

Cross-Site Request Forgery

Jodinėjam aukos sesiją

Pranešėjas: Martynas Kraujalis
martynas.kraujalis@gmail.com

CSRF – kas tai?

- Dar kitaip – Session Riding
- WEB sistemų pažeidžiamumas
- Žinomi atvejai nuo 1990m.
- Išnaudojamas web puslapio pasitikėjimas vartotoju
- Priverčia vartotoją atlikti jam nežinomus veiksmus pažeidžiamame puslapyje

Ar labai rimta?

- OWASP 2007 ataskaitoj – 5 vieta
- Kai kuriais atvejais – kodo vykdymas root teisėmis
- 2008 18mil. Korėjos eBay vartotojų prarado asmeninius duomenis
- CSRF + XSS = Samy MySpace worm

Kaip veikia?

- Pažeidžiami puslapiai - neprašo vartotojo autorizuoti kiekvienos užklauso (pasitiki sesijos ID)
- Priverčiam vartotoją išsiųsti užklausa
- Užklausa atlieka mūsų norimus veiksmus
- Ataka yra akla – mes nematom užklauso atsakymo vartotojui
- 2 užklauso tipai: GET ir POST

GET

- Nenaudoti GET užklausų veiksmų atlikimui, tik gavimui (HTTP specifikacija)
- Paprastas paveikslukas:

```

```

```

```
- Labai lengva išnaudoti (forumai...)
- Nereikia JavaScript

POST

- Irgi neišgelbės nuo CSRF
- `img src žymė` neveiks, bet...
- Nuoroda į kontroliuojamo turinio puslapį <http://hakeris.pvz.com/scriptas.php>
- Tereikia socialinės inžinerijos

```
<html>
  <body onload="fillframe()">
    <iframe name="someiframe" style="width:0px;height:0px;border:0px"></iframe>
  </body>
</html>

<script type="text/javascript">

function fillframe()
{

    mf = window.frames["someiframe"];

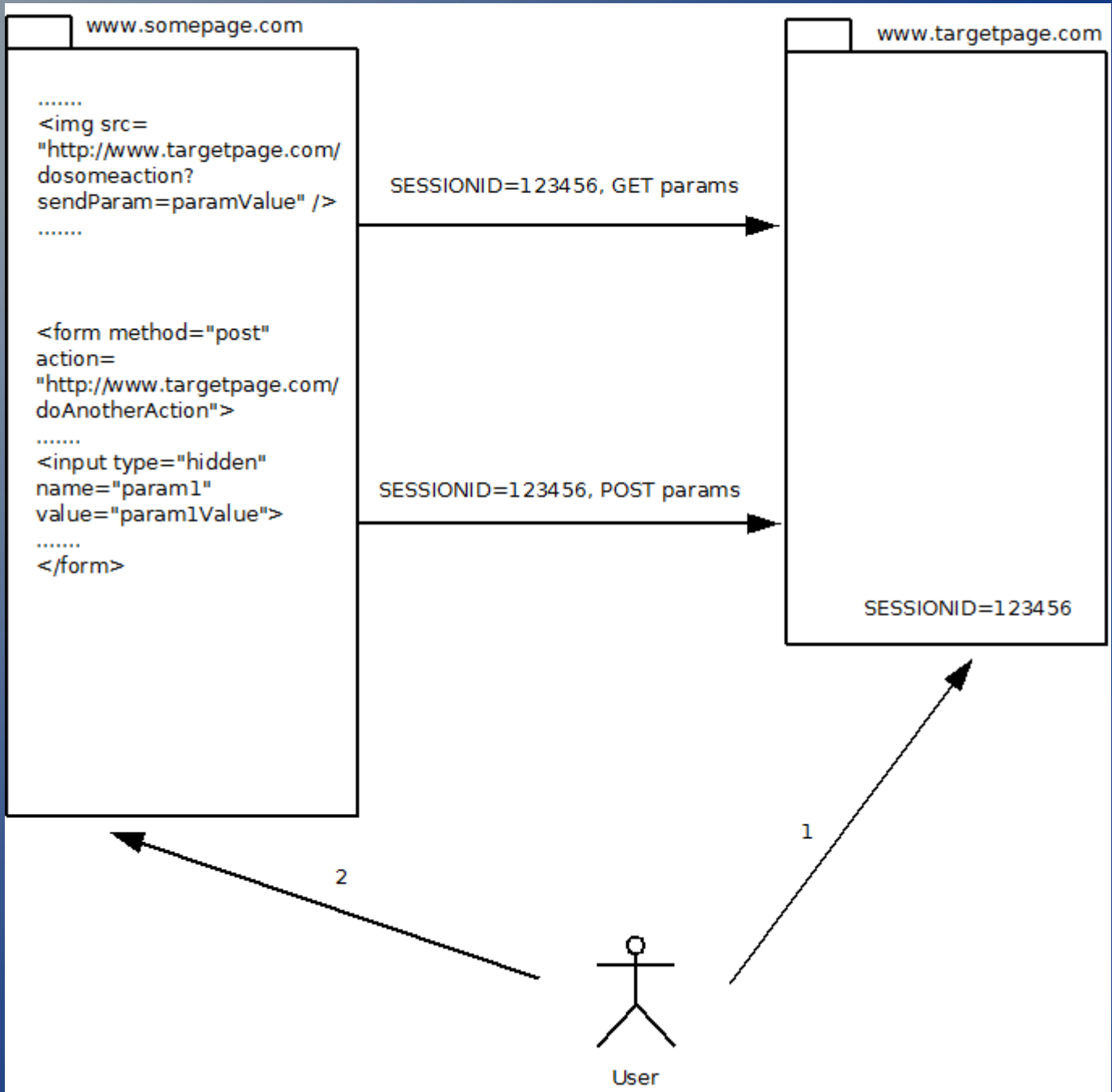
    html = '<form name="bankform" action="http://bankas.pvz.com/pervedimas" method="post">';
    html = html + ' <input type="hidden" name="suma" value="1000000"/>';
    html = html + ' <input type="hidden" name="kam" value="hakerioSaskNr"/>';
    html = html + '</form>';

    mf.document.body.innerHTML = html;

    mf.document.bankform.submit();

}

</script>
```



Ko reikia atakai?

- Taikinyis neturi tikrinti REFERER lauko (beveik visada)
 - Tam tikros naršyklių ir įskiepių klaidos leidžia klastoti REFERER
- Surasti URL'ą ar formą kuri atlieka reikiamus veiksmus
- Žinoti ir paduoti tinkamas parametrų reikšmes
- Privilioti auką į kenksmingą puslapį, kol ji yra prisijungusi atakuojamame puslapyje

Ką galima padaryti?

- Pakišti aukai savo prisijungimą – Login CSRF
 - kartu su crossdomain klaidomis buvo pademonstruota prieš [YouTube](#)
- Administravimo panelių atakos
 - firewall administravimas per web'ą
- Slaptažodžio keitimas
- Vidiniai tinklai, vidinės aplikacijos
- Daug kitų aplikacijų...

Kaip nesisaugoti?

- Vartotojo pusė, gali nedaug:
 - išėjus iš svetainės atsijungti
 - nenaudoti “atsiminti mane” funkcijos
 - el. pašto spam laiškuose nerodyti išorinių paveiksliukų, nespaudinėti nuorodų
 - MF plėtinys RequestPolicy ?

Kaip nesisaugoti?

- Serverio pusė, kas neapsaugos:
 - patvirtinimo langeliai, papildomi žingsniai
 - naudoti POST vietoj GET
 - tikrinti parametrą REFERER
 - užklausas su parametru X-Requested-With laikyti saugias (RoR, Django framework'ų klaidos)

Kaip apsisaugoti?

- Formose ir vykdančiuose URL'uose naudoti slapta žymę (token)
- Jei užklausa labai rimta, prašyti vartotojo autentifikuotis iš naujo
 - naudoja bankai
- Riboti sesijos galiojimo laiką
- Prižiūrėti crossdomain.xml faile Flash'ų teises
- Galima su JavaScript POST'u siųsti cookie turinį
 - cross domain policy saugo kad cookie bus pasiekiamas tik iš originalaus puslapio

Kaip apsisaugoti?

- Daug web framework'ų turi integruotas CSRF apsaugas
 - pasiimam POST užklausas netiesiogiai, per komponentą
 - formas generuojam per helper'ius (automatiškai prideda token'ą)
- Kaip dėl AJAX užklausų su X-Requested-With?
- Django - HTTP parametras X-CSRFToken
 - perrašom jQuery ajaxSend metodą, kad pridėtų X-CSRFToken

Šaltiniai

Wikipedia http://en.wikipedia.org/wiki/Cross-site_request_forgery

Criss Shiflet: <http://shiflett.org/articles/cross-site-request-forgeries>

SecureNet:

http://www.securenet.de/papers/Session_Riding.pdf

Django:

<https://docs.djangoproject.com/en/1.2/releases/1.2.5/#csrf-exception-for-ajax-requests>

Ačiū :)

Klausimai