Securitate Software

XI Vulnerabilități web

XSS, CSRF

Obiective

- prezentarea aspectelor teoretice din spatele vulnerabilităților Web comune
- prezentarea vulnerabilităților
  - Cross-Site Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)

Continut

CSRF

- Scenariu:
  - un utilizator este logat pe un site (cu un SessionID care nu a expirat)
  - chiar dacă închide fereastra browserului, sesiunea rămâne activă
  - dacă accesează un link de pe acel site își continuă sesiunea
- problema:
  - primește dintr-o altă sursă un link, ex.

`https://www.myBank.com/transfer.php?ammount1000&to=attacker`

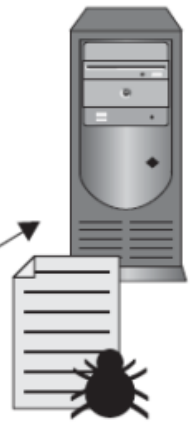
CSRF (II)

- de unde poate proveni link-ul?
  - un site malițios
  - un e-mail malițios (social engineering, spam)
  - mesaje pe site-uri de socializare, forumuri, etc.
- linkurile pot fi ascunse
  - din HTML (ex. "click aici pentru a vă vedea notele")
  - folosind URL-uri scurte (ex. goo.gl, tinyurl, etc)
  - IFRAME ascuns
  - <img src=...>
- se pot construi cereri HTTP POST malițioase

CSRF - exemplu

Step 0
The web site is sinfully architected to use text in the querystring as instructed from the user (read, delete, etc.).

Step 1
User logs on to the web site and authenticates.

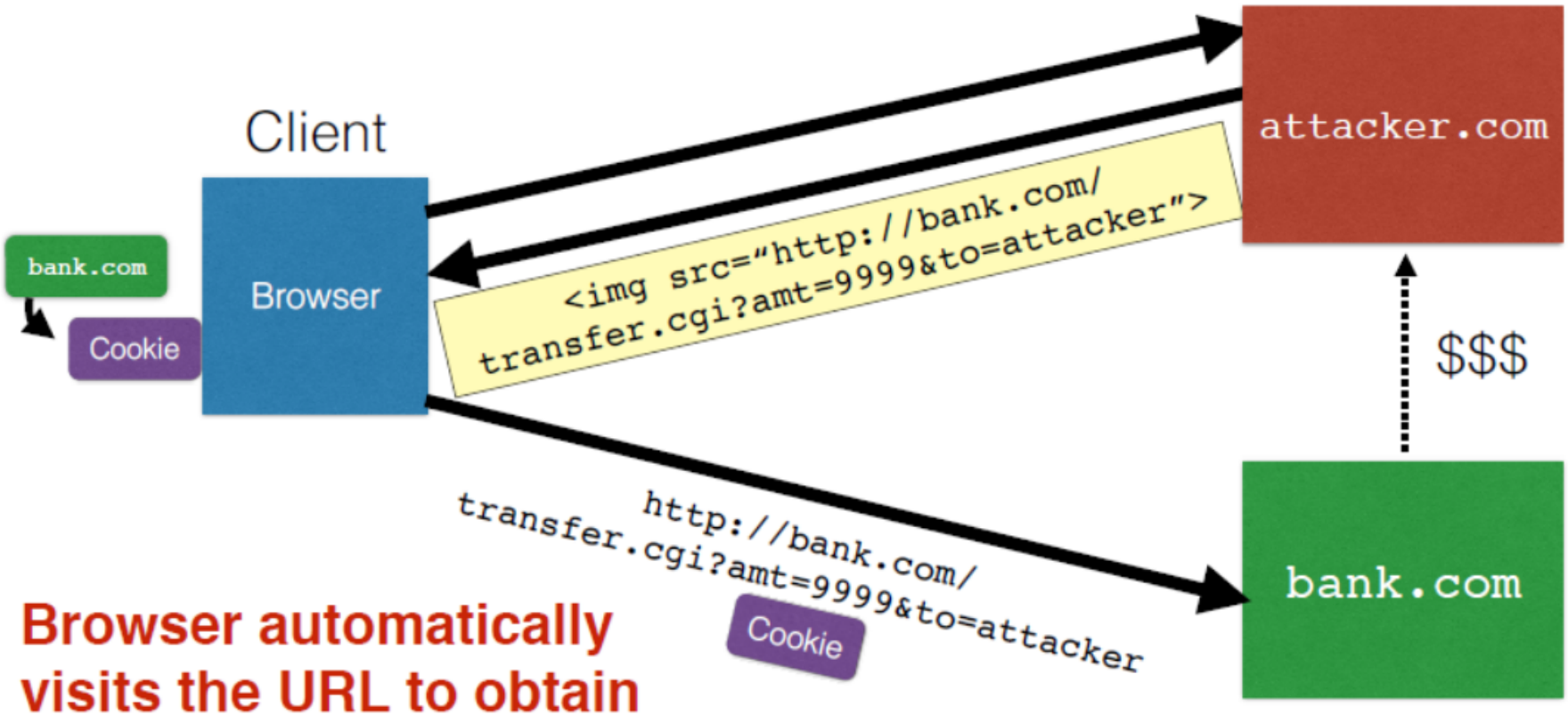


Step 3
Web server dutifully deletes the user's inbox!

Step 2
User opens a web page that includes a web application instruction in the query string.

<img src-http://www.server.com/foo.php?delete all>

CSRF - exemplu (II)



Browser automatically visits the URL to obtain what it believes will be an image

Protejarea împotriva CSRF

- verificarea câmpului REFERRER din cererea HTTP
- se acceptă doar pagini legitime de unde utilizatorul ar fi putut face cererea
- limitări
  - câmpul e opțional (nu este mereu prezent)
  - atac MITM (man in the middle)
    - cererea HTTP e modificată de un terț după ce aceasta este trimisă din browser, terț care interceptează comunicația

Protejarea împotriva CSRF (II)

- utilizarea unei chei speciale (token)
- poate fi inserată într-un hidden POST field, custom HTTP header, parametru GET, etc.
  - trebuie sa fie cât mai greu de prezis (la fel ca session ID)
  - ruby-on-rails folosește automat astfel de chei pentru toate linkurile
  - varianta recomandată
- problema de design
- cheia nu trebuie inclusă în cookie (poate fi aflată)
- session timeout
- 2-factor pentru tranzacții

XSS

- tipuri
  - stored XSS (persistent, type 2)
  - reflected XSS (non-persistent, type 1)
  - DOOM-based (type 0)
- atacuri
  - JavaScript injection
  - browser redirection
  - IFRAME injection
  - furt cookie sau session ID
- atacurile XSS sunt "interactive", depind de existența altor utilizatori activi în aplicație

XSS - pagini web dinamice

- codul trimis de server e dinamic
- JavaScript - limbaj foarte utilizat, client-side
- paginile sunt mult mai interactive
- conținutul poate fi modificat (de obicei DOM)
- se pot urmări acțiuni (mouse, tastatură)
- se pot efectua cereri HTTP și se pot interpreta răspunsuri
- se pot menține conexiuni permanente (AJAX)
- se pot citi și scrie cookie-uri

XSS / JS - implicații de securitate

- scripturile JS pot accesa date sensibile
- un script pe un site A nu ar trebui să poată accesa date ale paginilor de pe un site B
  - ex. scripturi de pe *attacker.com* nu ar trebui să:
    - poată modifica layout-ul paginilor de pe *bank.com*
    - acceseze apăsări de taste de pe *bank.com*
    - acceseze cookie-uri ale *bank.com*
- defensivă: SOP (Same Origin Policy)

XSS - vectori de atac

- serverul poate deservi cod JavaScript malițios, generat de atacator
- scopul este de a rula codul în browserul clientului
- abilitatea de a lăsa conținut pe server este punctul de intrare al unui astfel de atac
- problema constă în validări insuficiente (sau lipsa validărilor) asupra conținutului pe care utilizatorii îl pot stoca pe server

Stored XSS (persistent, type 2)

- aplicație guestbook
  - oricine poate lăsa comentarii
  - toate comentariile sunt vizibile într-o listă de către ceilalți utilizatori
  - un utilizator introduce ca și comentariu *<script >alert('XSS!')</script>*
  - scruptul se va executa în browserul tuturor celor care accesează lista de comentarii
- deosebit de grav, utilizatorul trebuie doar să acceseze pagina respectivă, de pe un site considerat sigur

Reflected XSS (non-persistent)

- serverul preia date din cererea HTTP și le "reflectă" în raspunsul HTTP
- exploatarea are loc atunci când un atacator provoacă victima să trimită către serverul vulnerabil o cerere cu conținut malițios, conținut executat în browser
- ex. parametru în URL
  `http://example.com/page?var=<script>alert('xss')</script>`
- câmpul *var* e afișat ca atare în raspunsul HTTP
- URL poate fi obfuscat (HTML escape sau folosind JavaScript) sau se pot folosi URL-uri scurte

DOM-based XSS

- un atac strict pe partea clientului (nu e implicată o comunicare client-server)
- numele provine de la gestiunea incorectă a obiectelor DOM (Document Object Model)
- clientul accesează un URL malițios
- conținutul URL-ului este interpretat de JavaScript și afișat în pagină
- ex:
  `http://example.com/page.html?var=Mary`
- pagina conține scriptul:
  var pos=document.URL.indexOf("var")+4;
  document.write(document.URL.substring(pos,
  document.URL.length));
- un atacator poate furniza următorul URL
  `http://example.com/page.html?var=<script>alert('xss')</script>`

XSS - atacuri

- JavaScript injection
  - acces la datele sensibile
  - modificarea conținutului paginii
  - clickJacking
- browser redirection
  - redirectare către pagini malițioase / reclame
- IFRAME injection
  - affiliate ads, malware scripts
- stealing cookies & session IDs

XSS - protejare

- input validation
  - filtrare / escape
  - neacceptarea codului JavaScript / HTML
  - sursa problemei poate fi ascunsă (în spatele unei interfețe dintr-un framework)
- testare manuală
- testare automată

XSS - exemplu Yahoo Mail mobile

- persistent XSS (stored)
- raportat către Yahoo în 11 noiembrie 2015
- e-mail cu conținutul
  ' "><svg/onload=prompt(1337)>
- la accesarea mesajului codul malițios se executa automat
- reparat în 21 noiembrie 2015
  `http://pwnrules.com/persistent-xss-yahoo-mail-inbox/`

Bibliografie

- “24 Deadly Sins of Software Security”, chapter 1, 2, 3, 4, pp. 3 – 88
- XSS vulnerability found on mobile site of Yahoo! Mail, http://www.scmagazineuk.com/xss-vuln-found-on-mobile-site-of-yahoo-mail/article/457357/