Hacking cookies in modern web applications and browsers

Dawid Czagan

About me

- Founder and CEO at Silesia Security Lab
- Bug hunter: security bugs found in Google, Yahoo, Mozilla, Microsoft, Twitter, Blackberry, ...
- Trainer: *Hacking web applications – case studies of award-winning bugs in Google, Yahoo, Mozilla and more*
- E-mail: dczagan@silesiasecuritylab.com
- Twitter: @dawidczagan

Motivation

- Cookies store sensitive data (session ID, CSRF token, ...).
- Multi-factor authentication will not help, when cookies are insecurely processed.
- Security evaluators underestimate cookie related problems.
- There are problems with secure processing of cookies in modern browsers.
- Consequences: authorization bypass, user impersonation, remote cookie tempering, SQLi, XSS, ...

Agenda

- Three perspectives of insecure cookie processing:
  - web application
  - browser
  - RFC 6265
- Selected problems will be discussed.

Secure flag & HSTS

- Secure flag set -> cookie sent only over HTTPS.
- Problem: insecure HTTP response can overwrite a cookie with Secure flag (RFC 6265; all browsers affected).
- HSTS (HTTP Strict Transport Security) implemented -> HTTPS traffic enforced.
- Problem: HSTS is not supported by Internet Explorer 10.
- Recommendation: use HSTS and Secure flag.

Importance of regeneration

- User is logged out and attacker learns user's cookie with session ID (XSS, disclosure over insecure HTTP, ...).
- Problem: session ID has not been regenerated after successful authentication.
- Consequence: user impersonation
- Other examples: CSRF token, persistent login token, ...

Server-side invalidation

- User logs out and cookie with session ID is deleted in user's browser.
- Problem: no server-side invalidation
- Consequence: user impersonation
- Assumption: attacker learned user's session ID, when user was logged in (XSS, disclosure over insecure HTTP, ...).

HttpOnly flag

- JavaScript cannot read a cookie with HttpOnly flag.
- Problem: access permissions are not clearly specified in RFC 6265.
- Cookie with HttpOnly flag can be overwritten in Safari 8
- Attack #1: switching a user to attacker's account.

  Profit: user enters credit card data to attacker's account.
- Attack #2: user impersonation

  Assumption: user logs in and session ID is not regenerated.

Domain attribute

- No domain attribute specified -> cookie will be sent only to the domain from which it originated (RFC 6265).
- Problem: Internet Explorer 11 will send this cookie additionally to all subdomains of this domain.
- Attack #1: cross-origin cookie leakage

  XSS on insensitive x.example.com has access to a cookie from sensitive example.com/wallet
- Attack #2: cookie leakage to externally managed domain (shared hosting)

Cookie tampering

- Problem: Safari 8 supports comma-separated list of cookies in Set-Cookie header (obsoleted RFC 2109).
- /index.php?lang=de,%20PHPSESSID=abc
- Attack #1: switching a user to attacker's account
- Attack #2: user impersonation

  Assumption: PHPSESSID is not regenerated after successful authentication.
- Attack #3: XSS via cookie
- More powerful attack with browser dependent exploitation

Underestimated XSS via cookie

- XSS via cookie ≠ local exploitation
- Remote attack #1: cross-origin exploitation

  XSS on insensitive x.example.com is used to launch XSS via cookie on sensitive y.example.com
- Remote attack #2: response splitting
- Remote attack #3: cookie tampering in Safari 8

Conclusions

- Security engineers/researchers should:
  - educate development teams
  - cooperate with browser vendors
  - discuss/improve RFC 6265

Thanks

Q&A