Identity Management Testing

Instructor: C. Pu (Ph.D., Assistant Professor)

Lecture 04

puc@marshall.edu

Test Role Definitions: Summary

- It is common in modern enterprises to define system roles to manage users and authorization to system resources.
- In most system implementations, it is expected that at least two roles exist
  - ***Administrators***
    - representing a role that permits access to privileged and sensitive functionality and information.
  - ***Regular users***
    - representing a role that permits access to regular business functionality and information.
- Well developed roles should align with business processes which are supported by the application.

Test Role Definitions: WordPress

Capability	Super Admin	Administrator	Editor	Author	Contributor	Subscriber
install_plugins	Y	Y (single site)				
install_themes	Y	Y (single site)				
list_users	Y	Y				
manage_options	Y	Y				
promote_users	Y	Y				
remove_users	Y	Y				
switch_themes	Y	Y				
update_core	Y	Y (single site)				
update_plugins	Y	Y (single site)				
update_themes	Y	Y (single site)				
edit_dashboard	Y	Y				
customize	Y	Y				
delete_site	Y	Y				

Capability	Super Admin	Administrator	Editor	Author	Contributor	Subscriber
moderate_comments	Y	Y	Y			
manage_categories	Y	Y	Y			
manage_links	Y	Y	Y			
edit_others_posts	Y	Y	Y			
edit_pages	Y	Y	Y			

- https://wordpress.org/support/article/roles-and-capabilities/#roles

Test Role Definitions: Test Objectives

- Validate the system roles defined within the application.
  - Sufficiently define and separate each system and business role to manage appropriate access to system functionality and information.

Test Role Definitions: How to Test

- Either with or without the help of the system developers or administrators, develop a ***role versus permission matrix***.
- The matrix should enumerate ***all the roles*** that can be provisioned and explore the ***permissions*** that are allowed to be applied to the objects including any ***constraints***.
- If a matrix is provided with the application, it should be ***validated*** by the tester.
- If it doesn't exist, the tester should generate it and determine whether the matrix satisfies the desired access policy for the application.

Test Role Definitions: Example

Role	Permission	Object	Constraints
Administrator	Read	Customer records	
Manager	Read	Customer records	Only records related to business unit
RoStaff	Read	Customer records	Only records associated with customers assigned by Manager
Customer	Read	Customer records	Only own record

Test Role Definitions: Example

- http://testfire.net/
- Admin
  - Username: admin
  - Password: admin
- Regular User
  - Username: jsmith
  - Password: Demo1234

Test Role Definitions: How to Test

- While the most thorough and accurate approach to completing this test is to conduct it manually.

- Spidering tools are also useful.
  - Log on with each role in turn and spider the application.
  - Tool: skipfish (Kali Linux)

Test User Registration Process: Summary

- Some websites offer a user registration process that automates (or semi-automates) the provisioning of system access to users.
- The identity requirements for access vary from *positive identification* to none at all, depending on the security requirements of the system.
- Many public applications completely automate the registration and provisioning process because the size of the user base makes it impossible to manage manually.
- However, many corporate applications will provision users manually, so this test case may not apply.

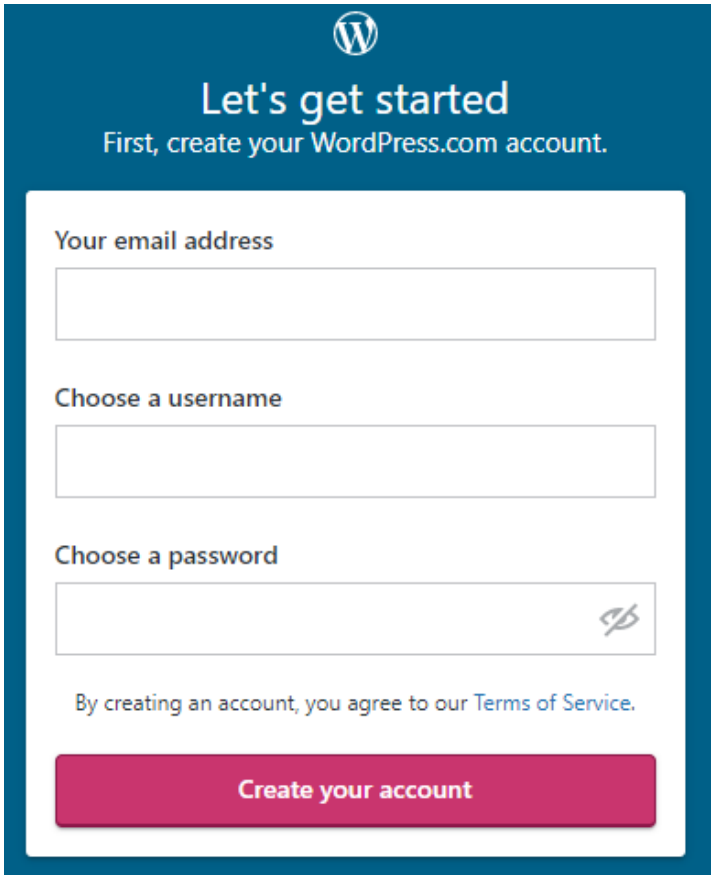
Test User Registration Process: Test Objectives

- Verify that the identity requirements for user registration are aligned with business and security requirements.
- Validate the registration process.

Test User Registration Process: How to Test

- Verify that the identity requirements for user registration are aligned with business and security requirements:
  - Can anyone register for access?
  - Are registrations vetted by a human prior to provisioning, or are they automatically granted if the criteria are met?
  - Can the same person or identity register multiple times?
  - Can users register for different roles or permissions?
  - What proof of identity is required for a registration to be successful?
  - Are registered identities verified?
- Validate the registration process:
  - Can identity information be easily forged or faked?
  - Can the exchange of identity information be manipulated during registration?

Test User Registration Process: Example - WordPress



Test User Registration Process: Example - Google

Google

Create your Google Account

First name | Last name

Username @gmail.com

You can use letters, numbers & periods

Use my current email address instead

Password | Confirm

Use 8 or more characters with a mix of letters, numbers & symbols

Sign in instead | Next

One account. All of Google working for you.

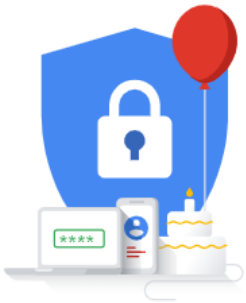
Google

Verify your phone number

For your security, Google wants to make sure it's really you. Google will send a text message with a 6-digit verification code. *Standard rates apply*

Phone number

Back | Next



Your personal info is private & safe

Testing for Account Enumeration and Guessable User Account: Summary

- The scope of this test is to verify if it is possible to collect a set of ***valid usernames*** by *interacting with the authentication mechanism* of the application.
- This test will be useful for brute force testing, in which the tester verifies if, given a valid username, it is possible to find the corresponding password.
- Often, web applications reveal when a username exists on system, either as a consequence of mis-configuration or as a design decision.
- For example, sometimes, when we submit wrong credentials, we receive a message that states that either the username is present on the system or the provided password is wrong.
- The information obtained can be used by an attacker to gain a list of users on system.
- This information can be used to attack the web application, for example, through a brute force or default username and password attack.

Testing for Account Enumeration and Guessable User Account: Summary

- The tester should interact with the authentication mechanism of the application to understand if sending particular requests causes the application to answer in different manners.
- This issue exists because the information released from web application or web server when the user provide a valid username is different than when they use an invalid one.
- In some cases, a message is received that reveals if the provided credentials are wrong because an invalid username or an invalid password was used.
- Sometimes, testers can enumerate the existing users by sending a username and an empty password.

Testing for Account Enumeration and Guessable User Account: How to Test

- Testing for Valid user/right password
  - Record the server answer when you submit a valid user ID and valid password.

- Result Expected:
  - Using WebScarab, notice the information retrieved from this successful authentication (HTTP 200 Response, length of the response).

Testing for Account Enumeration and Guessable User Account: How to Test

- Testing for valid user with wrong password
  - The tester should try to insert a valid user ID and a wrong password and record the error message generated by the application.

- Result Expected:
  - The browser should display a message similar to the following one:

Testing for Account Enumeration and Guessable User Account: How to Test

- Against any message that reveals the existence of user, for instance, message similar to

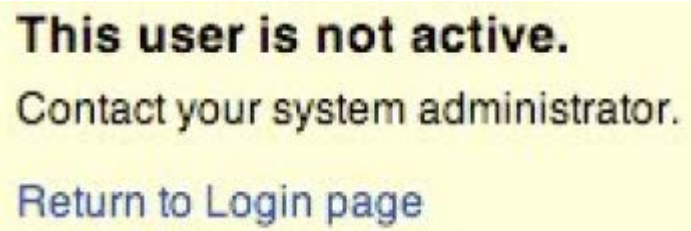
> Login for User foo: invalid password

- Using WebScarab, notice the information retrieved from this unsuccessful authentication attempt.

Testing for Account Enumeration and Guessable User Account: How to Test

- Testing for a nonexistent username
  - The tester should try to insert an invalid user ID and a wrong password and record the server answer (the tester should be confident that the username is not valid in the application).
  - Record the error message and the server answer.

- Result Expected:
  - If the tester enters a nonexistent user ID, they can receive a message similar to:



Testing for Account Enumeration and Guessable User Account: How to Test

- Generally the application should ***respond with the same error message and length to the different incorrect requests***.
- If the responses are not the same, the tester should investigate and find out the key that creates a difference between the two responses.
- For example:

  - Client request: Valid user/wrong password -->
    Server answer:'The password is not correct'
  - Client request: Wrong user/wrong password -->
    Server answer:'User not recognized'

  - The above responses let the client understand that for the first request they have a valid user name.
  - So they can interact with the application requesting a set of possible user IDs and observing the answer.

Testing for Weak or Unenforced Username Policy: Summary

- User account names are often highly structured and valid account names can easily be guessed
  - e.g.
    - Joe Bloggs account name is jbloggs
    - Fred Nurks account name is fnurks

Testing for Weak or Unenforced Username Policy: Test Objectives

- Determine whether a consistent account name structure renders the application vulnerable to account enumeration.
- Determine whether the application's error messages permit account enumeration.

Testing for Weak or Unenforced Username Policy: How to Test

- Determine the structure of account names.
- Evaluate the application's response to valid and invalid account names.
- Use different responses to valid and invalid account names to enumerate valid account names.
- Use account name dictionaries to enumerate valid account names.
  - 403,355 username of/at US
    - https://github.com/duyetdev/bruteforce-database