**Project Goal**

Retrieve and store individual’s Instagram login information (including username/phone number/email and password) without their knowledge.

**Overall Plan**

Send out a mass email to random Instagram users and warn them that their password will be reset if they do not briefly confirm their account information by clicking the provided link embedded within the email (phishing website).

**Challenging Issues**

Most issues arose during the planning and drawing board process of this task. Obviously the simplest and least-overhead way of implementing a fake website would be using a mixture of html, css, and javascript. This simplifies the creation of the site by avoiding the compilation of code within an unnecessary framework. A large framework is not needed for a one-page site like this one, however, this is what exposed the problem. Using solely html, css, and javascript, there is no built-in database structure. The main problem was figuring out what form of database would be appropriate to use. The sub-problem was attempting to figure out if the chosen framework would be able to accept communications from javascript protocol.   
  
Javascript was having issues connecting directly to the database. The challenging part to this issue was the lack of documentation on the web and the lack of error messages using javascript.

**Approaches and Techniques**

Luckily, one of the first attempts of database connection was successful. From extensive use with Ajax calls, it could be inferred that data could be transported to an excel file considering Ajax calls are made within javascript. As long as the information from the form could be extracted into a javascript function, that same information, in theory, will be able to be transported across a network.   
  
An Ajax call is basically a way of sending data in the form of json to server-side applications. The server-side storage application chosen was Google Sheets. The desired process begins with extracting the individual’s information form a simple web form. That information was then processed inside a javascript method. This javascript method checks to make sure all fields are filled with data and removes all characters of the password except for the first and last character (otherwise, this is illegal). After the data is processed, the javascript method then places the data into an Ajax method and sends the data to the Google Sheet database.   
  
Data could not be directly transmitted to the Google Sheet database in the absence of an appropriate framework. Avoidance of using a large framework with overhead was strongly desired. Knowing that Google Forms could easily submit data to Google Sheets, testing began to determine if data could be sent successfully to a Google Form. After a Google Form was implemented, that form was set to send data to the Google Sheets database that was previously implemented. Via Ajax through javascript, data was successfully sent and received by the Google Form. The Google Form’s text variables were filled with the Ajax data and successfully submitted to the Google Sheet. After assembling the Google Form middleman, the data was received from the user and stored into an online database without the use of a large framework.   
  
After the data is processed successfully, the user is redirected to Instagram’s main page to give the impression that the process was carried out by Instagram’s server, and not an adversary seeking confidential information.

**Tools Used**

* Visual Studio Code
* Google Chrome
* Instagram login pages (for reference)
* Google Sheets
* Google Forms
* Github

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedResults**

**Ajax Google Form Google Sheet**

Data is successfully transmitted through all three of the formats displayed above (starting from the left and ending at the right). The final result lies within the Google Sheet. The Google Sheet contains the username entered, the password entered, and a timestamp of when the entry took place.

**Conclusion**

I have learned the usefulness of Ajax calls and the various ways they can be used. If more time was provided, I would like to enhance the UI of the site. After this development, I realized how easy it would be to mistake a phishing site as an official site. I was able to make the phishing site look almost identical to an official Instagram login page (<https://www.instagram.com/accounts/login/?hl=en>).   
  
It is important to always check the source you are dealing with when browsing the web. Never enter personal sensitive information into a site that you are unfamiliar with and never click anonymous links sent through email. I have learned that it is very easy to trick individuals into providing you with their personal information.

**How to use the tool**

* Github repository:   
  <https://github.com/24hawkman/PlentyOfPhish>
* Google Sheets database link:   
  <https://docs.google.com/spreadsheets/d/1AWcYJunnufHZI087lDlxzJXtkBbxMfkW3Cqm9MfVkhE/edit?usp=sharing>

1. Clone the repo:   
   git clone https://github.com/24hawkman/PlentyOfPhish
2. Open Main.html in your web browser (I used Google Chrome)
3. Enter information on the mock phishing website. Submit
4. Once submitted, your information will be recorded into the Google Sheets database (accessible by link provided above)