Kentico

Web application security

Juraj Komloši

Agenda

Kentico

- Why is web app security so important?
- OWASP Top 10
- Cross-site scripting (XSS)
- SQL injection
- Input validation
- Client/server validation
- Session attack – session fixation
- Clickjacking
- CSRF
- Argument injection

Why is web application security so important?

Kentico

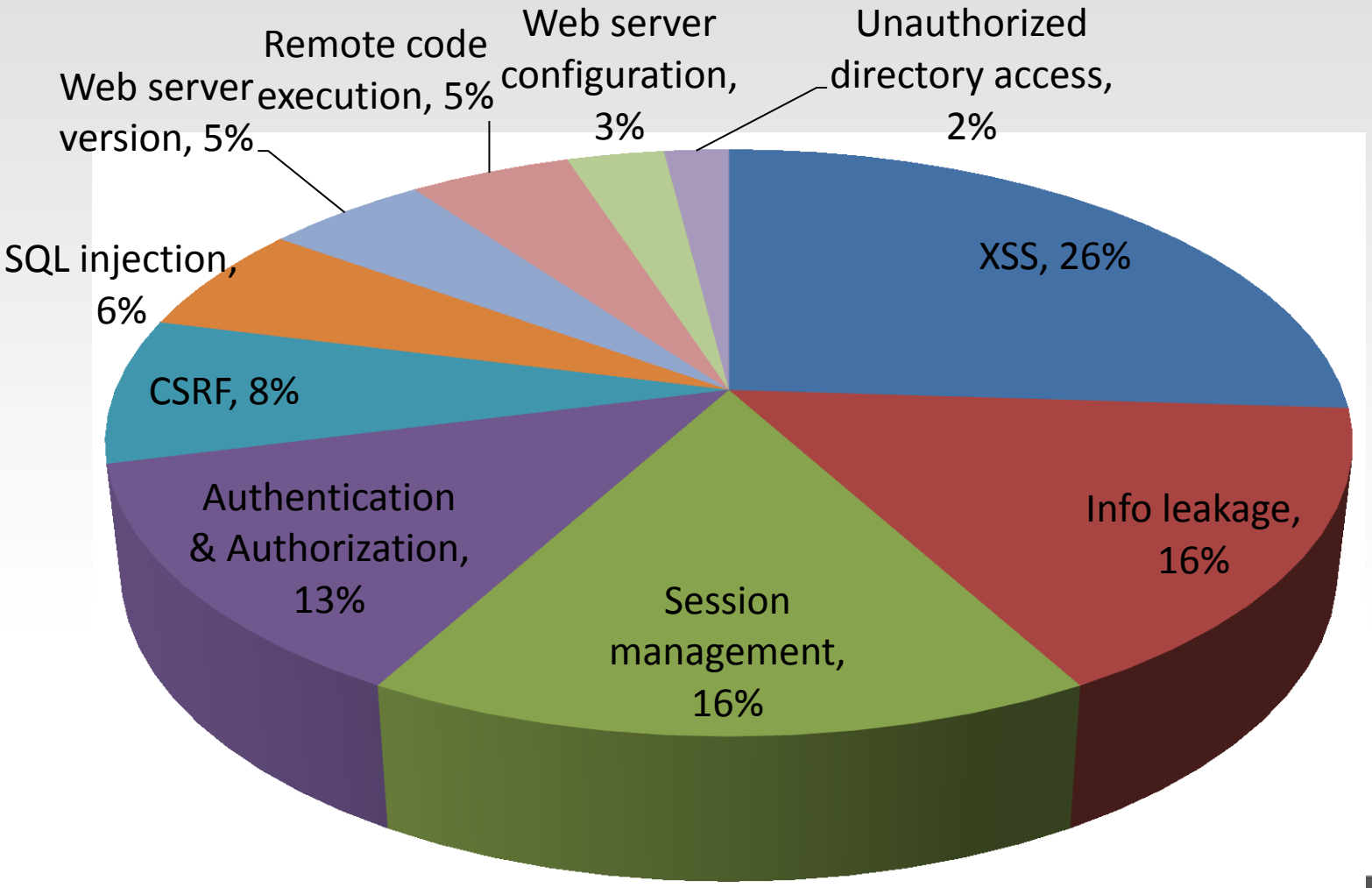
- **Identity theft (session attacks)**
  - Allow access to illegitimate functions or data
- **Compromising application (SQLi, XSS, LDAP...)**
  - Availability, integrity and confidentiality could be affected
- **Loss of service - DoS, DDoS**
- **Loss of reputation for organizations**
- **Attack's reasons:**
  - Commercial - Sony's PlayStation Network, Acer Europe,
  - Political - Iran's Ministry of Foreign Affairs,
  - Profit - e-shop administration, BitCoin stock exchange

OWASP Top 10

Kentico

- **OWASP - The Open Web Application Security Project**
  - open-source web application security project
- **OWASP Development Guide**
  - covers an extensive array of application-level security issues, from SQL injection through modern concerns such as phishing, credit card handling, session fixation, cross-site request forgeries, compliance, and privacy issues.
- **References**
  - OWASP official site
  - OWASP development guide
  - OWASP testing guide

2013 Application Vulnerability Population



http://blog.cenzic.com/2013/03/application-vulnerability-trends-report/

Cross-site scripting (XSS)

Kentico

- Enables attackers to inject client-side script
- Inject malicious code segments that are run by your server in the victim's browser
- Main types:
  - Persistant - the code is added to the web site
  - Non-persistant - the malicious code is contained in the URL
- Common targets:
  - Cookies
  - Social engineering
- XSS sources:
  - URL, Flash, videos…

Cross-site scripting (XSS)

Kentico

- **Result:**
  - Identity theft
  - Accessing sensitive or restricted information
  - Potential DoS attack
- **Finding XSS:**
  - Insert javascript code to:
    - GET/POST parameters, form fields etc.
- **Avoiding XSS:**
  - Encode user inputs -> HttpUtility.HtmlEncode
  - Web.config - > validateRequest="true"
  - Protect cookies -> HttpOnly flag
- **Demo**

SQL injection

Kentico

- Executing SQL code through a web application
- Result:
  - Attacker can read all data or the database schema, change it, edit it
- Finding SQL injection:
  - Insert ‘ (aphostrophe) to:
    - All inputs – GET/POST parameters, input fields
- Avoiding SQL injection:
  - Escaping aphostrophe
  - Stored procedures -> use query parameters
  - Do not use ‘exec’ function
- Demo

Input validation

Kentico

- All user input is evil
- White list validation
  - Involves defining what IS authorized
- Character-set – accept only expected set of characters
  - Amount - digits, zip code – regular expression
  - US states – drop down list
- Data format – accept only data containing the proper format
  - e-mail – letters, numbers, „@“, dots
- Escaping special characters
- Black list validation
  - Involves defining what IS NOT authorized
  - Code names – can not contain special chars ($#_.,)

Client/Server validation

Kentico

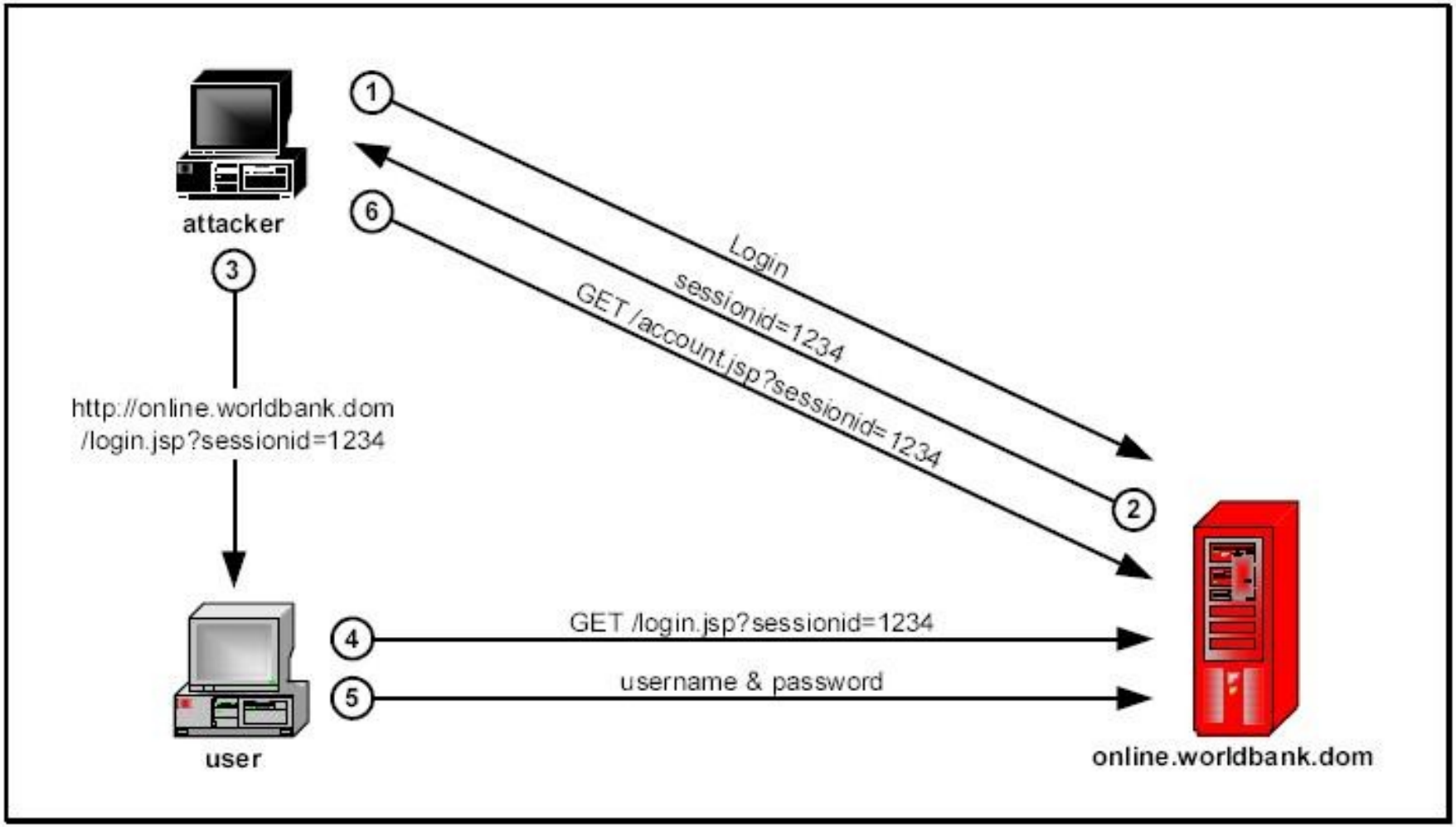
- **Client validation:**
  - Gives the user immediate feedback
- **Server validation:**
  - More advanced validations
- **Why do we need server side as well as client side validation?**
  - Client side validation may be subverted
- **Common mistake:**
  - Disabled UI doesn't allow to perform any actions
  - Form inputs are not validated
- **Demo**

Session attack – session fixation

Kentico

- Attacker fixes the user's session ID before the user logs into target server
- Eliminate the need to obtain user's session ID afterwards
- How it works:
  - Attacker logs in to the server (get SessionID)
  - Attacker sends link containing logon page with session ID to the victim
  - Victim opens link (session already exists, a new one is not created)
  - Victim logs in (using attacker session ID)
  - Attacker can access victim's account

Session attack – session fixation



Session attack – session fixation

Kentico

- Fixation SessionID can be done by:
  - URL argument
    - ~/login.aspx?session=123456
  - Hidden field
    - Impractical
  - Cookie
    - The most commonly used / the most vulnerable
- How can be cookie issued to the browser
  - Cross-site scripting (XSS)-> document.cookie="123456"
  - Meta tag injection
    - /<meta%20http-equiv=Set-Cookie%20content="sessionid=1234; %20Expires=Friday,%201-Jan-2010%2000:00:00%20GMT">.idc
- Demo

Clickjacking

Kentico

- UI is redesigned to carry some script code along the original code
- Tricks a user into performing undesired actions by clicking on a concealed link
- Clicking the visible buttons on the clickjacked page vs. performing actions on the hidden page
- How to avoid clickjacking:
  - X-Frames-Options: sameorigin | deny
- Demo

CSRF - Cross Site Request Forgery

Kentico

- Allows an attacker to take arbitrary actions as the victim against a web site
- How it works:
  - Victim is logged on internet banking portal
  - Attacker crafts HTML image element that references to bank portal
    - <img src="http://bank.example/payment?account=XY&amount=10000&for=XYZ">
  - Victim visits attacker web site
  - Victim's browser execute request with victim's cookie
- How to avoid CSRF:
  - Using POST instead of GET
  - Implementing secret tokens
- Demo

Argument injection

Kentico

- Attack based on tampering with input parameters
- Result:
  - Attacker can see data which he normally can not see
  - Attacker can modify data which he normally can not modify
- How to find argument injection:
  - Focus on query parameters
  - Try to enumerate integer values in query strings (e.g. IDs)
- How to avoid argument injection:
  - Changing query parameters to e.g. GUID (less predictive)
  - Check user permissions before modifying objects
- Demo

Q&A