CS 467
Spring 2021

Website Security Research Project: Mid-Point Check

Website Security Research Team Members

Reed Hardin
Michael Volz
Isfandyar Rabbani
Patrick Dougan

Submission Contents

Included in this Mid-Point Check is:

- This Mid-Point Report (PDF)
- GitHub Repo:
  https://github.com/PatrickDougan/Website-Security-Research-Project
- Working "target" site URLs (listed below)
- A zip file containing all our currently hosted code

Project Status

As of the writing of this project report, our team has created a series of "target" web sites to use as demonstration vehicles for various attacks. One set is hosted on AWS, using (somebody fill in the blanks here please). Another set of pages is hosted independently, and pointing at mySQL databases hosted on AWS. Each offers distinct features, such as login and authentication, dynamic content, and a search function in development. We will later combine these elements into one full scale website, with various attack surfaces and ways to penetrate them.

Test site #1, hosted on AWS EC2 Node server and local MySQL database:
http://ec2-54-202-128-184.us-west-2.compute.amazonaws.com:21001/

Description - A basic login and signup site. The signup page will create an account record in a local to that machine MySQL database. These credentials can then be used to “login” to the site, showing a landing page that confirms the user has logged in. We are researching various ways to keep this portion of the website pervious to attacks such as SQL injection, cross-site scripting, and broken authentication. So far, we have used a tool that functions to find SQL injection holes and populating junk data in our database. Even though there is little code to exploit currently, the tool has successfully created a lot of junk data in our databases, demonstrating how the data could be even further manipulated. We will be finding out just how very soon!

Test site #2, hosted independently running NodeJS v10.24.1:
http://volzpdx.com/test_site/
This is a fairly straightforward NodeJS database front-end using handlebars and express to interact with a mySQL database. The inspiration here is a tool for a middle-school band teacher (various bands with different directors, students, music, and instruments - each with data). This site has already been used to test SQL injection attacks.

Test site #3, hosted independently running NodeJS v14.16.1:
http://volzpdx.com/test_site2/
Very similar to test site #2 but a different database, with slightly different settings (which will likely prove useful in demonstrating SQL injection attack mitigation)

Test site #4, hosted independently running NodeJS v10.24.1:
http://volzpdx.com/test_site3/
(still in progress - this is intended to use passport authentication)

Our next step is for each team member to dive into an individual vulnerability, understand it, write and exploit for it, and publish a set of instructions for how to replicate the attack. Each team member is responsible for two vulnerabilities, thus eight total should be demonstrable by the end of the project. These will be published in a user-friendly access point via our GitHub repository page.

User Instructions

In its current state, a user would have to navigate manually to each of the links

for our websites developed and hosted for testing purposes. Once there, the user can interact with separate elements of our web app to experience the progress of each feature. The features are already into testing phase, using tools that are meant exploit common vulnerabilities, so a TA would be welcome to conduct their own penetration testing, and report their results to us.

User instructions for the final version of our sites will be accessed via our GitHub repository (linked above) listed out via the README.md. If we determine that a more elaborate demonstration mechanism is needed, we will create that and link to it from that page.

For the cross site scripting vulnerable website, you will be able to add a name, email, and comment. The comment field is susceptible to stored XSS attacks. If you tag the comment in <script></script> you will be able to have code executed on any client that visits the webpage. Total character length of the comment must be 255 characters or less.Currently not implemented on the main page.

For the sensitive data exposure, a demonstration will be provided such that a non-indexed web page (e.g., an office phone-number list, or company org. chart) exists in one site where it is insecure and openly accessible (even if it's not indexed or directly linked to) while on an otherwise identical site, it is secure behind a user-based password authentication protocol.

For broken access control, a demonstration might include a manipulation of a JSON Web Token (or similar) can elevate a user's privileges past what is authorized, allow for abusive entry, manipulation, or saving of sensitive data.

References:

Code references:
https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html
https://sumantmishra.medium.com/how-to-deploy-node-js-app-on-aws-with-github-db99758294f1
https://github.com/achari/nodejs_mysql_example/blob/master/controllers/registerController.js
https://medium.com/@as.vignesh/install-mysql-on-amazon-linux-in-aws-ec2-e31a842f7ae9

https://codeshack.io/basic-login-system-nodejs-express-mysql/
https://dvwa.co.uk/
https://www.hackthissite.org/pages/index/index.php

Analytical references:
https://www.ptsecurity.com/ww-en/analytics/web-application-attacks-2019/#id11
https://www.sophos.com/en-us/dede/medialibrary/pdfs/other/sophossecuritythreatreport2013.pdf
https://softwareontheroad.com/nodejs-jwt-authentication-oauth/