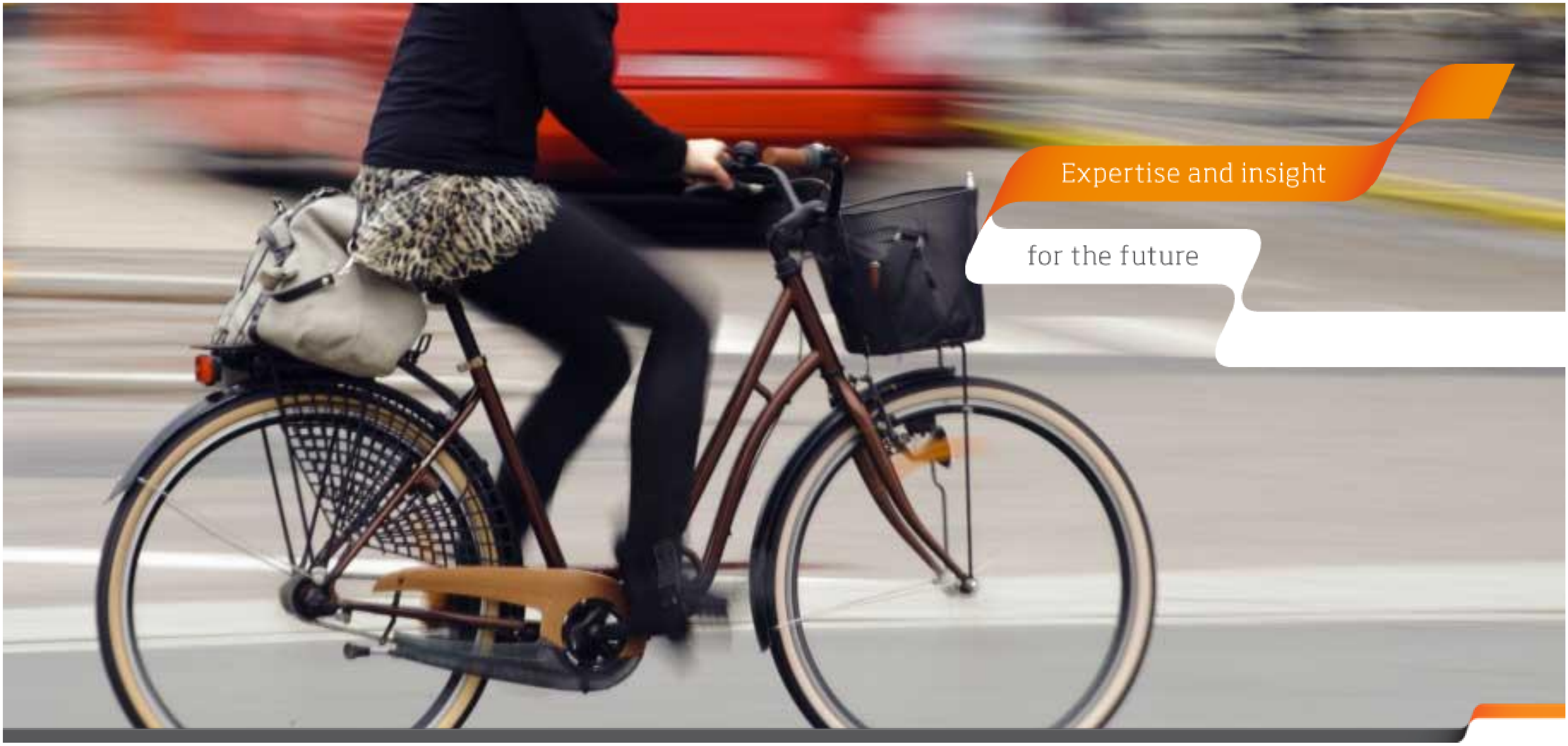


Attacking other users

XSS and CSRF

Helsinki Metropolia University of Applied Sciences

Attacking other users

- Cross-Site-Scripting XSS
  - Exploits missing input validation in web application code
  - Basic root cause is similar as in SQL-injection, but in XSS the user data is not used in database
  - User given data is embedded in HTLM without input filtering -> malicious user can control the HTML
  - With XSS attacker can execute arbitrary JavaScript code in victims browser
- Reflected / Stored / DOM-based XSS

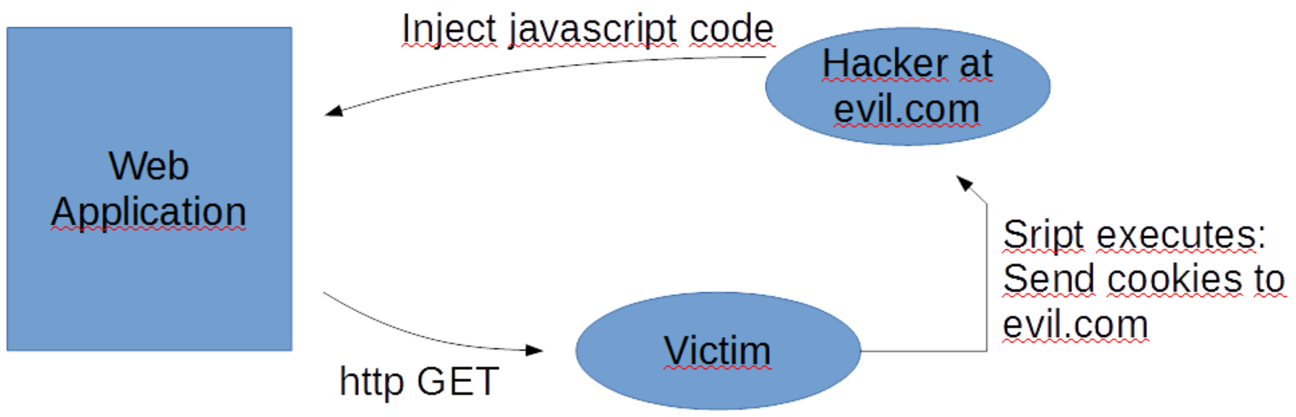
Helsinki Metropolia University of Applied Sciences

Reflected XSS

- The most common XSS type
- User input is not stored in web application
- Usually attacker provides a link to the victim
  - Link points to XSS vulnerable page with attack payload
  - URL shortener services can be used to masquerade the attack
  - Victim clicks the link and the attack payload is executed
- Well crafted attack is very difficult to notice

Stored XSS

- Unfiltered user input is stored in the web application
- Extremely dangerous vulnerability
- All users visiting the vulnerable page will be attacked



DOM XSS

- Document Object Model based XSS
- Client side Javascript uses URL to modify webpage content
- Attacker crafts a malicious URL with attack payload
    - Client side JavaScript modifies the webpage HTML and inserts the attack

XSS payloads

- Session cookie stealing a.k.a session stealing
- Sensitive data stealing
- Virtual defacement
- Stealing usernames and passwords

- Basically anything you can do with JavaScript

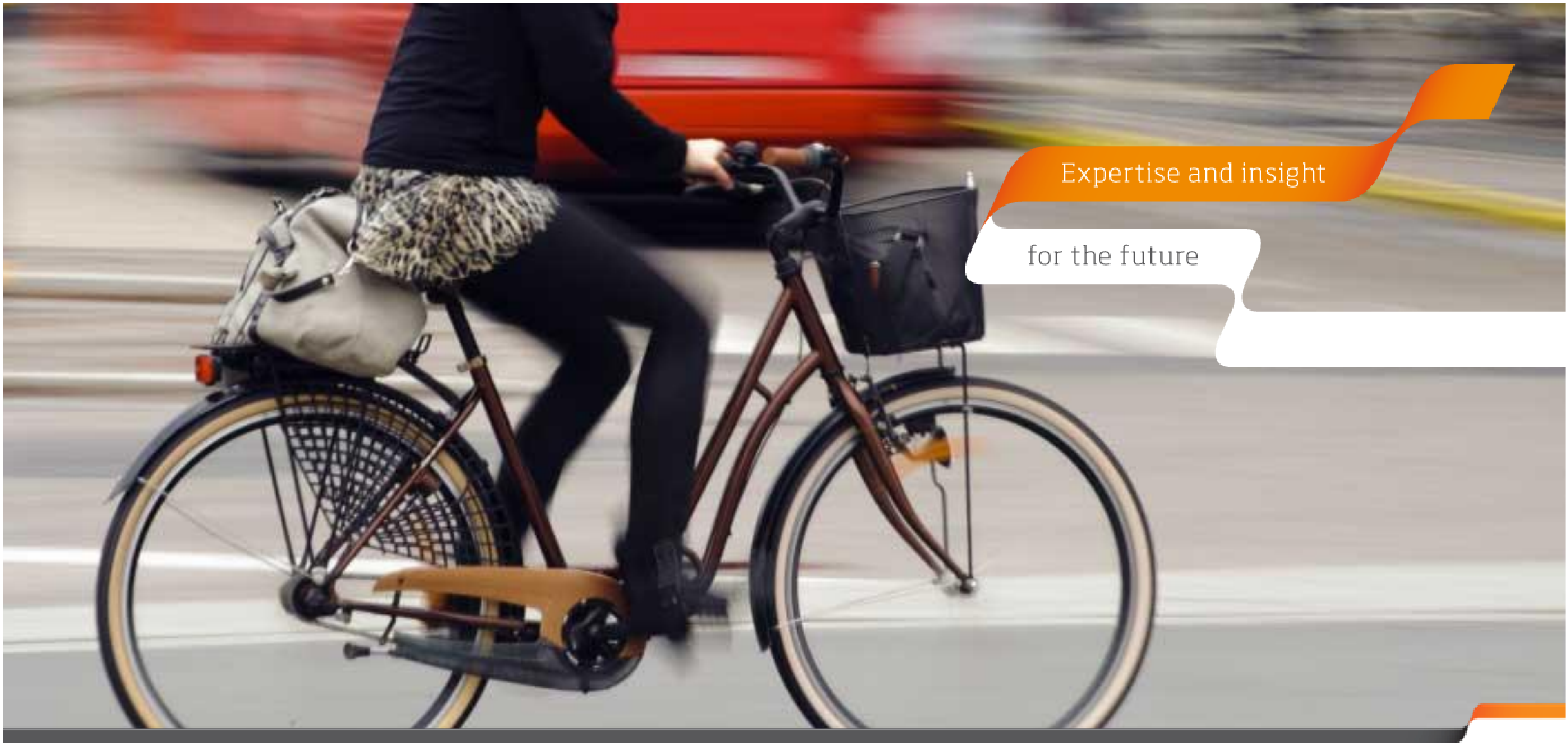
Finding XSS vulnerabilities

- Goal: Modify parameters (POST/GET) and make that appear in HTML

1. Insert a known text to parameter such as testxxtest
2. Find the text in page source
3. Insert special characters to see if there is any filtering (<>!—’’)
4. If no filtering is implemented, XSS is verified
5. If filtering is in place, try to evade filtering

Cross Site Request forgery

- Basic functionality
  - Malicious website makes a request to valid URL in a web application (such as banking application)
  - Session cookies of the web application are added by the browser since the URL is valid
  - Malicious website can therefore execute functionality on the web application without victim’s concession
- Relies on predictable URLs
  - CSRF tokens are added in URLs to defend against CSRF



THANK YOU!

- www.metropolia.fi/en/
- www.facebook.com/MetropoliaAMK
- Kimmo.Sauren@metropolia.fi

Helsinki Metropolia University of Applied Sciences