Session Fixation – the Forgotten Vulnerability?

OWASP

Michael Schrank†,
Bastian Braun†,
Martin Johns*,
Henrich C. Pöhls†

† Institute of IT-Security and Law,
University of Passau

* SAP Research

The OWASP Foundation
http://www.owasp.org

Outline

- Background
- Exploits & Impact
- Practical Experiments Outcome
- Solution: Session Fixation Protection Proxy
- Conclusion & Future Work

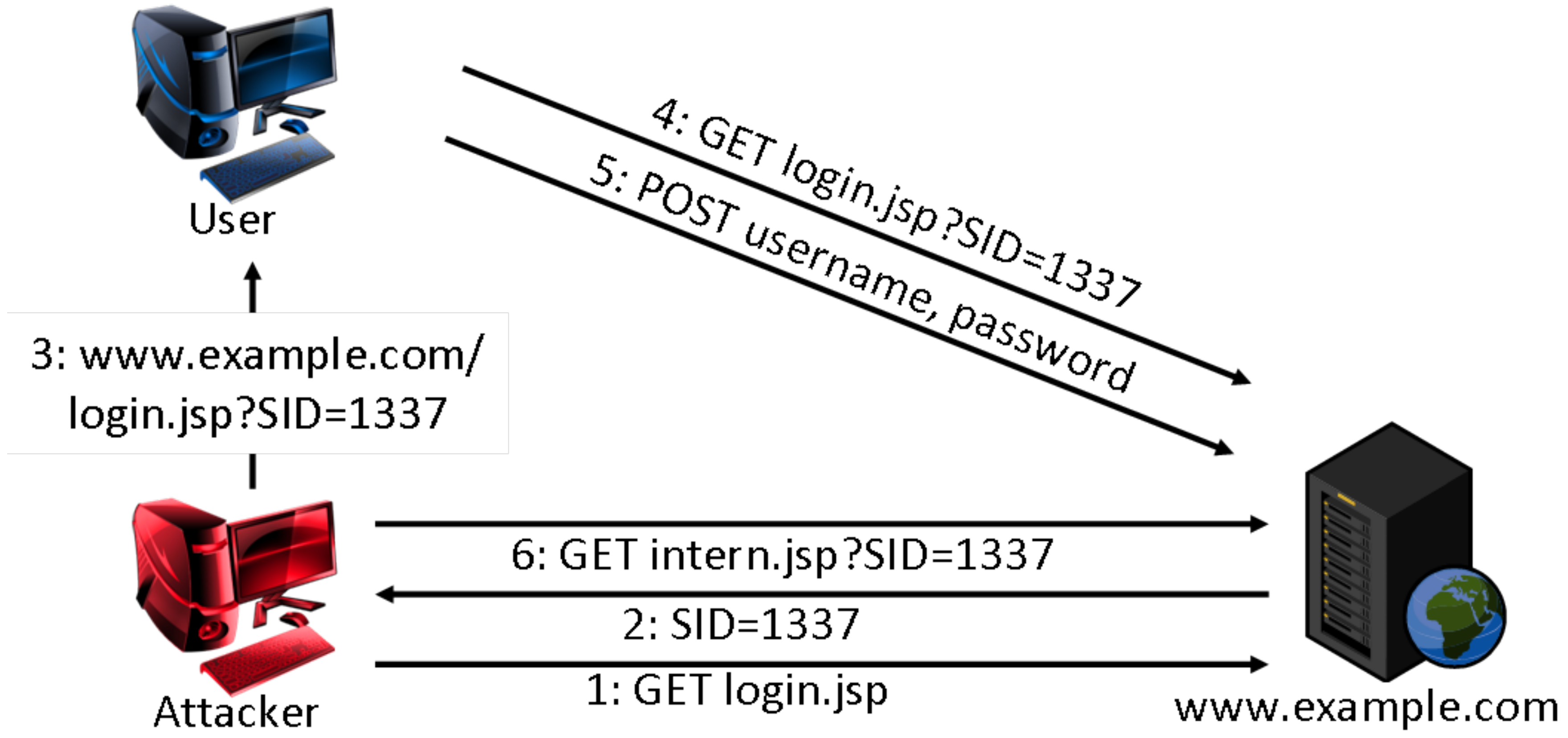
Background

- **Session Fixation known for several years (at the latest from 2002)**
- **Little attention compared to XSS, SQLi, CSRF**
  - Little awareness in developers' world
- **Session Management not provided for HTTP (stateless)**
  - Fallback procedure: session tracking by identifier (ID)
    - Cookie, URL parameter, hidden form field
    - Carry ID with every request
- **Session Management + Authorization Management**
  - Mismatch of responsibilities: framework vs developer
  - Session management done by programming framework/ application server
  - User authentication/authorization is application's duty

Session Hijacking Reloaded – Session Fixation

- **Attack sketch**
  - Attacker sets victim's session ID instead of session ID theft
  - Victim authenticates using attacker provided session ID
  - Attacker resumes authenticated session making use of known session ID
- **Session Fixation starts before user authentication**
- **Attack vectors: two attack stages**
  - 2nd stage: Session's authentication level is raised for the provided "fixed" session ID
  - 1st stage: needs other vulnerability to set SID
    - XSS, meta tags, cross-protocol attack, sub domain cookie bakery, http response splitting, http header injection

Session Fixation in a nutshell


User
4: GET login.jsp?SID=1337
5: POST username, password
3: www.example.com/ login.jsp?SID=1337
6: GET intern.jsp?SID=1337
2: SID=1337
1: GET login.jsp
Attacker
www.example.com

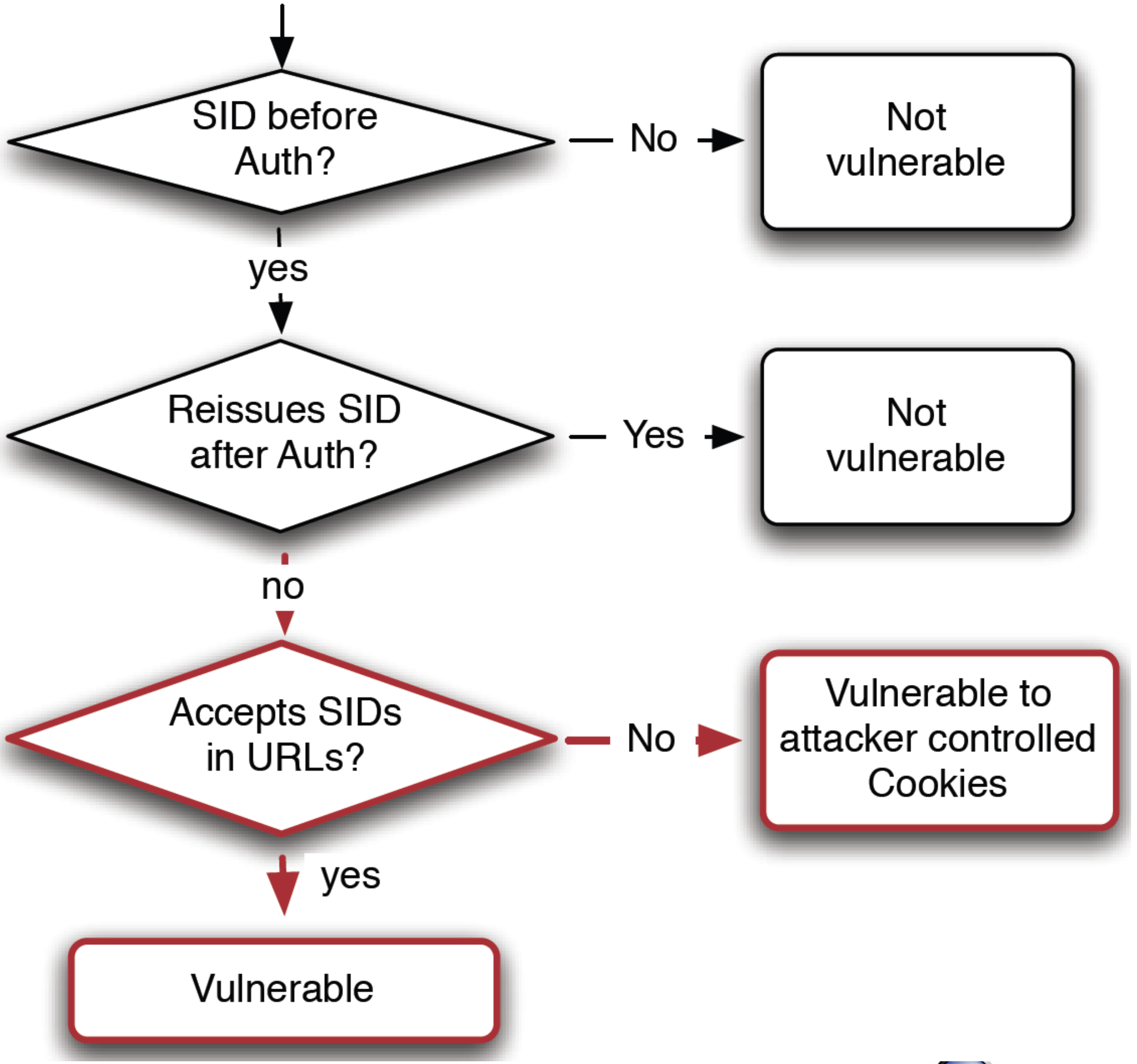
Impact & Discussion

- **First stage attack preconditions**
  - Mislead victim into clicking on a link
  - Set cookie via other vulnerability
  - Make the victim log into his account and meet that time frame
- **Session Fixation preconditions**
  - Application is vulnerable
  - If session is bound to IP or browser: additional obstacle
  - Individual session ID needed for every victim
- **But: if all conditions are met Session Fixation is severe attack**
  - Full impersonation of victim mostly without any notice

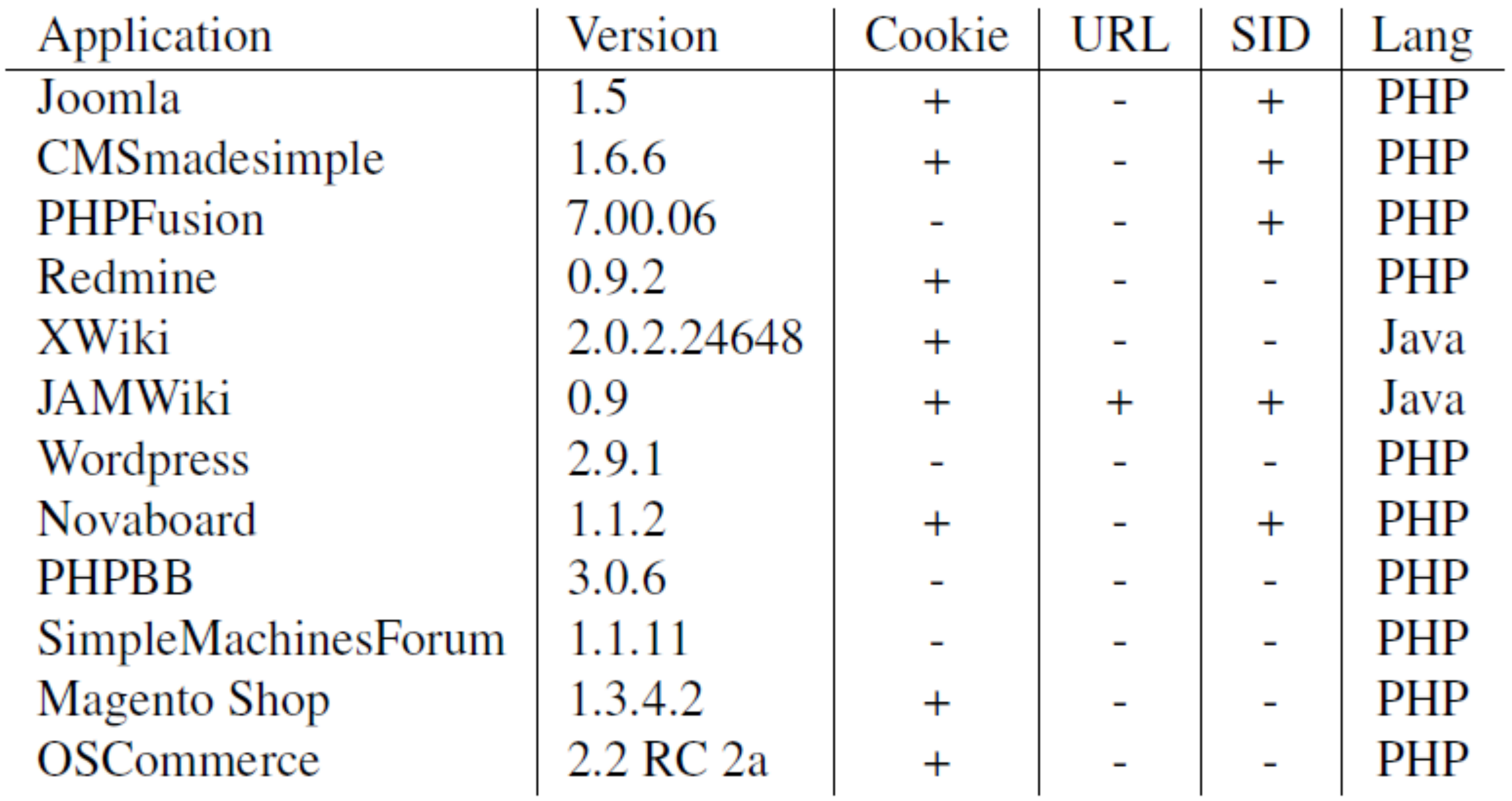
1st case study: open source CMS

Default configuration vulnerable to Session Fixation?

- If yes, we "only" need first stage attack



1st case study: open source CMS



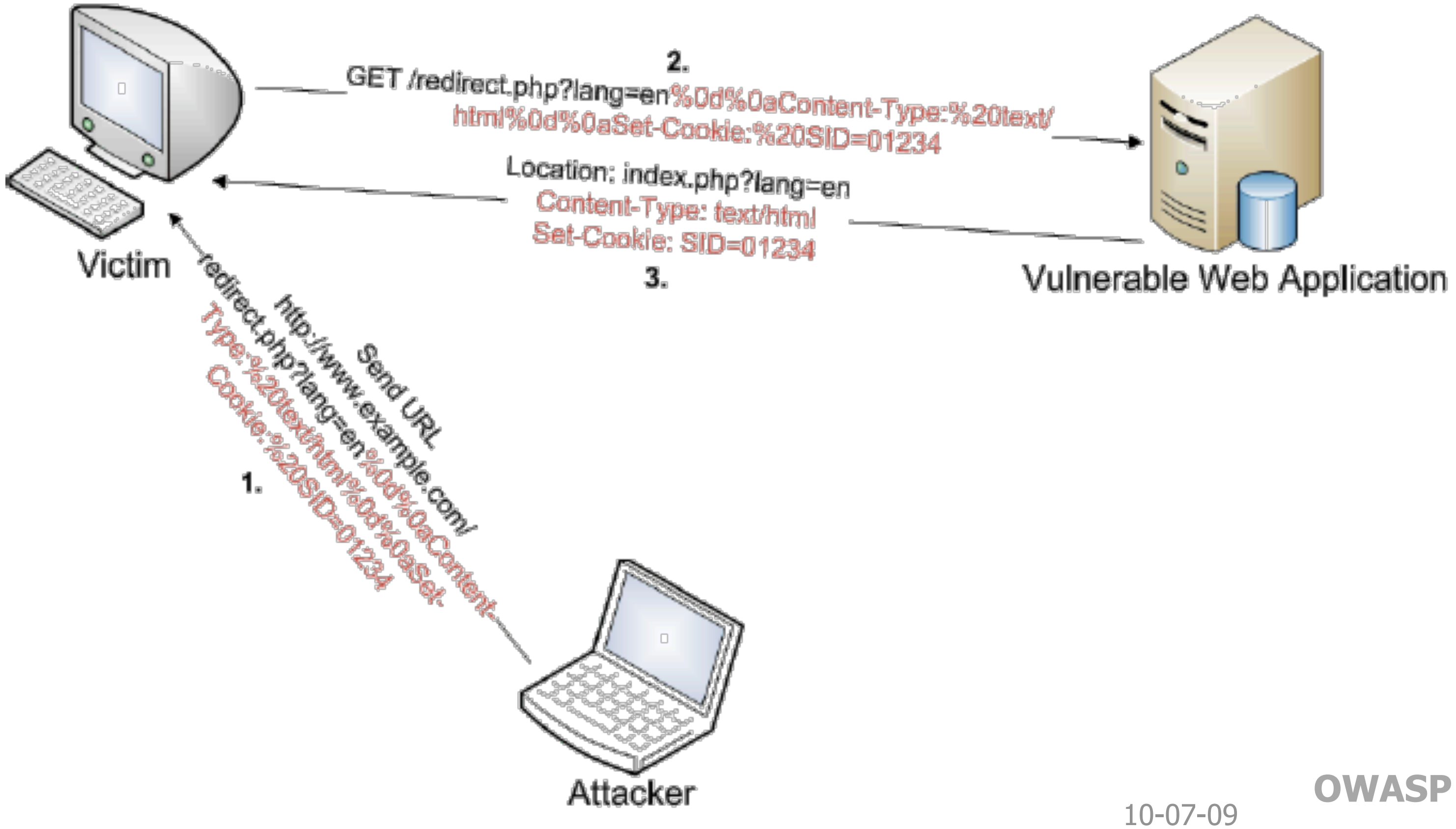
Application	Version	Cookie	URL	SID	Lang
Joomla	1.5	+	-	+	PHP
CMSmadesimple	1.6.6	+	-	+	PHP
PHPFusion	7.00.06	-	-	+	PHP
Redmine	0.9.2	+	-	-	PHP
XWiki	2.0.2.24648	+	-	-	Java
JAMWiki	0.9	+	+	+	Java
Wordpress	2.9.1	-	-	-	PHP
Novaboard	1.1.2	+	-	+	PHP
PHPBB	3.0.6	-	-	-	PHP
SimpleMachinesForum	1.1.11	-	-	-	PHP
Magento Shop	1.3.4.2	+	-	-	PHP
OSCommerce	2.2 RC 2a	+	-	-	PHP

- **Cookie: CMS accepts foisted cookies**
- **URL: CMS accepts session ID via URL parameter**
- **SID: CMS accepts arbitrary SID values**

2nd case study: HTTP header injection

- First stage attack: attacker sets cookie on client side
- Our case: user defined data taken for redirection

```
header("Location: http://localhost/index.php?
lang=".$_GET['lang']);
```



2.
GET /redirect.php?lang=en%0d%0aContent-Type:%20text/html%0d%0aSet-Cookie:%20SID=01234
Location: index.php?lang=en
Content-Type: text/html
Set-Cookie: SID=01234
3.
Victim
Vulnerable Web Application
Send URL
http://www.example.com/redirect.php?lang=en%0d%0aContent-Type:%20text/html%0d%0aSet-Cookie:%20SID=01234
1.
Attacker

2nd case study: HTTP header injection

- **Results:**
  - PHP: vulnerable in version < 4.4.2, < 5.1.2
  - J2EE: not vulnerable
  - CherryPy: vulnerable
  - Perl: partially vulnerable (name ended with colon)
  - Ruby on Rails: recently patched

Session Fixation

Case studies:

- 9 out of 12 open-source Content Management Systems (CMS) vulnerable to session fixation
- 2 out of 5 web application frameworks (at least partially) vulnerable to http header injection
- 5 out of 8 web application frameworks vulnerable to session fixation (different work)

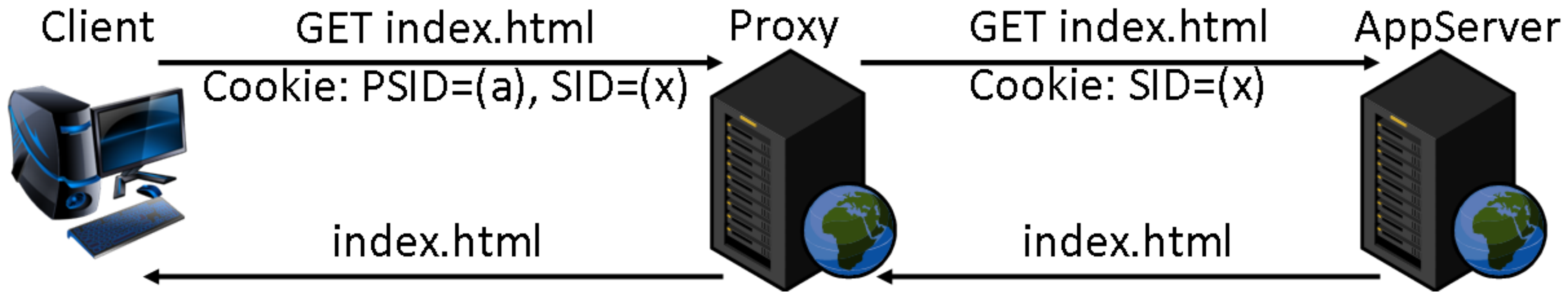
Session Fixation - Solution

- **Fixing vulnerability straightforward: renew session ID if authorization level raises**
- **However: vulnerability on server side, risk on client side (like cross site scripting case)**

→ little interest by application providers to find & fix

- **Our proposed solution:**
  - a proxy to strip off fixated session identifiers
  - Implements transparent session handling between client and proxy
  - Either on client side or on server side

Session Fixation Proxy



- **Proxy links PSID *a* and SID *x***
- **If proxy receives request with unknown SID y, all session ids are stripped off and a new session is established**
- **AppServer never sees PSID**

Conclusion

- **Public level of attention still rather low**
- **Despite given results: real world applications tested**
  - Popular web services vulnerable (2 out of 4)
  - Online Banking web sites vulnerable (2 out of 5)
  - Internet access provider (1 out of 1)
- **Risk exists though fixing is fairly easy**
- **Business partner uses proxy to buy time**
- **Proxy on server side – no big deal**
- **Proxy on client side – session ID detection not trivial**
  - Future Work!

Case Study

Framework	API	AutoRotate	Conf. Fallback	AutoDisable	
Java Server Faces	-	-	-	-	🟥
Struts 2	-	-	-	-	🟥
Spring (Security)	+	+	+	-	🟩
Zend	+	-	+	-	🟨
Cake PHP	+	+	+	-	🟩
ASP.NET	-	-	+	-	🟥
Web2py	-	-	-	+	🟥
Django	+	+	-	+	🟩

- **API: provides API to rotate SID**
- **AutoRotate: SID is rotated on every request (default configuration)**
- **Conf. Fallback: URL parameter fallback behavior configurable**
- **AutoDisable: URL parameter fallback is disabled per default**