Web Application Pentesting

Vivek Ramachandran

SWSE, SMFE, SPSE, SISE, SLAE, SGDE Course Instructor

Certifications: http://www.securitytube-training.com

Pentester Academy: http://www.PentesterAcademy.com

Mitigating CSRF with Tokens

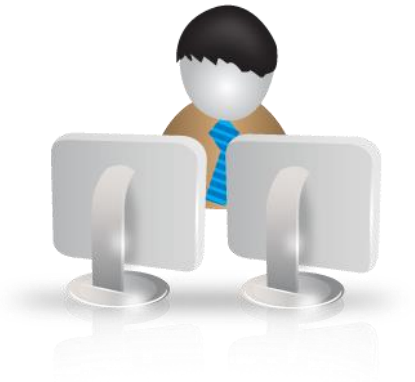
Why does CSRF work?

- What is the problem?
  - Cookie is sent automatically with requests
    - Attacker need not know the value of the cookie
  - No other “session” or “operation” validation
  - How do we know the user is logged in and using the website?

Tokens: Per Session / Request

- Solution to CSRF
  - Identify critical operations in the Application
    - Basically URLs / API endpoints
  - Each critical operation request must be accompanied with a “token”
  - Token is
    - Long, Random, not repeated for application lifetime
    - Unique per session or even per operation
    - Part of URL in GET
    - Hidden Field in POST (forms)
    - Attacker cannot know / predict this token and hence cannot create requests to exploit the operation

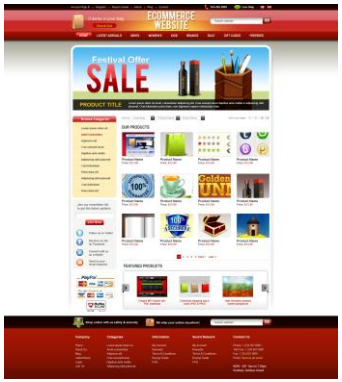
User Interaction without Tokens



HTML Form

POST Form Data

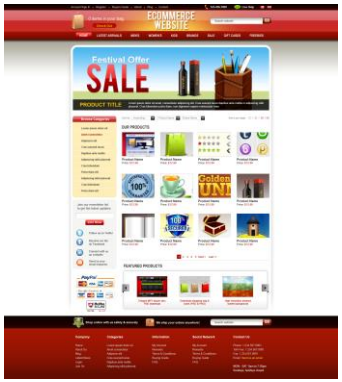
Cookie



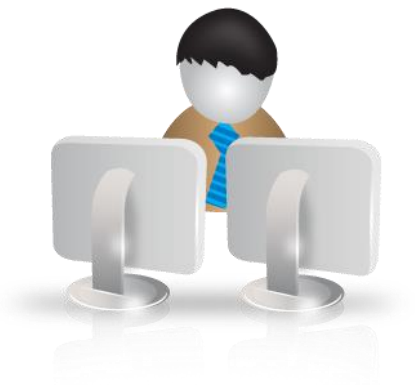
ACCEPTS

POST Form Data

Cookie



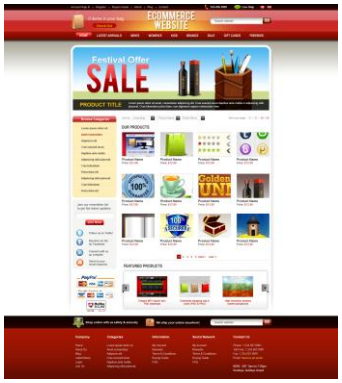
User Interaction with Tokens



HTML Form + **TOKEN**

POST Form Data + **TOKEN**

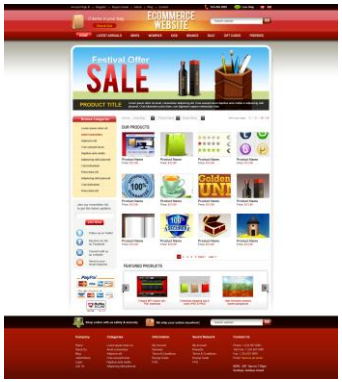
Cookie



TOKEN Missing! REJECT!

POST Form Data

Cookie



Demo

- OWASP WebGoat

https://github.com/WebGoat/WebGoat/wiki/Installation-(WebGoat-6.0)

What could go wrong? Token Security

- Is the token long enough?
- Is it random enough? Entropy?
  - Should not be derived from known / predictable data
- Never repeated or reused?
- Stored insecurely
  - GET requests in history files, server logs etc.
- Is sent over a secure medium? HTTPS?
  - Sniffing and MITM

OWASP CSRF Guides

- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
- https://www.owasp.org/index.php/Reviewing_code_for_Cross-Site_Request_Forgery_issues

Pentester Academy