Assignment 7

Mark, Henry, Jun, Tolu, Philip

**XXS :** Henry Qiu

**IP :** 73.61.15.252

1. What vulnerability did you exploit? How does it work?

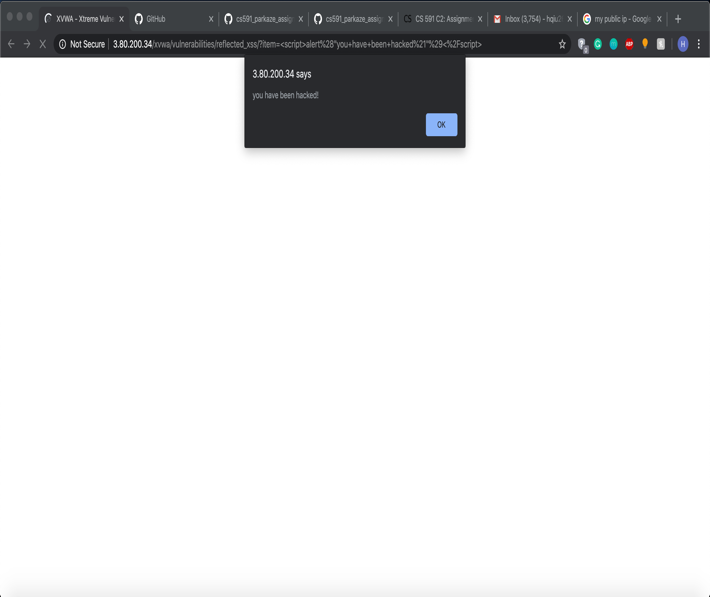
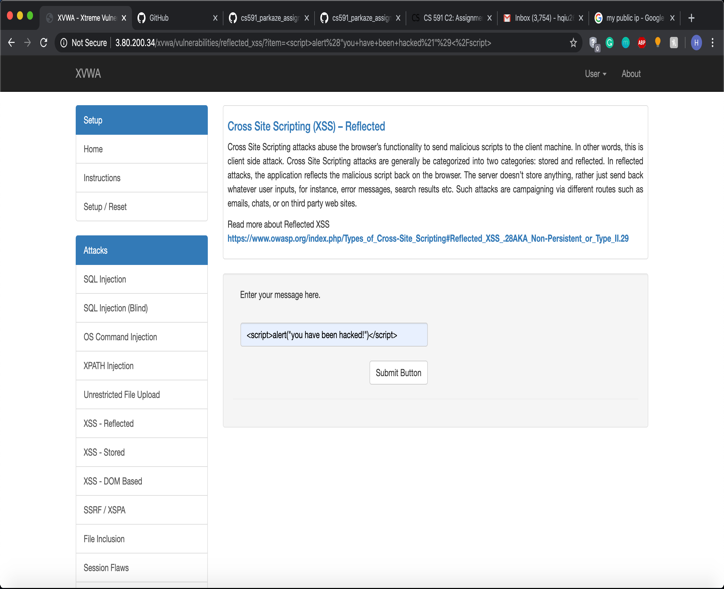
I exploited the reflected xss vulnerability which was performed by entering the script, “<script>alert("you have been hacked!")</script>” into the message field which runs the alert with a message. This is because the web application receives the HTTP request and sends it back to the user without being sanitized. In doing so, the user could be sent a script that sends them to a malicious website. This attack can be embedded into a link, such as, “<http://3.80.200.34/xvwa/vulnerabilities/reflected_xss/?item=%3Cscript%3Ealert%28%22you+have+been+hacked%21%22%29%3C%2Fscript%3E>” which can be sent to an unsuspecting user and lead them to run the script.

1. Is your application subject to this particular vulnerability? Why/why not?

No, because our mobile application uses firebase when receiving data back to the application. In doing so, firebase sanitizes and escapes all html pages which stops the application from executing any HTTP responses in an unintentional way. In addition, due to it being a mobile application, it is not vulnerable to xss because there does not exist an embedded web application within it.

1. How will you protect against this vulnerability in your application?

The vulnerability can be protected by data escaping which prevents any data being viewed as part of the code, and will not be exploited through the use of scripts and html in the url field. Nevertheless, our mobile application is not currently connected to a web application, therefore, there is a lack of vulnerability.

****

**SQL Injection Blind:** Philip Huang

IP : 209.6.72.158

1. What vulnerability did you exploit? How does it work?

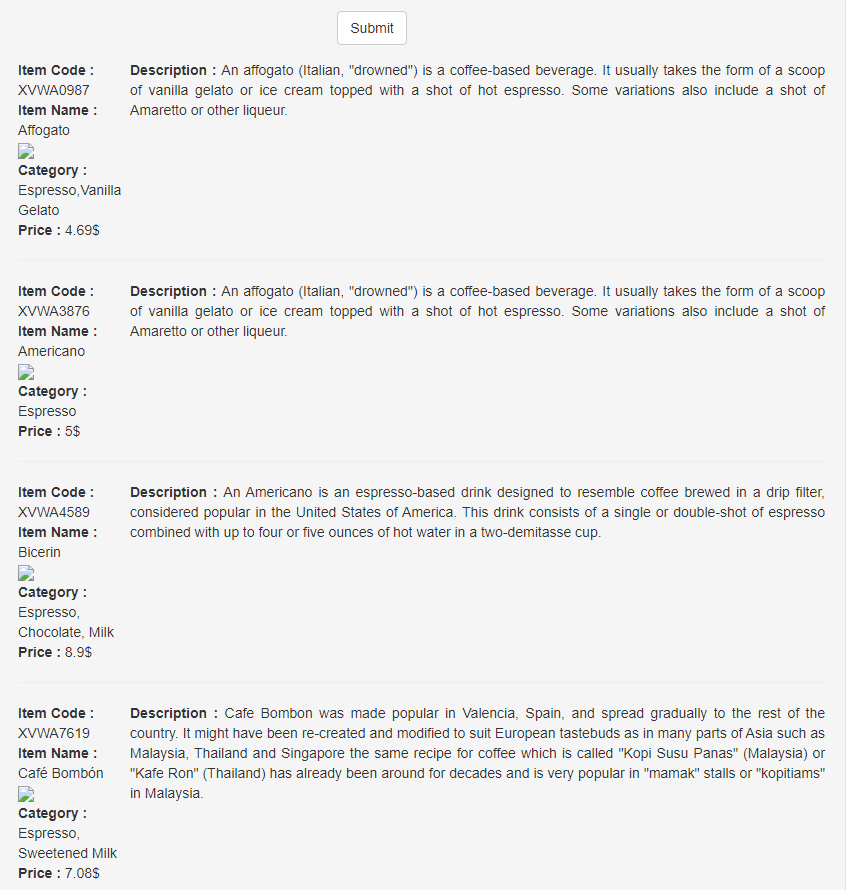
The Blind SQL Injection. I injected (' or TRUE#) without the parenthesis, which is queried as (search='' or TRUE#). This Boolean condition will return TRUE, and since # is the comment delimiter for sql, everything after TRUE will be commented out allowing us to successfully view all of the things in the database.

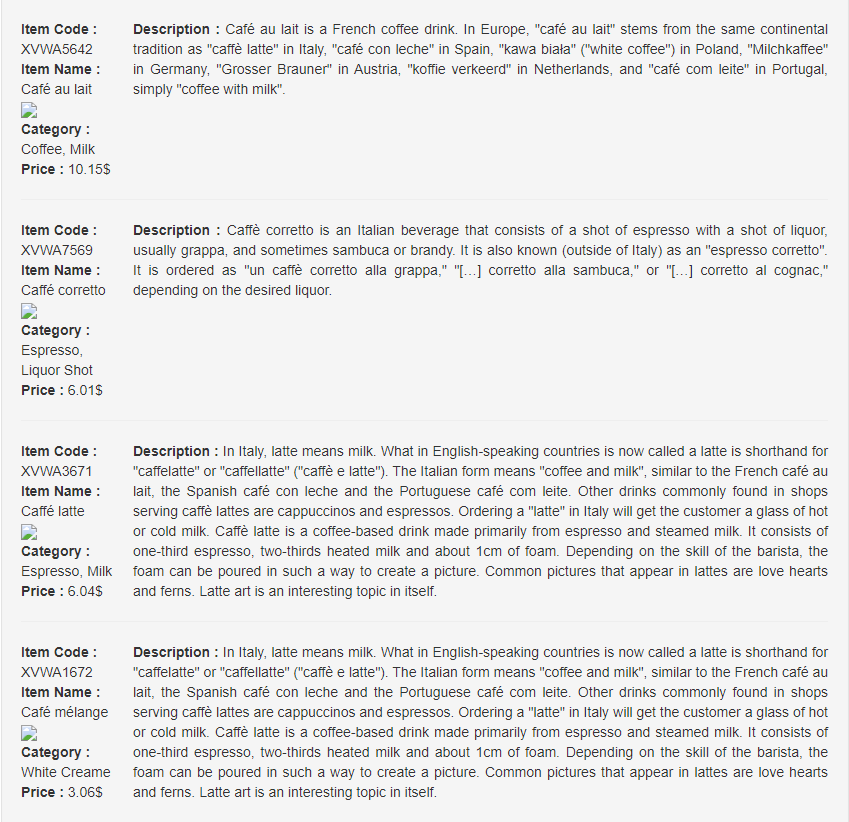
1. Is your application subject to this particular vulnerability? Why/why not?

My application is not vulnerable to this type of attack because Firebase is a NoSQL based cloud storage where everything is stored as JSON objects.

1. How will you protect against this vulnerability in your application?

Two ways to protect against this vulnerability is to not use SQL based databases, or to properly sanitize inputs into the form.

****

****

****

**CSRF :** Jun Hao Lei

IP : 2601:182:cd01:9a83:d1a2:5fb5:876f:ce8c

2601:182:cd01:9a83:e5b0:fbfe:fc50:90b6

1. What vulnerability did you exploit? How does it work?

Heres the html code that the victim user has to click:

<html>

<body>

<img src="http://3.80.200.34/xvwa/vulnerabilities/csrf/?passwd=freemoney&confirm=freemoney&submit=submit" onerror = "loadImage()">

<script>

</script>

</body>

</html

I exploited the csrf vulnerability of the website. I made an html file called csrf\_nsfw that executes a img that loads a script to a link http://3.80.200.34/xvwa/vulnerabilities/csrf/?passwd=freemoney&confirm=freemoney&submit=submit. The image doesnt actually load into anything so the victim wouldnt know what actually happens.

This js link call by use of image will load that exact link and will fill in password as freemoney and confirm as freemoney and also submit to model how an actual person who decided to change password manually will do. But because the victim is logged in and the page does not redirect to the actual page, the victim will unknowingly get his password changed by the attacker.

1. Is your application subject to this particular vulnerability? Why/why not?

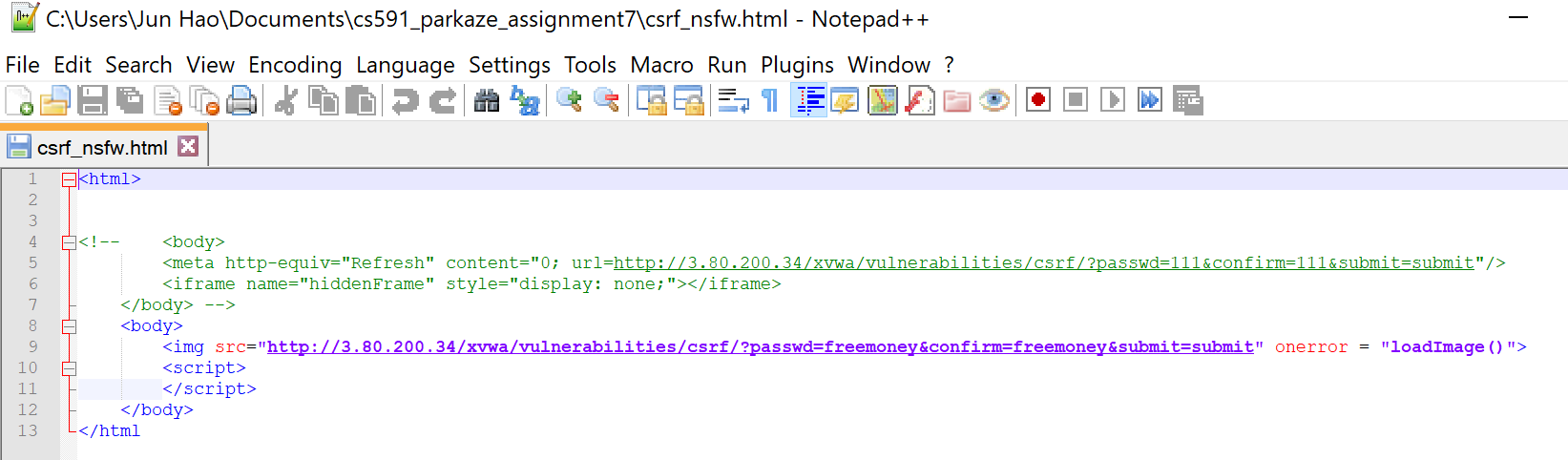
Our application parkazes is a mobile app and is not subject to link redirect attacks that fills in password values on click. Even if it was form request attack which I tried but for some reason would not actually fill in the form for the xvwa website, our app is not vulnerable to request of this type as since webform attacks are different in the csrf variant in mobile apps. Thus our app parkaze is not vulnerable to the attack.

1. How will you protect against this vulnerability in your application?

We do not handle request forgery attacks via passwords in this application. That is already built upon by the parkaze group prior to us entering the group. It is a mobile app and we handle front end and backend for new/upgraded functionality of the app.

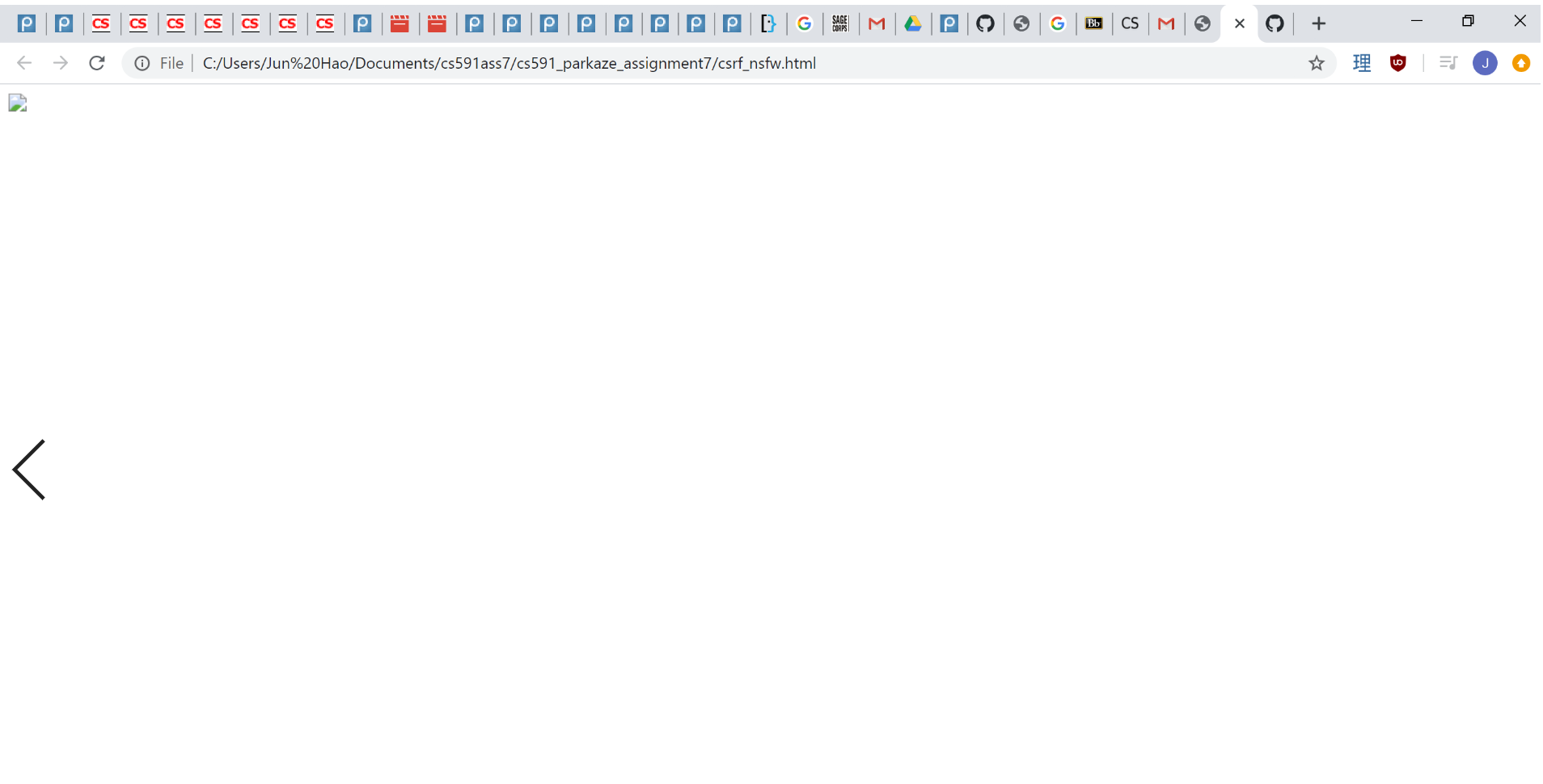
1.

Client opens un unsuspecting html file while logged in…



2.

Opens a weird broken error image but if you are logged in it will change the password…:



**OS Command Injection :** NAME

IP :

1. What vulnerability did you exploit? How does it work?
2. Is your application subject to this particular vulnerability? Why/why not?
3. How will you protect against this vulnerability in your application?

**PICTURE HERE**

**Unrestricted File Upload :** Mark White

IP : 81.185.170.49

1. What vulnerability did you exploit? How does it work?

The first exploit was uploading unrestricted files by ignoring client-side restrictions by editing HTML with inspect element on Safari. I noticed that img and js directories were viewable as file systems. Inside the img directory, there is another directory called uploads which is where the uploaded are stored. One can access the uploaded file by navigating the directories, thus the uploaded file executed. <http://3.80.200.34/xvwa/img/uploads/m.php>

Using php, we can then modify the other files in the website and potential operating systems.

1. Is your application subject to this particular vulnerability? Why/why not?

Potentially, all images are uploaded to firebase which probably limits the permissions of the client to execute files on firebase servers. Locally on the IOS app, IOS restricts app abilities to execute code and services so any damage would be local to the app.

1. How will you protect against this vulnerability in your application?

Minimise the app privileges on the IPhone and do better file type checking rather than name checking.

