Multi-Post XSRF Web App
Exploitation, total pwnage

Adrien de Beaupré
SANS ISC Handler
Tester of pens
Certified SANS Instructor
Intru-Shun.ca Inc.
Introduction

• Web application vulnerabilities.
• What is XSRF?
• Code.
• Demo.

Founding member of the Fellowship Of Testers Of Pens #FOTOP
Doing a pentest

• I enjoy performing penetration tests, I also enjoy teaching how to do penetration testing correctly.

• Never consider the vulnerabilities in isolation, using them in combination truly demonstrates the risk and impact.

• Pillage and pivot!

• The list of things that one application was vulnerable to was quite impressive!
Vulnerabilities

• Content can be framed.
• Cross Site Scripting XSS.
• Method interchange (POST and GET).
• Cross Site Request Forgery XSRF.
• User enumeration via forgot password function.
• Administrators can disable their own account.
• ... and many others...
XSS

• Cross Site Scripting, aka XSS, script injection.
• The application had a help desk function.
• Any user could create a trouble ticket.
• Ticket will be first viewed by administrator.
• Script executes in the administrator browser.
• Administrator can perform all of the functions vulnerable to XSRF.
XSRF

• Cross-Site Request Forgery is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated (OWASP).

• This attack takes advantage of the application trusting any transaction coming from a browser where a user has an existing and valid session, or where one can be established automatically (SSO).
XSS + XSRF

• XSS and XSRF are like cousins, that get along very well, sneaky mischievous cousins.

• Persistent Admin XSS within the application guarantees the attacker that the administrator will execute the code, the code will execute the XSRF transaction(s).

• We want more than one transaction, hence the multi-post.
Requirements for XSRF

• The use has an existing or can create automatically a valid session.
• The transaction has predictable parameters.
• The victim click on a link, or types in a URL, or XSS.
• The transaction is attractive to the attacker 😊
• The attacker has knowledge of the above, or is very lucky.
XSRF Functions

• All the interesting ones in this app are as admin.
  • Add a user.
  • Change user password.
  • Put user in group.
  • Enable user.
  • Disable user.
  • Logout.
What did we want to do?

• We needed a demo to show C-levels.
• Add a new user and in administrator group.
• Lockout the super-user account.
• Logout the super-user account.
• Did the functions in the correct order.
• Each function would wait for last to complete.
• Was all in one HTML page, one click shopping!
• Would force the administrator to view a certain Rick Astley video.
Rick Astley

OK, we didn't do the last one, that would be WAY too mean.
Google-Foo

• Needed to find multi-post XSRF code.
• Found Tim Tomes @lanmaster53
  http://www.lanmaster53.com/2013/07/multi-post-csrf/
• We based our code on his. He and Ethan Robish based their code on
Pseudo code

• HTML to inject in XSS.
• 1 form per transaction.
• 1 iframe per form.
• `<body onload="runme();">`
• JavaScript:
  – The `runme` submits the forms one by one, in order, waiting for each one to complete.
Needed a vulnerable app

• Back to Google.
• I needed an app that was vulnerable for the live demo -> Omeka.
• Free open source content management application.
  http://omeka.org/
Omeka

• Is vulnerable to XSRF in admin functions.

• I could not use the application that I did the pen test for as a demonstration or public PoC.

• Omeka is easy to install and easy to exploit!
Dashboard

A new version of Omeka is available for download. Upgrade to 2.2.2

0 0 0 0 3

Thanks, Roy

Recent Items
Add a new item

Recent Collections
Add a new collection

Powered by Omeka | Documentation | Support Forums

Version 2.2 | System Information
Username*  Username must be 30 characters or fewer. Whitespace is not allowed.
foo

Display Name*  Name as it should be displayed on the site
bar

Email*  foo@intru-shun.ca

Role*  Roles describe the permissions a user has. See documentation for details.
Super
username=foo&name=bar&email=foo%40intru-shun.ca&role=super&active=0&active=1&submit=Save+Changes
Change Password

New Password*
Password must be at least 6 characters long.

Repeat New Password*

Save Password
new_password=password123&
new_password_confirm=password123&submit
=Save+Password
XSS

• The persistent XSS in Omeka is in the API keys function.

    <form action="http://omeka/admin/users/api-keys/1" method="POST">
    <input type="hidden" name="api_key_label" value='\<script\>location.href="http://omeka/xsrf-code.html"\</script\>' />
    <input type="hidden" name="update_api_keys" value="Update+API+Keys" />
    <input type="submit" value="Insert" /></form>
welcome to p0wned by XSRF!

All your base are belong to us
CATS: ALL YOUR BASE ARE BELONG TO US.
Demo showed the attack

• We used visible iframes for the demo.
• We didn’t have to, could have easily made them invisible and have the happy victim watching a video.
• No not that one, rickrolling is mean, and unprofessional in this case.
The OWASP Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

I used the Zed Attack Proxy both to identify the XSRF issue and to create the PoC code.
XSRF Defences

• Add a random nonce/token to important forms. The only real defence, performance issues.
• referer HTTP header, Origin header, Captcha, reauth, OTP. UX!!!!!
• Use multiple forms, easily bypassed by multi-post technique.
Demo

Now we cross our fingers

https://isc.sans.edu/forums/diary/18507

Note: no goats were harmed in the demo gods sacrifice!
Thanks!

Adrien de Beaupré
SANS ISC Handler
Certified SANS Instructor
Independent Consultant
Intru-Shun.ca Inc.
#BSidesOttawa
@adriendb
adrien@intru-shun.ca