Username Enumeration Using Selenium

Raaghavv Devgon
Northeastern University
Boston, MA, USA
devgon.r@northeastern.edu

Abhishek Ningala
Northeastern University
Boston, MA, USA
ningala.a@northeastern.edu

Jugal Joshi
Northeastern University
Boston, MA, USA
joshi.ju@northeastern.edu

***Abstract*—The objective of this project is to perform username enumeration on top 500 websites from the Majestic Million database. We have tried to curate a tool which is generic enough to be used to perform the attack based on the error messages shown on the website on multiple websites. For this project we analyzed 500 websites, each website was unique in their architecture which made the task at hand challenging, but we have maneuvered our way out of the problem and have been able to perform username enumeration on them. The tool has been created using python and selenium, and we were able to test and perform the enumeration successfully on 50 websites.**

Source Code: GitHub

I. Introduction

User enumeration can be defined when a malicious actor is allowed to use brute-force techniques to either guess or confirm valid users in a system. Often web applications reveal when a username exists on the system, either as a consequence of misconfiguration 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. We used this error message technique as the foundation of our tool.

II. Challenges at hand

Each website is unique, each web developer has their own way of developing a website, therefore, if a website has a login page, it will have a different username id, name, and certain attribute, they can also be similar in some cases, but as we tried to evaluate more websites, we realized we would have to tackle this problem, as it is different for each website. Similar could be said for the error messages, each website has its own way of stating - ”Incorrect email address. Please enter information again”. Some websites also require an email address to login, some require usernames. Therefore, we decided to manually visit each of the websites and create individual scripts for the websites so that we could build a database for the error messages. There are also different pages on a website via which a website could be susceptible to this attack, namely: sign up , login and forgot password, we tested on all three possible pages to check if they were indeed vulnerable.

III. Implementation

We created our tool using Python 3 and Selenium. The websites were first tested using their individual script, then we modified our generalized script to check if it worked in a generalized landscape. As we are dealing with multiple ways a website can take input, we leveraged the attribute - XPath of a website, we realized the XPath did not differ a lot for websites and we were able to enumerate 50 websites successfully while keeping track of 5-8 XPath attributes. This helped us to send the input of a possible username or email precisely in the respective input fields. As the enumerator uses IP address, we also added the functionality of TOR as a service, which is used as a proxy to help the user not to get their IP blocked while using the application. We have created two functions that can be used based on the flag (P or NP) where P denotes to check if the error message depicts the username/email to exist, and the latter is related to the error messages which are related to the fact the current username does not exist. Each of the functions have their message database, the HTML text on the websites upon entering a username is parsed using Beautiful soup and then the content is cleared of all special characters and empty spaces using regex. This approach was adopted as some of the websites like Adobe, start to insert special characters and empty spaces in the error messages on repeated tries. Then we checked each of the text on the HTML against the message database we had created.

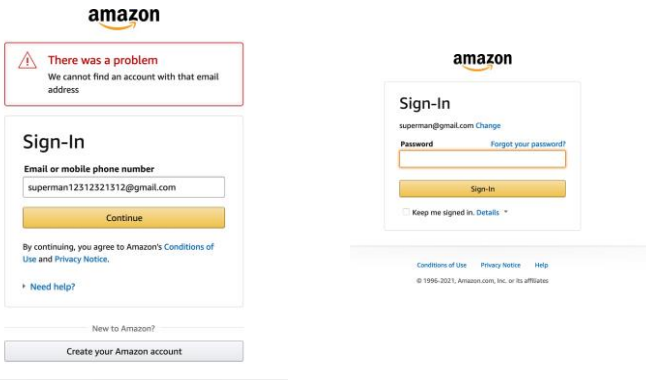

Fig. 1. Enumerating when the username doesn’t exist, by this method you can see if superman@gmail.com exists, it will not show the error message shown in the first figure

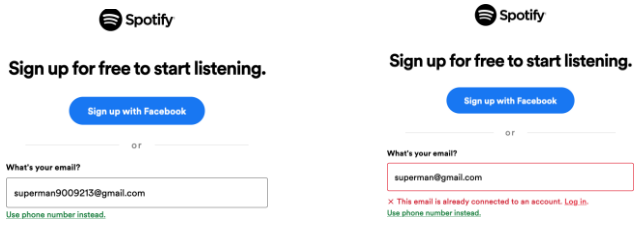

Fig. 2. Enumerating when the username exists, by this method you can see if superman@gmail.com exists, it will show the error message shown in the second figure.

IV. Analysis of the tool

A. Advantages of using the tool.

1) We found that the tools which are available online for performing username enumeration were limited i.e they would only work for one particular website, for example WPScan, whereas the script we created could be applied to more than one website.
2) The tool, can be easily extended to other websites, we are currently enumerating over the message database generated for 50 odd websites, if the error message on the website you are trying to enumerate doesn't exist in the two message files provided, then you would probably have to insert the message in the database.
3) The user can even supply multiple websites to enumerate and the tool will still work fine.

B. Disadvantages of using the tool.

1) Currently the tool sleeps for a given amount of time between processing and enumerating, which might cause some delay to get the messages.
2) The time complexity is O(n) as the message found on the target website will be checked against each message in the database.
3) Some of the websites know they are accessed by selenium, therefore they would throw a CAPTCHA after certain number of attempts, thwarting any further attempts to enumerate by the user.
4) The tool works on 50 of the 100 websites we found to be vulnerable.
5) One of the issues of the tool is that it needs a URL to enumerate, so if the login page, sign up page don't exist on a separate link, it would be difficult for the tool to find the login button on the website, for example: Twitch.

V. Problems faced

We faced a lot of challenges during the implementation of this project, namely because of the uniqueness of each website and the attempt to generalize them for our tool:

1) Some websites did not allow us to access via TOR, this thwarted our tool, unable to enumerate.
2) There were also situations where the TOR IP was of a different country, hence leading us to a website which was not in English (specifically happened in Google), which made it difficult to match the error messages.
3) Some websites captchas, which didn't allow us to send the respective responses.
4) Exponential Backoffs.
5) Some sites display "Accept cookie" messages, at times, the enumeration cannot proceed until one interacts with this message. The only workaround is to stop and rerun the program if the program remains idle for a long time. We noticed this happens only for a few sites, it worked properly most times so this was not deemed a major problem.
6) We were also at times, unable to locate element to send our input to, for example, Instagram.

VI. Findings

We created two functions for our tool as discussed above, one which deals with messages where you can enumerate when the username doesn't exist, and the messages where you can enumerate with the messages when the username exists.

1) We found that approximately 95% of the vulnerable websites could be enumerated with just the sign up or registration pages
2) We found that some adult websites and dating websites also suffered from this problem. This can potentially have adverse effects on the users as it violates their privacy. Websites like Redtube and OkCupid were found to be vulnerable.
3) Some of the websites like Spotify, didn't require the user to fill up the entire registration form, if they did, it would've thwarted our attempt to enumerate as they also required CAPTCHA, but currently as of writing this paper, if you enter an email address and press enter on their registration page, you will see a message, irrespective of the fact that you haven't filled the form entirely.
4) Even FAANG websites (Facebook, Amazon, Apple, Netflix, Google) were vulnerable to this form of username enumeration.

VII. Future work

There are significant ways this tool can be improved. Currently how the tool checks if the message exists in the database, is by comparing each of the messages with the text on the website. We could leverage a sub-string matching algorithm to compute the match and reduce the computational overhead. We can also have a set of specific keywords and use that for enumeration. Another interesting approach could be to use the semantic analysis from Natural Language Processing to find the generic messages and what they mean using the text. Of course, we could also try to incorporate other ways to perform username enumeration using timed-based response etc, this would improve the tool and also bolster the confidence in the results.

VIII. Contribution

Each member of the team contributed to the project. All members did a great job of enumerating each website and creating individual scripts for the websites. Abhishek did a lot of manual work and helped figure out the XPATH solution. Jugal helped in selenium and visualizing the generalized Script. Raaghavv incorporated the individual scripts into a generalized version. All members helped to test the 50 websites multiple times. The script was moved into the Docker environment by Raaghavv and the report have been composed by all members of the team.

IX. Summary

To conclude, we successfully developed a tool using selenium which used the error message based technique to perform username enumeration. We performed the attack on 100 websites that were deemed vulnerable and our tool helped enumerate on 50 of those websites. Websites such as Spotify, Netflix, Google, Yahoo, Amazon, Adobe can be enumerated using our tool. We know the ways our tool can be improved, and we hope to improve it in the future to create a more robust and efficient enumerator, which can be used on multiple websites.

References

[1] OWASP: owasp.org

[2] Laverty, Patrick. “User Enumeration Explained: Techniques and Prevention Tips: Rapid7 Blog.” Rapid7, Rapid7 Blog, 19 Dec. 2019, www.rapid7.com/blog/post/2017/06/15/about-user-enumeration/.

[3] Jake. “5 Ways to Enumerate Usernames in Web Applications.” Laconic Wolf, 23 Aug. 2018, laconicwolf.com/2018/08/22/5-ways-to-enumerate-usernames-in-web-applications/.

[4] Nanu. “What Is Username Enumeration Vulnerability ?” Free Learning Tech, 16 Dec. 2020, freelearningtech.in/what-is-username-enumeration-vulnerability/.

[5] Mohamed, Ahmed Elhady. “WordPress Username Enumeration Techniques and How to Fix Them.” Medium, Medium, 21 Nov.2017

[6] “Getting Started.” Getting Started :: Documentation for Selenium,

[7] “Tor Browser User Manual.” Tor Browser User Manual — Tor Project — Tor Browser Manual, tb-manual.torproject.org/.