Assignment 2

Chetan Sahrudhai Kimidi - ckimidi

***Abstract*—Completing XPATH Injection, from WebGoat 7.1 Injection Flaws**

I. Introduction

The XPATH query language is used for data selection and querying in XML-based databases. When compared to SQL-injection, XPATH-based injection is pro-attacker. We can design endless XPATH-based injections that will function on any database, which use XPATH. SQL injections are limited to the particular variant of SQL that is being utilized by the database. Furthermore, XPATH has no Access Control Lists (ACLs). SQL ACLs define which users may do what actions on which data. Our injections may access whatever data in the XML that we choose because this control isn't implemented in XPATH, which means easier sensitive data access.
In this assignment, I successfully accessed all the other employees data, from a login form.

II. My Approach

I figured out that when I try to login as Mike, the internal XPATH query would look somewhat like -
string(//Employee[username/text()='Mike' ...)

Now, I employed the "appending a universally true case" approach, as always. So, I appended a couple of true cases, using the OR operator instead of the usually used AND operator. The query was **'Mike' or 1=1 or 'a'='a**, which would internally look like,

string(//Employee[username/text()='Mike' or 1 = 1 and password/text()='test123' or 'a' = 'a']/account/text())

So, this is not only injecting a true case into the username, to show all the usernames in the database, but is also injecting another true case into the password input, to allow access, all thanks to the OR operator. I finally accessed all the employees data, as seen in Fig. 1.

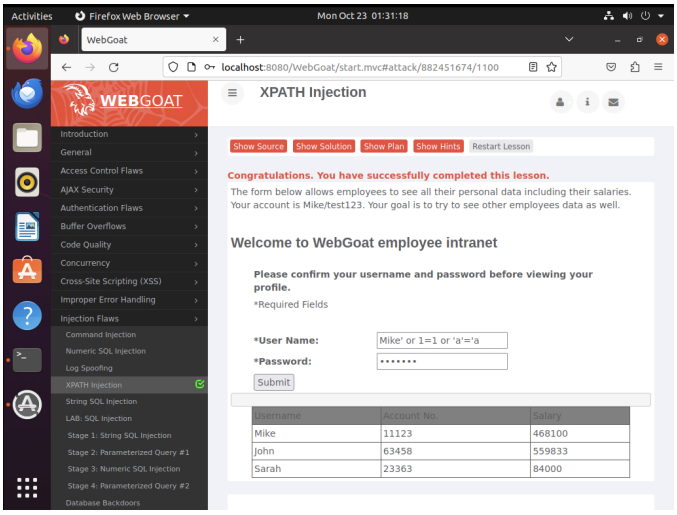

Fig. 1. Sarah is being underpaid!

III. Conclusion

This concludes the second assignment, and I was successfully able to pull all of the employees data from the XML-based database.

References

[1] XPATH Injection - OWASP
[2] A beginner's guide to XPATH Injection