai yra įvykdoma.



Apsauga nuo CSRF atakų

- Efektyvus sprendimas yra kiekvienam veiksmui generuoti atsitiktinę simbolių seką (žetonus - token'us), kuri, vykdant nuorodą, patikrinama. Tokiu atveju saugi nuoroda atrodytų taip: http://...toclearing=6352&toAcc=46718259&amount=1000&token=Jdf1S19cQ ., kurios paskutinis elementas kaskart būtų skirtingas. Analogiškai token'ai turėtų būti įterpiami į formas HIDDEN laukų pavidalu.

```
POST /action.php HTTP/1.1
Host: www.server2.com
...
Cookie: PHPSESSID=gdfkeh4jkfbg...
Content-length: 42

toClearing=6352&toAcc=46718259&amount=1000&token=Jdf1S19cQ
```

Apsauga nuo CSRF atakų

- Vartotojui pakartotinai pateikti autentifikavimo duomenis toje pačioje HTTP užklausoje norint alikti bet kokią operaciją susijusią su jautria informacija.
- **Pavyzdžiui:** Pinigų pervedimas ir kitos panašaus lygmens operacijos, kuriose reikalinga ypač „gera“ apsauga, kad klientas nenukentėtų nuo nepageidautintų asmenų įsikišimo siekiančių pasipelnyti ar pasisavinti reikalingą informaciją.

Apsauga nuo CSRF atakų

- „Session cookies“ trukmės apribojimas, siekiant apsaugoti vartotoją nuo panašaus tipų atakų, kartais „Session cookies“ galiojimo trukmė ir nulemia vieno ar kito nepageidauntino asmens blogą veiklą, kadangi „įsibrovėliui“ nepakanka laiko išanalizuoti vartotojo reikiamų duomenų pradėti vykdyti kenkėjiškiems veiksmams.

Apsauga nuo CSRF atakų

- Pakartotinis vartotojų autentifikavimas prieš atliekant užklausas:
  - » Pakartotinis vartotojo slaptažodžio įvedimas.
  - » CAPTCHA testas.
  - » Ir kitos priemonės.

  Vartotojas prieš atsijungiant nuo WEB aplikacijos, kiekvieną kartą privalo išsiregistruoti iš svetainės, kurioje yra prisijungęs.

CRLF ataka / HTTP response splitting

- Daugelis protokolų naudoja naują eilutę askirti informacijai.
- CR = Carriage Return
  - » Pvz: ASCII 0x0d.
- LF = LineFeed
  - » Pvz: ASCII 0x0a.

CRLF ataka / HTTP response splitting

- CRLF ataka
  - » Įsibrovėlis įterpia reikšmę 0x0d0a, kur vartotojo įvestis nėra tinkamai apdorojama.
  - » Yra sudaromi netikri žurnalo (angl. log) įrašai,...
- HTTP response splitting (HTTP užklausų padalijimas)
  - » Ypatingas CLRF atakos atvejis.

HTTP response splitting

- Įterpiamas CRLF į HTTP atsakomąją antraštę. (angl. response header).

```
$x=_GET['language']
header("Location: http://www.example.com/index_lang.php?language=$x");
```

```
GET /redirect.php?language=swedish%0d%0aContent-Length:%200
%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Length:%2032%0d%0a
%0d%0a<html>MallorysPage</html> HTTP/1.1
Host: www.example.com
...
```

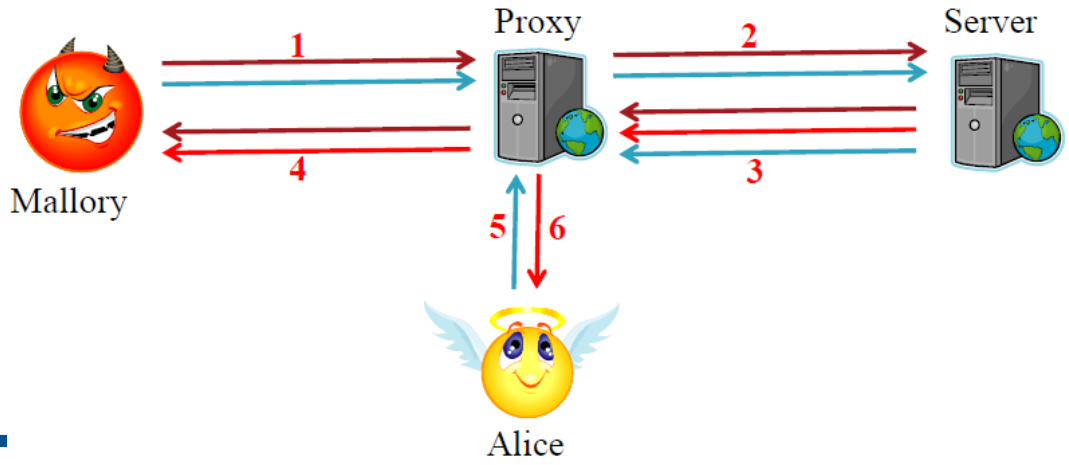
```
HTTP/1.1 302 Moved Temporarily
...
Location: http://www.example.com/index_lang.php?language=swedish
Content-Length: 0
```

```
HTTP/1.1 200 OK
Content-Length: 32

<html>MallorysPage</html>
...
```

HTTP response splitting

1. Mallory siunčia 2 užklausas serveriui naudojantis proxy (antra užklausa yra /index.php)
2. Proxy serveris persiunčia užklausas serveriui.
3. Serveris atsako į abi užklausas (proxy serveris mato kaip tris atsakomasias užklausas).
4. Proxy serveris persiunčia (sandėliuoja informacija) pirmus du „responses“ (trečias yra tiesiog išmetamas).
5. „Alice“ užklausia /index.php puslapį.
6. Proxy serveris perduotą Mallory‘s sandėliuojamą /index.php puslapį.



SQL injection

- XSS ir CSRF atakos yra svetainės lankytojų taikinys.
- SQL injekcija yra pačios svetainės taikinys – paprastai tai duomenų bazė.
- **Idėja.** Reikia mąstyti jog SQL užklausa yra vykdoma vartotojo įvesties nurodytomis reikšmėmis/parametrais – tada galima kontroliuoti kokie parametrai gali būti siunčiami užklausoms.

SQL injection

- Lengva apsaugoti nuo SQL injekcijos.
  - » Be abejo nevisada programuotojai taisyklingai įgyvendina formuodami užklausas arba tiesiog neįgyvendina duomenų validavimo, ko pasekoje yra pritaikoma SQL injekcija. Perimdami vartotojų paskyras, sunaikindami svarbia informaciją, bei atlieką kitą piktavališką veiklą.

MySQL query

- MySQL duomenų bazės užklausa yra atliekama naudojant funkciją mysql_query().

```
$uname = $_POST['username'];
$pass = $_POST['passwd'];
$result = mysql_query("SELECT * FROM login WHERE
                        username='".$uname."' AND password='".$pass."'");
if ($result) {
    if (mysql_num_rows($result) == 1) {
      session_regenerate_id();
      ...
}
```

- Problema: Nėra vartotojo duomenų įvesties tikrinimo/validavimo.
- Programuotojas tikisi jog įvestis atrodys: vartotojo vardas = „Alice“, slaptažodis = “slapt1547”.
  - » Programuotojo klaida: Kaikuriems vartotojams nerūpi, ko programuotojas tikisi.

MySQL injections

- Manyti pateikiama įvestis atrodo taip:
  - » Vartotojo vardas = Alice and slaptažodis = a ‘ OR ‘x’=‘x

Tada užklausos simbolių eilutė (angl. String) atrodys:

```
SELECT * FROM login WHERE username='Alice' AND password='a' OR 'x'='x'
```

Sprendimas: naudoti funkciją mysql_real_escape_string(), Kuri praleis specialius MySQL simbolius. Pavyzdžiui viengubos kabutės ‘ ir dvigubos kabutės ”.

Tai paprastai turėtų būti naudojama visada prieš siunčiant „string“ į užklausą. Bet ne tik siunčiant pateiktą slaptažodį „string“ pavidalu.

Comparing hashes (maišos funckijos reikšmių palyginimas)

- Geresnis prisijungimo testas, slaptažodžių maišos reikšmių palyginimas.
  - » Praeitas atakos tipas šiuo atveju nesuveiks žemiau pateiktame pavyzdyje:

```
$uname = $_POST['username'];
$pass = $_POST['passwd'];
$result = mysql_query("SELECT * FROM login WHERE
              username='".$uname."' AND password='".hash("sha256",$pass)."'");
if ($result) {
    if (mysql_num_rows($result) == 1) {
      session_regenerate_id();
      ...
}
```

Let's avoid MD5 and SHA-1

Kodėl nepatartina naudoti MD5 ir sha-1 ?

Comparing hashes (maišos funckijos reikšmių palyginimas)

- Taip pat, injekcija gali būti pritaikyta vartotojo vardui „username“ pvz:
  - » username=Alice’-- and passwd=anything

  -- MySQL šis ženklas reiškia komentarą, tai reiškia jog dalis už šito ženklo bus ignoruojama, tai „and passwd=anything“.

  Taigi priėjome išvada jog nepakanka naudoti mysql_real_escape_string() funkcijos ignoruojančios MySQL specifinius žymėjimus, simbolius, tik vartotojo vardui, reikia taikyti šią funkciją abiems „username“ ir „password“ .

Papildomos problemos

- mysql_real_escape_string() nevisada pakanka šios funkcijos.
- Kas jeigu bus naudojamas integer tipo kintamasis?
  - » Šiam tipui kabutės yra nereikalingos.

```
$id = $_POST['id'];
$result = sqlite_query($db, "SELECT * FROM users WHERE id={$id}");
```

  - » Jeigu įvestis id=0; DELETE * FROM users.
  - » Šiuo atveju niekas nebus ignoruojama ir užklausa sėkmingai bus įvykdoma, taip panaikindama visus lentelės duomenis.

Papildomos problemos

- **Sprendimas 1**: Naudoti kabutes net ir skaičiams.
- **Sprendimas 2**: Konvertuoti į integer arba float, jeigu tai reikšmės tipas, kurios tikėjaisi, o ne injekcijos pavidolo simbolių kratinys.(žr. Pavyzdyje).
- **Pavyzdys**: konvertuok_į_int(--), bandant konvertuoti tokį simbolį ar simbolių seką į sveiko tipo kintamąjį, nebus galimybės. Tačiau tai nėra optimaliausias variantas siekiant užtikrinti duomenų bazės saugumą nuo SQL injekcijų, programuotojas gali ir pamiršti, kaikuriuose vietose atlikti duomenų konvertavimą.

Paruoštieji sakiniai (angl. Prepared statments)

- Optimaliausias variantas siekiant apsaugoti nuo SQL injekcijų.
- **Idėja.** Atskirti SQL logiką nuo pateikiamų duomenų.
- Papildomai yra *pasiekiamas efektyvesnis* našumas „bendravimui“ su SQL:
  - » Logika siunčiama tik vieną kartą.
  - » Tiktai duomenys yra siunčiami užklausai.

Paruoštieji sakiniai (angl. Prepared statments)

- Prepared statment:

```
“SELECT * FROM login WHERE username=? AND password=?“
```

- Kiekvienai naujai užklausai nuspręsti kokius duomenis naudosite.
- „?“ klaustuko simbolį galime naudoti tik tam tikrose vietose, pvz: username=? ir password=?
  - » Šio simbolio naudojimas nėra galimas lentelių ir stulpelių pavadinimas nurodyti. Pačio klaustuko reikšmė yra išskyriama vieta tam tikram parametrui, kuris bus nurodomas naudojant tam tikras komandas, kurių panaudojimas yra pateikiamas sekančioje skaidėje.

Pavyzdys, paruoštųjų sakinių

- **Įspėjimas.** Paruoštųjų sakinių naudojimas galimas tik naudojant **mysqli** papildinį. Tokie sakiniai nebus įvykdomi jeigu naudosite mysql plėtinį.

```
$uname = $_POST['username'];
$pass = $_POST['passwd'];

$db = mysqli_connect('host','mysqlUser','mysqlPassword');

/*Prepare the statement by giving the SQL logic*/
$stmt = mysqli_prepare($db,"SELECT * FROM login WHERE username=? and password=?");

/*Bind parameters and result, execute and fetch parameters*/
mysqli_stmt_bind_param($stmt,"ss",$uname,$pass);
mysqli_stmt_execute($stmt);
mysqli_stmt_bind_result($stmt,$u_name,$u_pass,$u_email);
mysqli_stmt_fetch($stmt);

if ($u_name) {
  /*User is authenticated*/
  session_regenerate_id();
  ...
```

Duomenų bazės slaptažodžio slėpimas

- Netalpinti prisijungimo duomenų: vartotojo vardo bei slaptažodžio prie duomenų bazės jūsų, programos kode (angl. source code).

```
$db = mysqli_connect('host','mysqlUser','mysqlPassword');
```

- Geriau naudoti „httpd.conf“ failą ir jame nurodyti prisijungimą prie duomenų bazės...

```
<Directory /www/somefolder>
   php_value mysql.default.user     myusername
   php_value mysql.default.password mypassword
   php_value mysql.default.host     server.
</Directory>
```

Duomenų bazės slaptažodžio slėpimas

```
<Directory /www/somefolder>
   php_value mysql.default.user     myusername
   php_value mysql.default.password mypassword
   php_value mysql.default.host     server.
</Directory>
```

- Prisijungimui naudoti: $db = **mysqli_connect();**
- Arba su “aiškiais” parametrais:

```
$db = mysqli_connect(ini_get("mysql.default.user"),
                     ini_get("mysql.default.password"),
                     ini_get("mysql.default.host"));
```

Kitos apsaugos priemonės

- Paslėpti informaciją jog yra naudojamas PHP.
  - » expose_php = off, nuslepia faktą jog PHP yra naudojamas „response header‘iuose“.

  Užtikrinti jog phpinfo() negali būti pasiekiamas nuotoliniu būdu. Tai suteikia įsilaužėliui labai daug naudingos informacijos, pradėti piktavališkiems veiksmams skirtiems pakenkti jūsų WEB aplikacijai.

  Pakeisti failo tipo plėtinius interpretuojamus PHP, faile httpd.conf:

  AddType application/x-httpd-php .php - > gali būti pakeičiamas pvz: į .html ar kitą failo plėtinį. Taip apsunkinantys nepageidauntinų asmenų, kurie šnipinėja jūsų svetainę, siekdami išanalizuoti jūsų svetainės sandarą.

CAPTCHA

- CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart – visiškai automatizuotas viešas Tiuringo testas kompiuteriams ir žmonėms atskirti) – Carnegie Mellone universitete sukurtas testas, leidžiantis nustatyti ar tam tikra sistema naudojasi žmogus. Terminas atsirado 2000 metais. **Pagrindinė testo mintis**: pateikti vartotojui tokią užduotį, kurią lengvai sprendžia žmogus, tačiau kurią neįmanoma ar bent jau labai sudėtinga išspręsti kompiuteriui. Šiuo metu plačiausiai taikomos yra atvaizdų atpažinimo užduotys – vartotojui siūloma įvesti iškreiptus simbolius paveiksliuke.

CAPTCHA

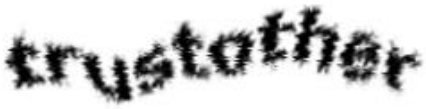
- CAPTCHA testai nebūtinai turi būti vizualiniai kaip įprasta. Bet kokia dirbtinio intelekto problema (pvz., kalbos atpažinimas), gali būti CAPTCHA testo pagrindu. Taip pat CAPTCHA testas gali būti toks, kuris reikalautų semantinio teksto supratimo (pvz., koks nors trivialus klausimas), ar nupiešto daikto apibūdinimo (pvz., parodoma gyvūno nuotrauka, o naudotojui reikia įvesti gyvūno pavadinimą).

CAPTCHA

[Captcha.net]

[Megaupload]

[NIH]



[Wikipedia]

[Reddit]

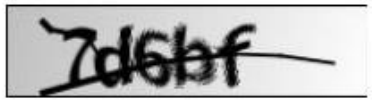
[eBay]

[Baidu]

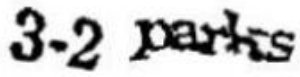
[Authorize]

[Digg]

[Slashdot]



[CNN]



[Recaptcha]

[Skyrock]

[Google]

[Blizzard]

CAPTCHA

- Kuriant CAPTCHA testus reikėtų atsižvelgti į tai:
  - » Nenaudoti fiksuoto ilgio generuojamo teksto.
  - » Naudoti keletą teksto iškraipymo metodikų.
  - » Jeigu naudojami žodynai (knygos, ar kito pobūdžio tekstiniai dokumentai), susikurti naujus žodynus, reguliariai juos atnaujinti. Taip pat pateikti ne vieną žodį, o kelis iš skirtingų žodynų, taip apsunkindami nepageidautinų asmenų naudojančių piktavališkas programas skirtas informacijos atpažinimui bei jos kaupimui taikant automatinį įvedimą į įvesties laukelį skirtą atlikti CAPTCHA testui.
  - » CAPTCHA testai gali būti sudaryti iš sunkiai suprantamų klausimų dirbtiniam intelektui, kuriuos galėtų atsakyti tik žmogus, klausimų sąrašas taip pat turėtų būti atnaujinamas.

CAPTCHA testo užduotis

- Kokį testą galėtumėte sudaryti, turint šiuos trijų skirtingų tipų: **Paveikslėlis, Klausimas, Žodis** objektus? Visi turimi objektai turi sąveikauti tarpusavyje.