FINAL REPORT

WEB BROWSER ADDON FOR DETECTING SQL AND XSS VULNERABILITIES

COLIN BROPHY
CYBERCRIME & IT SECURITY
30/04/21

Contents

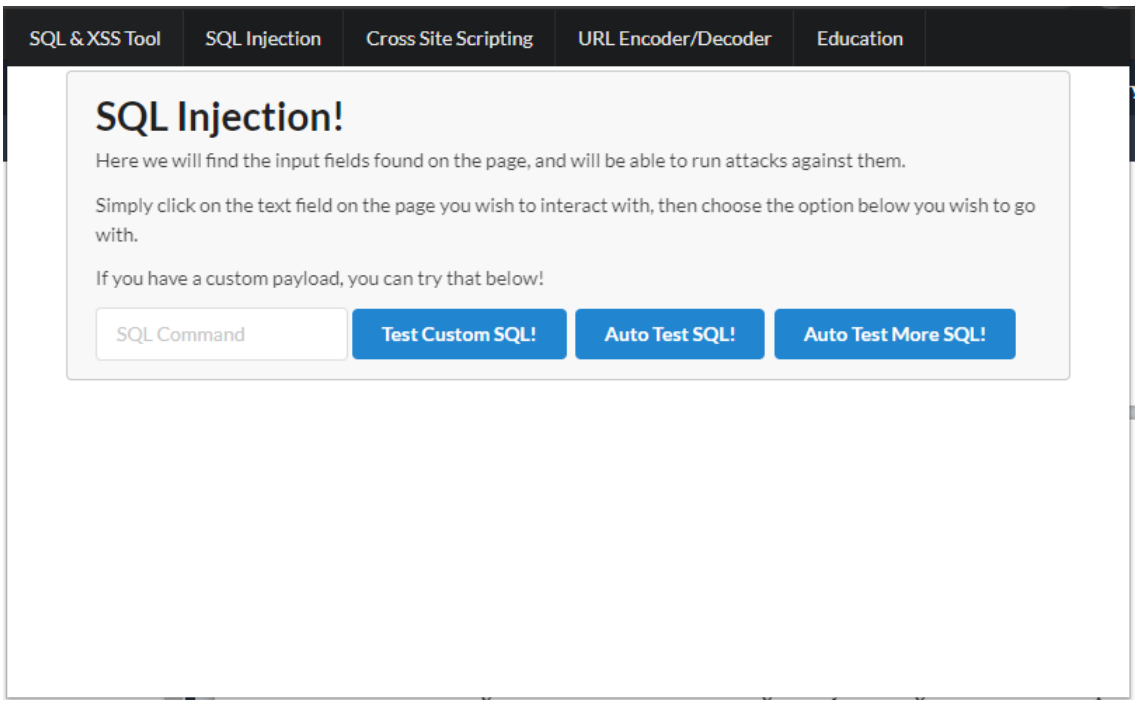
INTRODUCTION

The following document will outline my project to date, the challenges I faced and overcome, and the technologies used/ new skills learned along the way. I will begin with a brief description of what the project is, I'll talk about how my final project matches up with my aspirations from the start, and finally I will talk about what I could have done differently if I were to attempt this project again.

DESCRIPTION OF SUBMITTED PROJECT

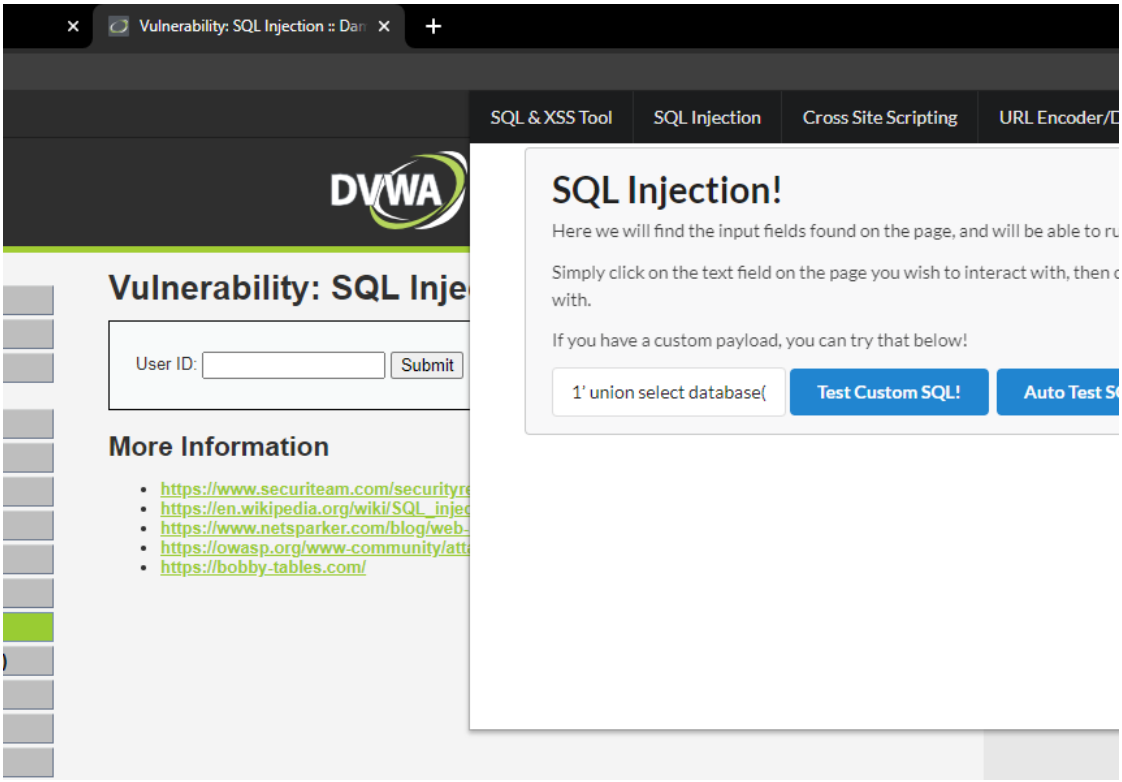
My final project, while not complete as per my initial idea, has turned out to be a good proof of concept. The idea was for a web browser extension that would test the current web page for Cross Site Scripting and SQL Injection vulnerabilities. It was to be a must have extension for Web App Developers, Pen-testers and college students studying IT. What I ended up with, given the circumstances, is a proof of concept extension which can inject XSS and SQL payloads into a web page and run them for you.

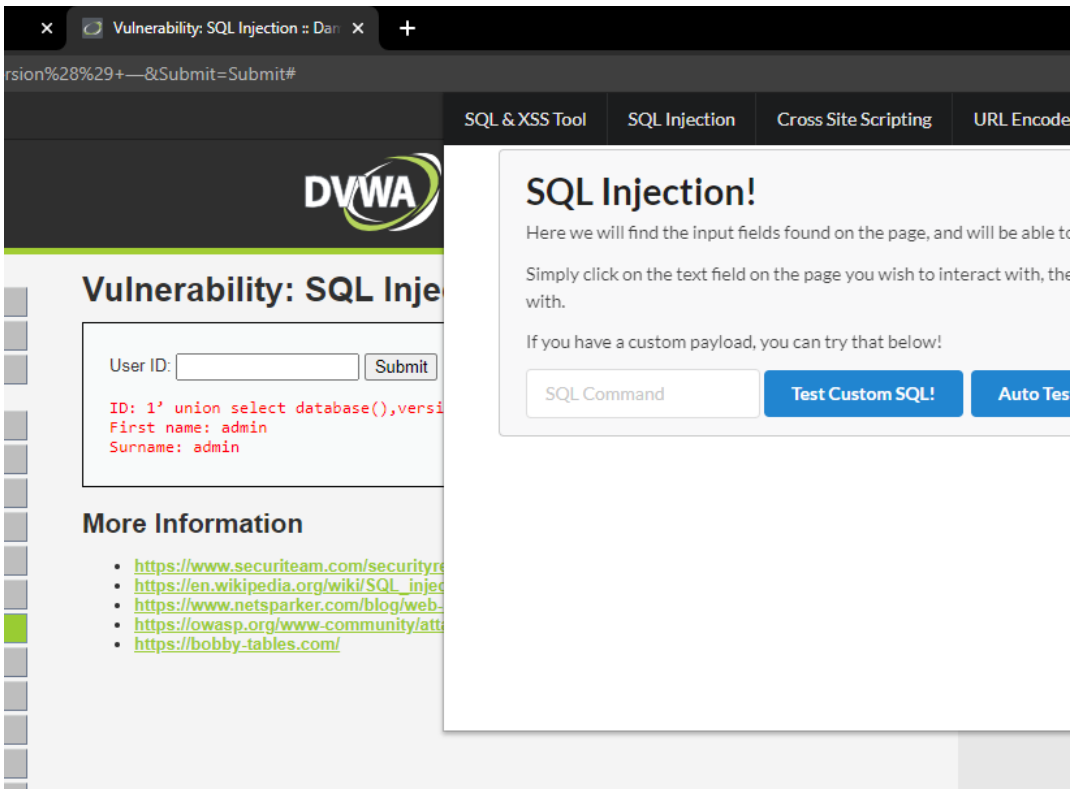
SQL INJECTION



The above photo shows SQL Injection tab of the end result. The tab has an input field where the user can input custom payloads should they have any specific ones they would like to try. Alongside that are two buttons which automatically test set payloads that are known to work with the DVWA (Damn Vulnerable Web App). I decided, given the time constraints, to make the extension work with the DVWA as a proof of concept for how it would work in the field.

There are two separate buttons as the automatic running of multiple payloads is not something I could get done in time. When either of the buttons is pressed, the payload is entered into the current active input field on the page and is ran, to which the DVWA gives back a result, as below:





If I had more time and a greater understanding of JavaScript/ Chrome web app development, I would implement a single button, which runs a series of tests that are held in an array in the code, and it would compile a report of which payloads were successful and which weren't.

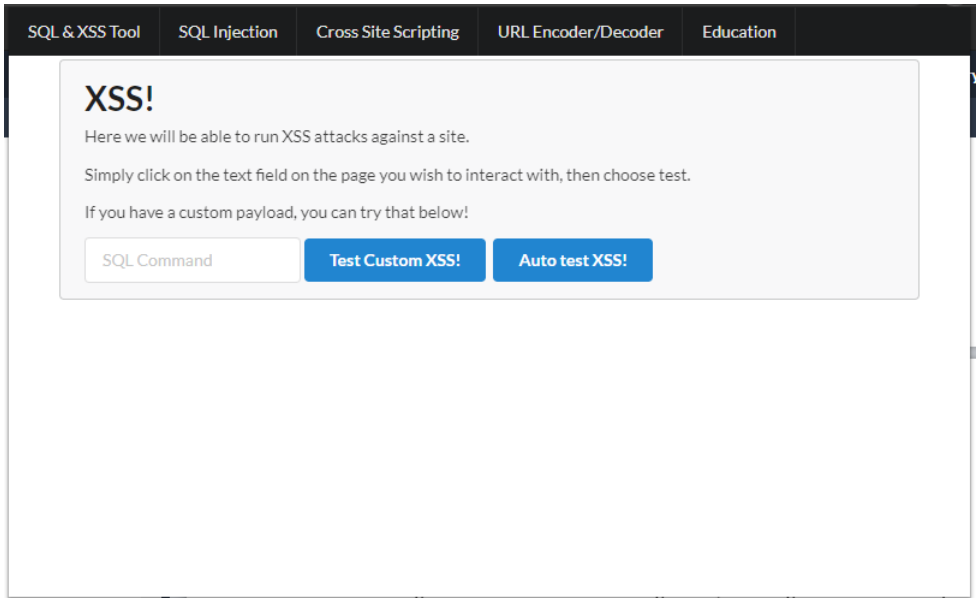
```
var SQLPayloads = [
    "1' union select database(),version() -",
    "%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #"
];

function testCustomSql() {
var testValue = document.getElementById('SQLToTest').value;
  chrome.tabs.executeScript(null, {
    code: 'var ToTest = ' + JSON.stringify(testValue)
}, function() {
    chrome.tabs.executeScript(null, {
        file: 'html/testForSql.js'});
  });
}
document.getElementById('testCustomSql').addEventListener('click', testCustomSql);

function testSQLOne() {
var testValue = SQLPayloads[0];
  chrome.tabs.executeScript(null, {
    code: 'var ToTest = ' + JSON.stringify(testValue)
}, function() {
    chrome.tabs.executeScript(null, {
        file: 'html/testForSql.js'});
  });
}
document.getElementById('testSqlOne').addEventListener('click', testSQLOne);
```

At the top of the code seen above, we can see the array with the two payloads I have in place. This would have been expanded upon if I had gotten the automation working, but there are only two, for the two buttons mentioned earlier. The function below the array (testCustomSQL) takes in the value in the input field of the extension and runs that, the one below it (testSQLOne) runs the payload in the first slot [0] of the array.

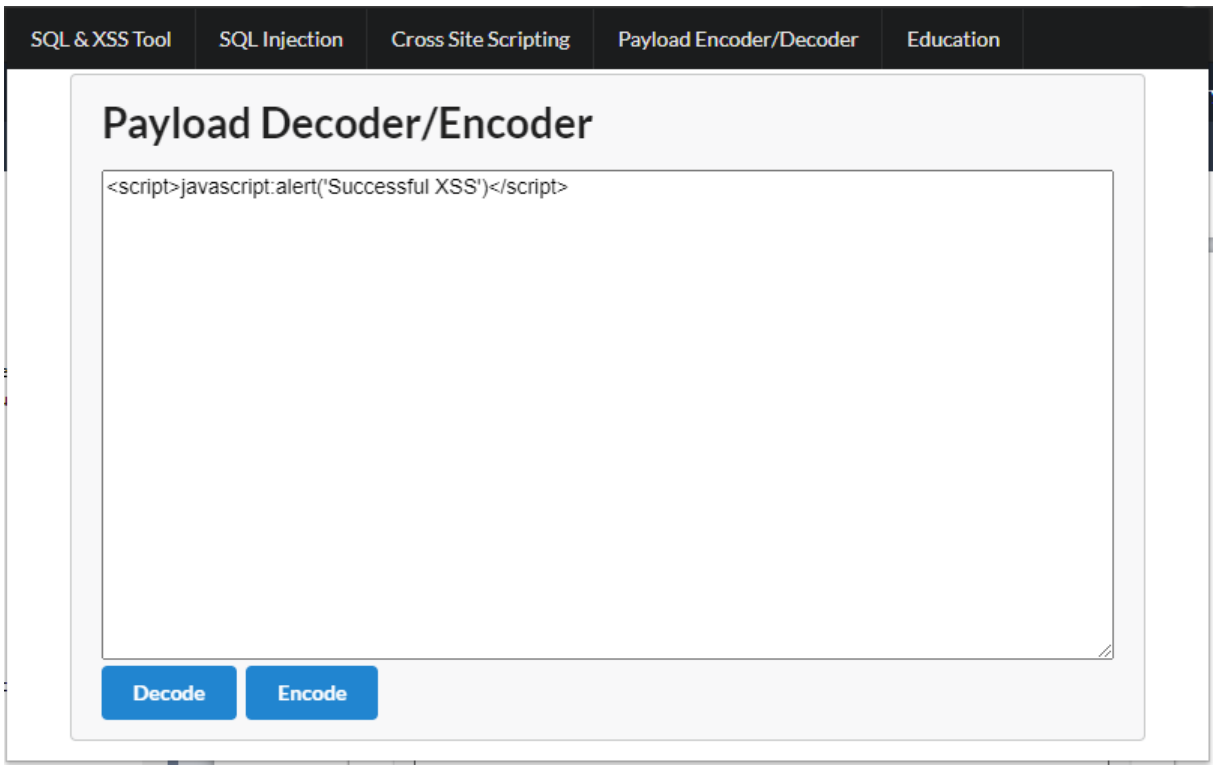
CROSS SITE SCRIPTING

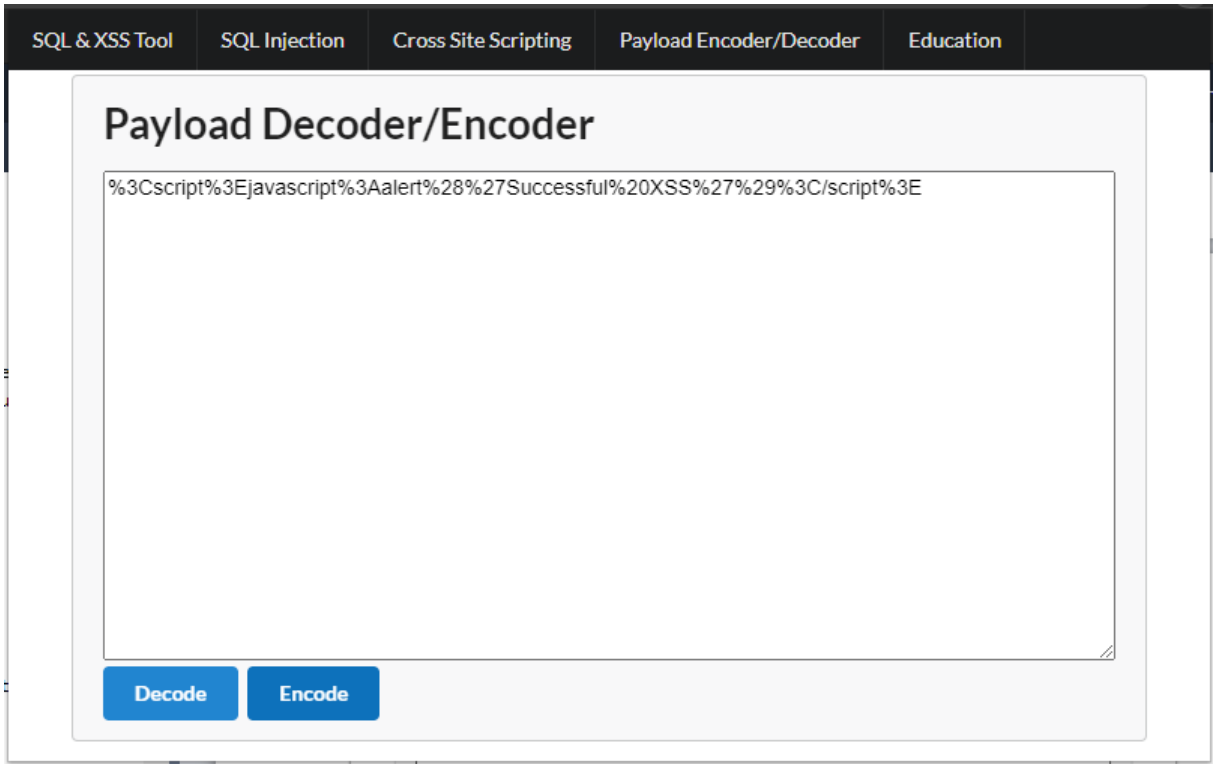


The XSS Tab of the extension works in a similar fashion. Seen above, we have the same input field, to test custom payloads, along with an auto test button which tests a payload known to work with the DVWA. The idea with the XSS tab was also to automate multiple tests and report back to the user.

PAYLOAD ENCODER

The third tab of the extension is a tool for the end user to encode/ decode payloads such as XSS payloads, or URL's. The user simply enters in the payload or the URL they wish to encode/ decode and clicks the relevant button. As per the example below, any special characters get changed to their encoded form, or from the encoded form back to the original ASCII form.


SQL & XSS Tool
SQL Injection
Cross Site Scripting
Payload Encoder/Decoder
Education
Payload Decoder/Encoder
<script>javascript:alert('Successful XSS')</script>
Decode
Encode


SQL & XSS Tool
SQL Injection
Cross Site Scripting
Payload Encoder/Decoder
Education
Payload Decoder/Encoder
%3Cscript%3Ejavascript%3Aalert%28%27Successful%20XSS%27%29%3C/script%3E
Decode
Encode

DESCRIPTION OF CONFORMANCE TO SPECIFICATION

SPECIFICATION

As my development of the extension continued, I realised what wouldn't work, what could be done better, and what was just easier given the time-frame we had. Originally, I wanted the application to work for Chrome and Firefox, but there are a few reasons I stuck to just Chrome:

- Chrome has a larger user base, more documentation for creating extensions.
- Learning Chrome extension development was taking up enough time on its own.
- The given time-frame did not allow creating, essentially, two extensions.

The idea at the beginning was for the extension to be able to run on any web page, list all the input fields on the current web page and allow the user to target each one individually. A series of tests would then be run and a report would be given back to the user as to what payloads worked and what didn't. The extension would give the user the ability to test SQL Injection and Cross Site Scripting vulnerabilities separately. Added bonuses for the end user would include a payload encoder for trickier sites that weren't vulnerable to basic attacks. An education tab would allow users access to a wealth of info regarding the type of testing they wish to do.

The final product differs from this in that it doesn't fully accomplish the automated testing. The addon does include a working payload encoder, which doubles as a URL/ Payload decoder also. ASCII text is entered into the text box and the Hex equivalent, ready for use in a URL attack are presented back to the user.

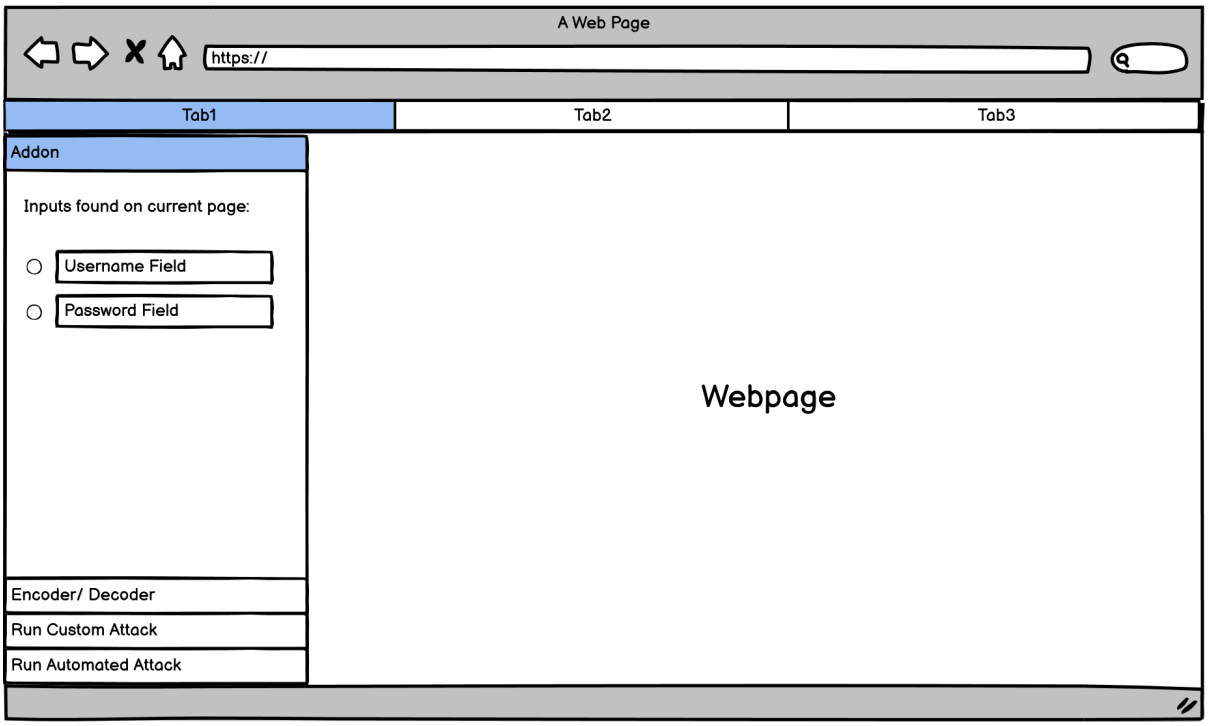
The education tab gives basic info on what each type of testing is all about, and includes an option to visit the OWASP site for more info on each section.

The testing section is where things go unfinished. What the extension, in its current state, doesn't do is automate the running of a given set of payloads and report back to the user. The extension also doesn't return back all the available input fields found on the page, instead a simpler solution was created. The extension works against the current active input field. The user simply clicks into the field they wish to attack, then opens the extension. The reasoning for this was that it was proving difficult to pull all input fields from any website I visited as they all use different ID's, Names, and input types, not all of which could be accounted for. Also, filling up the extension with all available input fields began to crowd up the page and

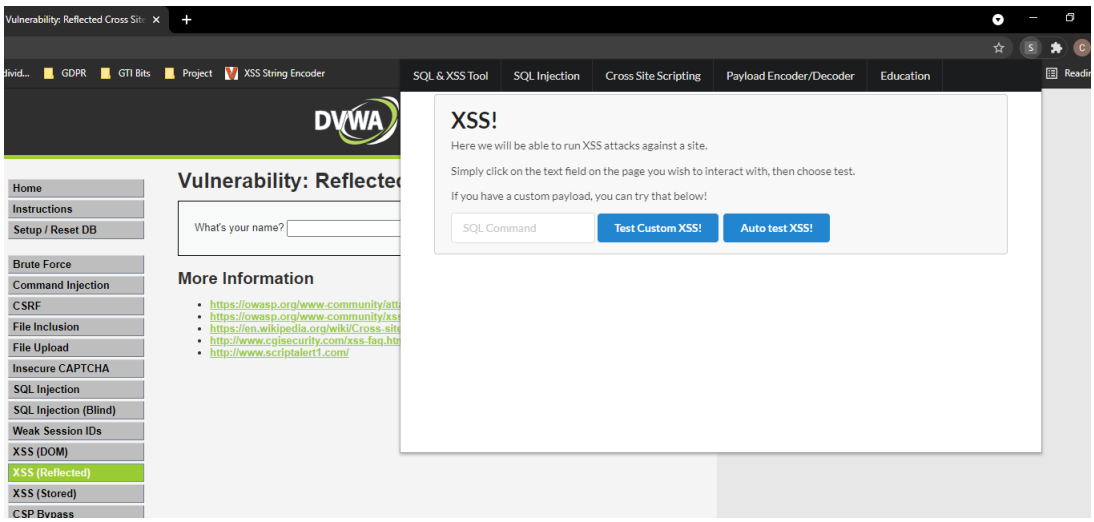
looked messy. The current implementation is an upgrade over my initial idea in this regard as it works with any web page the user visits.

DESIGN

The design also got an overhaul from my initial idea, which I believe gives a better end user experience. Below is a screenshot of the initial idea:



As outlined in my Design Manual, the initial idea was for a sidebar to be displayed over the left side of the browser, drawn over the current web page, the input fields would be drawn inside the extension and the option to run attacks would be at the bottom. Below is the current design:



The design I ended up with is better in many ways. First of all, I got rid of the idea of drawing all the input fields on the page within the extension to keep the user interface clean. Secondly, the extension simply pops up from the button in the top right of the browser, this is the default action for chrome extensions and I stuck with it for simplicity, allowing more time to work on other functionality.

DESCRIPTION OF LEARNING

TECHNICAL

This whole project has been a learning exercise from start to finish, both technically and personally. Everything I encountered along the way was new to me. I had never done much in-depth JavaScript and have never looked into chrome app development which made everything more difficult.

I went into the project thinking the web app was no more than a small web page that can interact easily with other web pages, that's were the issues began. Chrome web apps don't work how you'd expect. For example, inline scripts are not allowed in a chrome extension, so it therefore cannot directly interact with the DOM of the current web page due to security implementations. This became a problem as the design of the extension was done completely in HTML and CSS, so when it came to add the JavaScript functionality I didn't understand why the code wasn't working and stuff was breaking. Instead the interact using background and content scripts. The content script is run on every web page as soon as it's loaded, so can be used to gather useful info such as the amount of input fields on the page. Inside the content script is where you would have your functions, such as the function to type a payload into

the input field and execute it. This meant there needed to be messaging going on between the extension and the content script. The extension would send a message and the content would listen. This proved extremely different as it was a new concept to me. I worked on this method for weeks, trying to get different functions to run when certain messages were sent, with little to no luck.

Just when I was starting to lose hope for the messaging, I asked on stack overflow and within a couple hours, I had a response. "chrome.tabs.executeScript()". This one function does exactly what it says on the tin. It executes whatever script you tell it to in the current tab, what I thought and was told previously to be impossible! This cut my code down by hundreds of lines and allowed me to freely manipulate the DOM.

JAVASCRIPT

As 80% of my code was JavaScript, I needed to have a fairly good understanding of how it works, inside and outside a chrome extension. I found it extremely difficult to follow along with tutorials online as I wasn't sure what the code was actually doing. Following along was a copy-paste exercise, teaching me very little of how to apply what I learned to my own application. I took some time away from my project to get stuck into the basics of JavaScript and worked my way up to more intermediate/ advanced stuff, but due to being limited with time, I stuck to what would actually benefit me. I gained a deeper understanding of HTML manipulation which was vital to the workings of the extension.

It was nice to learn something new and to get back into a bit of coding, as difficult as it can be!

HTML & CSS

HTML CSS are also hugely important to creating a professional and premium look and feel to the extension. These topics are covered in the early years of my course, but as the course goes on they aren't a focus anymore. A quick refresher was all that was needed however, and I was on autopilot in no time.

I used a CSS library called "Semantic-UI" in the creation of the extension which cut my coding time down by a significant amount. The ability to place a button and assign it a class and have the library do the rest was a huge help.

PERSONAL

As with everything in life, there was absolutely some personal learning throughout the project. The usual take-aways being "I need to stop getting distracted", "I need to set more aside solely for project", and "I need to stop putting work off for 'another time'". These are always the personal challenges I face with everything in life, be it college or work. I have a hard time of saying "Right, I have X amount of work to do, I'll set aside some time on Monday for this and then Tuesday I can....." and then before the sentence is even finished I'm in the kitchen seeing if there's food in the fridge!

TIME MANAGEMENT

The fact that I am at this point now and the project isn't "complete" has absolutely taught me to be less like what I mentioned above. Perhaps the next project I take one in life I will remember the times I wasted valuable progress time during this project and won't let myself make the same mistake again. As much as I would like to be the student who says "putting away X amount of time a day towards the project was paramount to getting my project complete on time" I unfortunately am not. This is in part to my own lack of time management, pressures put on us from 5 other modules all looking for 100% from us at the one time, and of course Covid-19, which has had a huge negative toll in all aspects of our lives.

At home learning has not been easy for any of us this year. Lecturers feeling like they're teaching a brick wall and students having broadband issues, not being able to attend lectures, the list goes on. I feel it was harder to get help from lecturers this way and I felt really disconnected from it all. There were days I'd nearly forget I was in college at all. I am absolutely a student who needs to be there or its "out of sight out of mind".

THE POSITIVES

As much as I can put myself down for what I didn't do, I need to look at everything I did do! I took on a large project, learned new skills, and a new programming language. I also got a better understanding of topics covered throughout my course such as SQL Injection and Cross Site Scripting. It's always good to apply concepts learned in class into real life situations, you don't truly understand what you learned until you do. Another hands on approach to these vulnerabilities to revise the work we did in labs is what I needed.

As outlined above, I also developed some of my interests a bit more. Programming was something I was looking to get back into so I don't get rusty, and having to book up on multiple

programming languages at once certainly helped out with this! Though I was worried about the JavaScript side of things, due to its poor reputation among the entire programming community, I grew to somewhat enjoy coding in it, once you learn its quirks and features. There's something extremely satisfying in turning an inevitable error into a working feature in the application.

REVIEW OF PROJECT

CURRENT STATUS

I would like to say most of the development of my application was successful. Yes, on the surface it is missing a major component, but there's a lot you might forget to look at going on here. The extension itself, the UI, the learning of new programming languages and the documentation. When looked at as a whole, there's a lot of work gone into the project. The project has been brought to a stage where it is presentable and has many working features. Though it isn't available on the Chrome web store, there's no reason I couldn't upload it if the extension was fully complete as per my initial idea.

FUTURE VERSIONS

Of course, if I had more time, the extension would have much more functionality. The testing would be automated and reported to the user. For web developers, mitigation techniques would be suggested, which can be implemented and then tested again until the results come back negative. Other features such as an RSS feed tab would be added also, allowing the user to keep up to date with the current vulnerabilities/ mitigation techniques, to secure their application before they're even hit. I would also have coded the extension to work in more browsers such as Firefox if I had more time, to reach a larger target audience and expand the user base.

OVERALL

Overall I am proud of what I have created as it's functional and has a minimal design I am happy with. It may not be 100% what I set out to achieve, but there is progression there, and lessons learned I will take with me forever.

ACKNOWLEDGEMENTS

There are, of course, plenty of people I would like to thank for helping me along the way with this project.

My project supervisor, Paul Barry, for the weekly meetings which kept me on my toes and didn't let a week go by without seeing progress. His words of confidence as the project went on helped keep me motivated as the weeks went by, without which the will to finish the project may have lessened over time.

I would also like to thank my lecturer, Richard Butler, for the initial idea for the project, and being there for help on certain security implementation questions I had whenever I had them!

Though there are plenty of people I could list who gave me a kick up the behind and kept me on track, nobody did it like my mother. She was forever checking up on me to ensure I was working on my project and not distracted for the thousandth time that day! She was forever keeping me fuelled during my coding hours with my headphones on when I would forget to eat otherwise, without her motivation I would have slacked a lot more!