Jan Rehwaldt

University of Tartu, Secure Programming Techniques, March 2012

*Exercise 1: Find and study two web application frameworks that offer protection mechanisms against Cross-Site Request Forging (CSRF) and compare the CSRF protection features of these frameworks against each other.*

|  |  |  |
| --- | --- | --- |
|  | Java GWT / Spring? | Ruby on Rails |
| Website |  | www.rubyonrails.org |
|  |  |  |
| Synchronizer Token Pattern  Add a generated session-unique token to requests (via previous GET request) | Yes, disabled by default (since Django 1.2) | Yes, enabled by default (since Rails 2.0, before with Plugin CSRF Killer) |
| * POST/PUT/DELETE/GET | Yes/Yes/Yes/No | Yes/Yes/Yes/No |
| * Protected request types | HTML, AJAX, others unknown | HTML, AJAX, others manually |
| * What happens on CSRF? | HTTP 403 Forbidden is send to user | Exception ActionController ::InvalidAuthenticityToken is thrown |
| * [RFC 2616](http://tools.ietf.org/html/rfc2616.html)-Compliant regarding un/safe operations | Yes | Yes |
|  |  |  |
| Double submitted cookies  Send a secure value via header and form and verify match on server | No | No |
|  |  |  |
| Non-working/weak protections: | | |
| * Checks referrer on retrieval   Referrers can easily be faked (HTTP) | Yes, only for HTTPS | No |
| * Using a Secret Cookie   Cookies are always send and thus easily available | No | No |
| * Only Accepting POST Requests   POST requests can easily be faked | No | No |
| * Multi-Step Transactions   If attacker may predict the transaction steps CSRF is still possible | No | No |

**Sources:**

* <http://archives.ryandaigle.com/articles/2007/9/24/what-s-new-in-edge-rails-better-cross-site-request-forging-prevention>
* <https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet>
* <https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)>
* <https://docs.djangoproject.com/en/dev/ref/contrib/csrf/>
* <http://guides.rubyonrails.org/security.html>