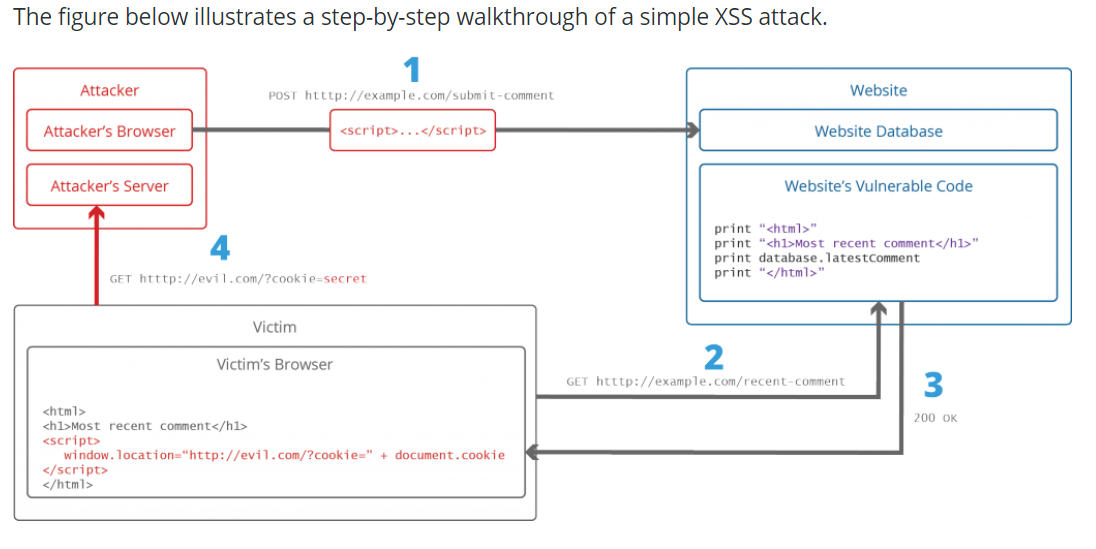
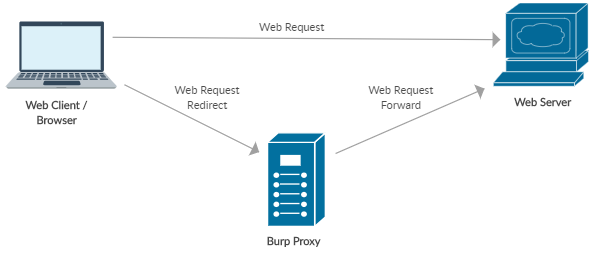
**Introduction-**

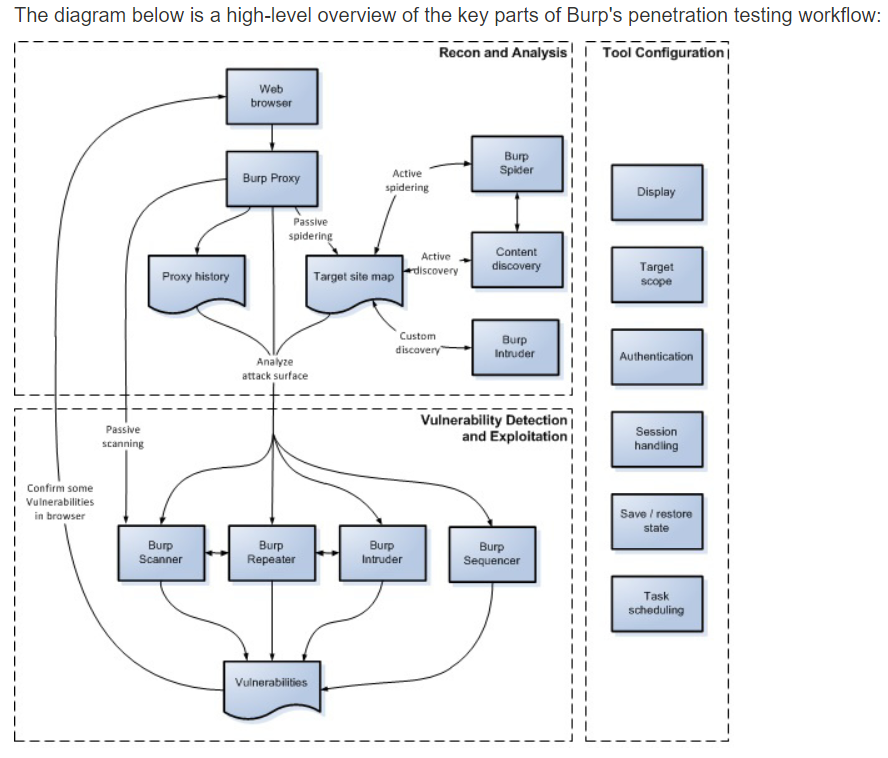
The main objective of this project is to perform session hijacking, cross-site scripting, sql injection, brute force attacking and reverse brute force attack via password cracking, on a self-made locally hosted website. The Session Hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token. Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker stores malicious script in the data sent from a website's search or contact form. In XSS, Attack payload is executed as a result of modifying the the original client side script, so that the client side code runs in an “unexpected” manner. We will be injecting scripts using some tools as a payload. Delivering a payload directly to the victim. Victim requests a page containing the payload and the payload comes embedded in the response as a script.

**Proposed System-**



**Architecture Diagram-**





**Literature Review-**

**Preventing Persistent Cross-Site Scripting (XSS) Attack By Applying Pattern Filtering Approach-** Cross-Site Scripting (XSS) vulnerability is one of the most widespread security problems for web applications, which has been haunting the web application developers for years. Various approaches to defend against attacks (that use XSS vulnerabilities) are available today but no single approach solves all the loopholes. After investigating this area, we have been motivated to propose an efficient approach to prevent persistent XSS attack by applying pattern filtering method. In this work, along with necessary background, they presented case studies to show the effectiveness of our approach.

**Impact Analysis of Preventing Cross Site Scripting and SQL Injection Attacks on Web Application-** Web applications provide immeasurable large facilities to the users. The usability and popularity of web applications have expanded. This has caused various types of attacks over them. SQL injection and XSS (Cross Site Scripting) attacks are very famous to exploit the web applications. The proposed Intrusion Detection System is a container based approach that is based on a mapping model. In this, a request to query mapping is applied to recognise and prevent such class of attacks. The impact measurement of this container based approach on the web server is calculated using http load and autobench tool. The web application performance measurement based on various parameters such as average page time, pages per second, memory and processing time for container based approach has been carried out and compared with the existing approach.

**A Comprehensive Inspection Of Cross Site Scripting Attack-** Cross Site Scripting attack (XSS) is the computer security threat which allows the attacker to get access over the sensitive information, when the javaScript, VBScript, ActiveX, Flash or HTML which is embedded in the malicious XSS link gets executed. In this paper, we authors have discussed about various impacts of XSS, types of XSS, checked whether the site is vulnerable towards the XSS or not, discussed about various tools for examining the XSS vulnerability and summarizes the preventive measures against XSS.

**CROSS SITE SCRIPTING (XSS) ATTACK DETECTION USING INTRUSTION DETECTION SYSTEM-** Nowadays diverse kind of attacks is being launched in Cyber Space among which Cross-Site Scripting (Web Application Attack) is amongst top attacks of all time. Proposed work, suggest an outline for a system that can detect Cross-Site Scripting (known as XSS) attack using Intrusion Detection system (IDS). This work focuses on the detection of XSS attack using intrusion detection system. Here attack signature is utilized to detect XSS attack. To test the usefulness and effectiveness of proposed work a proof of concept prototype has been implemented using SNORT IDS. It is observed that proposed system correctly detected XSS attack.

**Implementation-**

**Tool used-** To host the website locally, **XAMPP** server has been used. The web browser that has been used to perform these attacks is **Mozilla Firefox**. The software used to perform these attacks and detection is **Burp Suite Community Edition.**

1. **Session Hijack-**

-Run alert(document.cookie)

<script>alert(document.cookie);</script>

-Use Burp Suite to add proxy, analyse traffic and copy cookie id.

-Paste that cookie id to new browser and session continues from their.

1. **Steal information by injecting form in the web page using some vulnerable scripts-**

-Find any text area in form which reflects same input into the web page as entered.

<h3>Please login to proceed</h3> <form action="http://evil.org">Username:<br><input type="username" name="username">

</br>Password:<br><input type="password" name="password"></br><br><input type="submit" value="Logon"></br>

-Inject malicious script into that form and see resulting output.

-Inject script containing code for malicious form asking username and password.

-When user enters username and password it is shown in burp suite or can be redirected to our server.

**Defence/Detect-**

1. **Use Burp Suite to check for malicious target spots in our website.**

-Check if at any place on web page can we insert data.

-Download list of all possible javascripts payload and inject them one by one in that place.

-Burp Suite does this smoothlessly, it has inbuilt payloads also which can be used too.

-After running all scripts Burp Suite shows status, if its 200 means script injected, like this we can find vulnerability in our page.

**Steps in Burp Suite-**

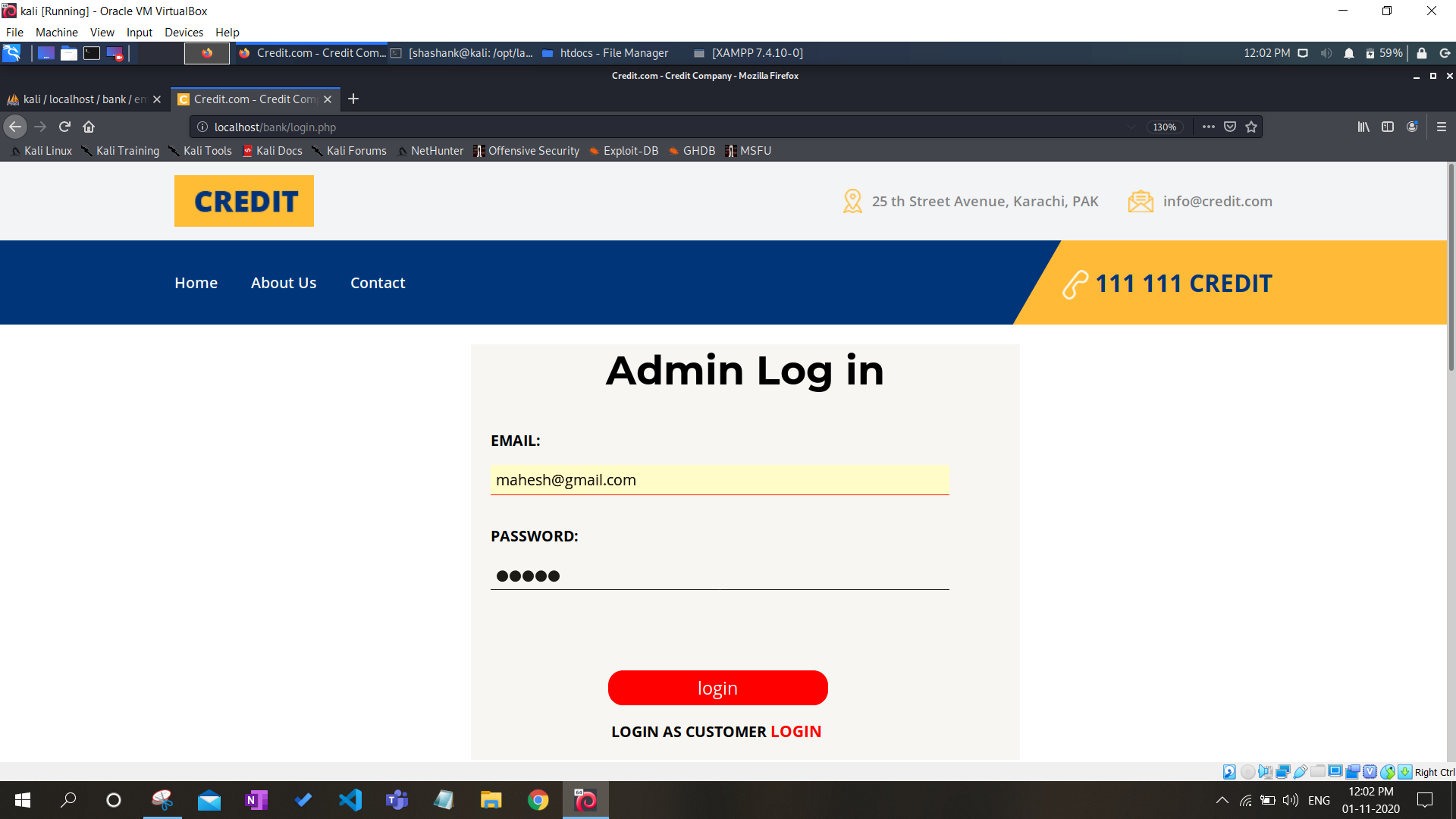
1. Set your browser to local proxy setting of 121.0.0.1 as ip and 8080 as port address for local host so that burp suite can start detecting requests using proxy.
2. Run website in localhost and detect requests made by it in Burp->Proxy->Intercept tab.
3. To check for vulnerable page, if that request has anywhere “variable name”=”some value” then send this request to Repeater.
4. In Repeater we will check whether our changes in variables value is reflected in web page or not, if it is reflected it is vulnerable to XSS otherwise not.
5. If it is found vulerable to XSS send this page to Intruder and configure position where payload to be inserted.
6. Add payload from list available to upload your own list.
7. Hit attack and check output.

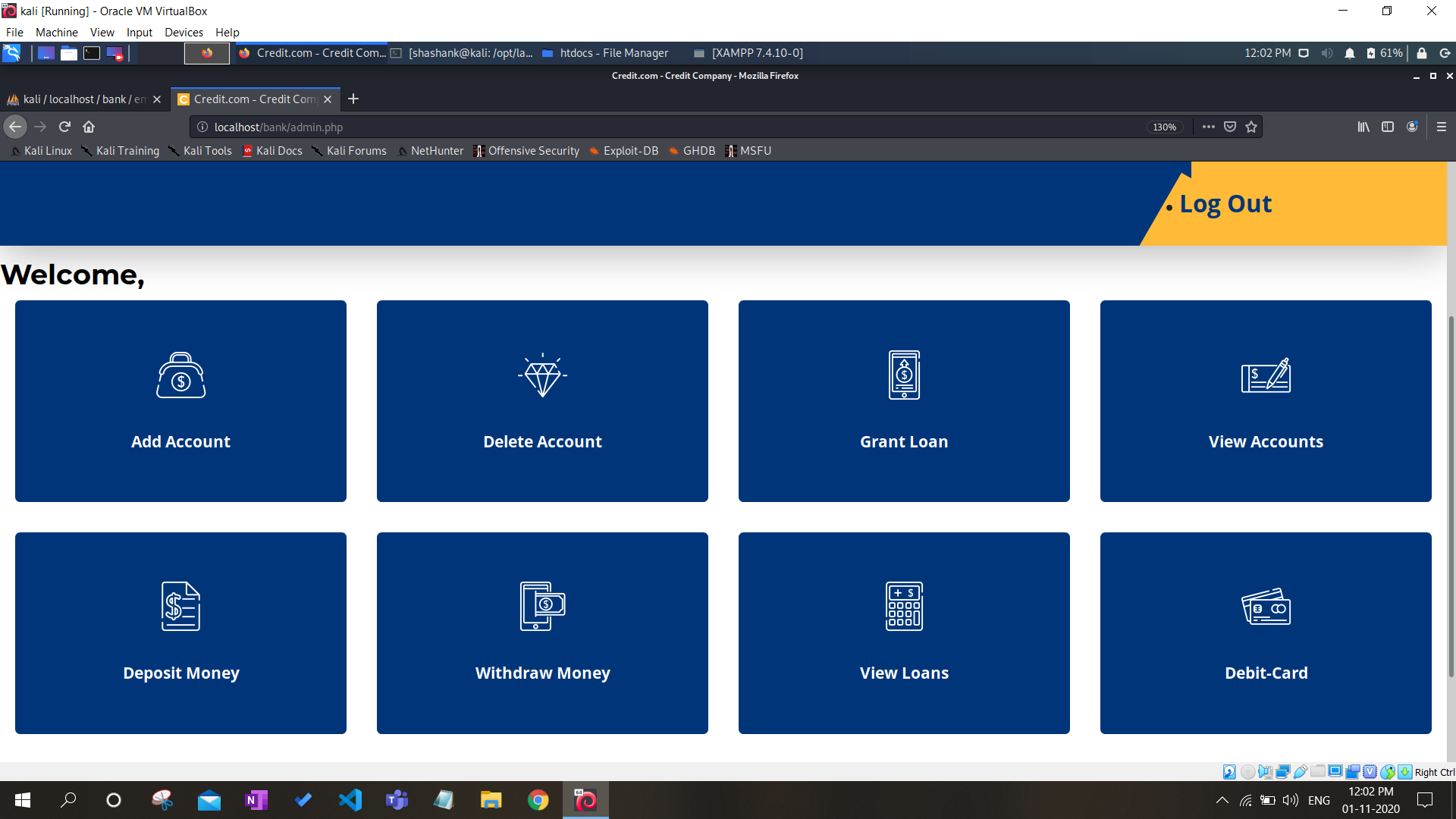
**Screenshot-**

**Main website-**



Login page-



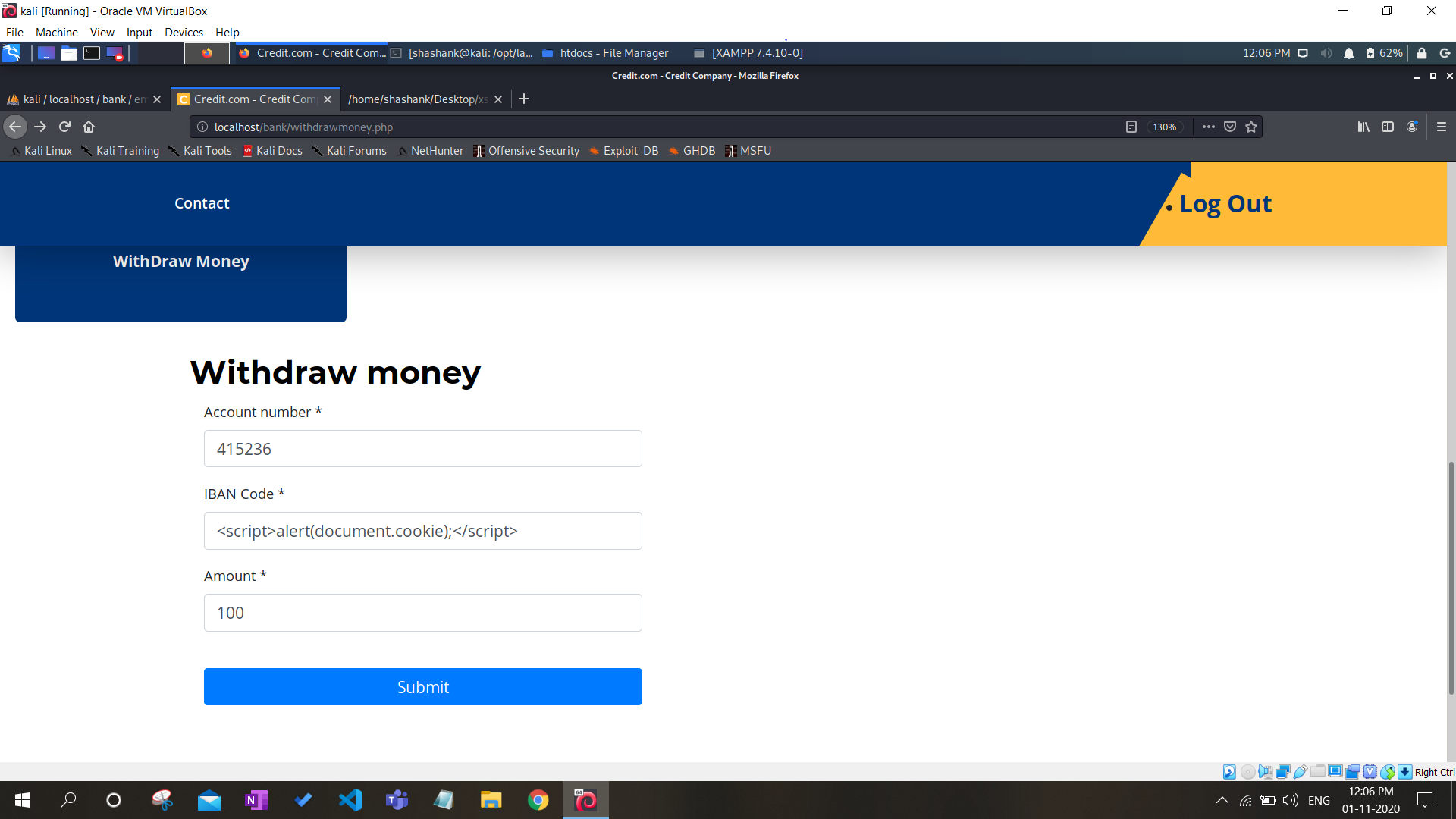
Main page-

**Attack-**

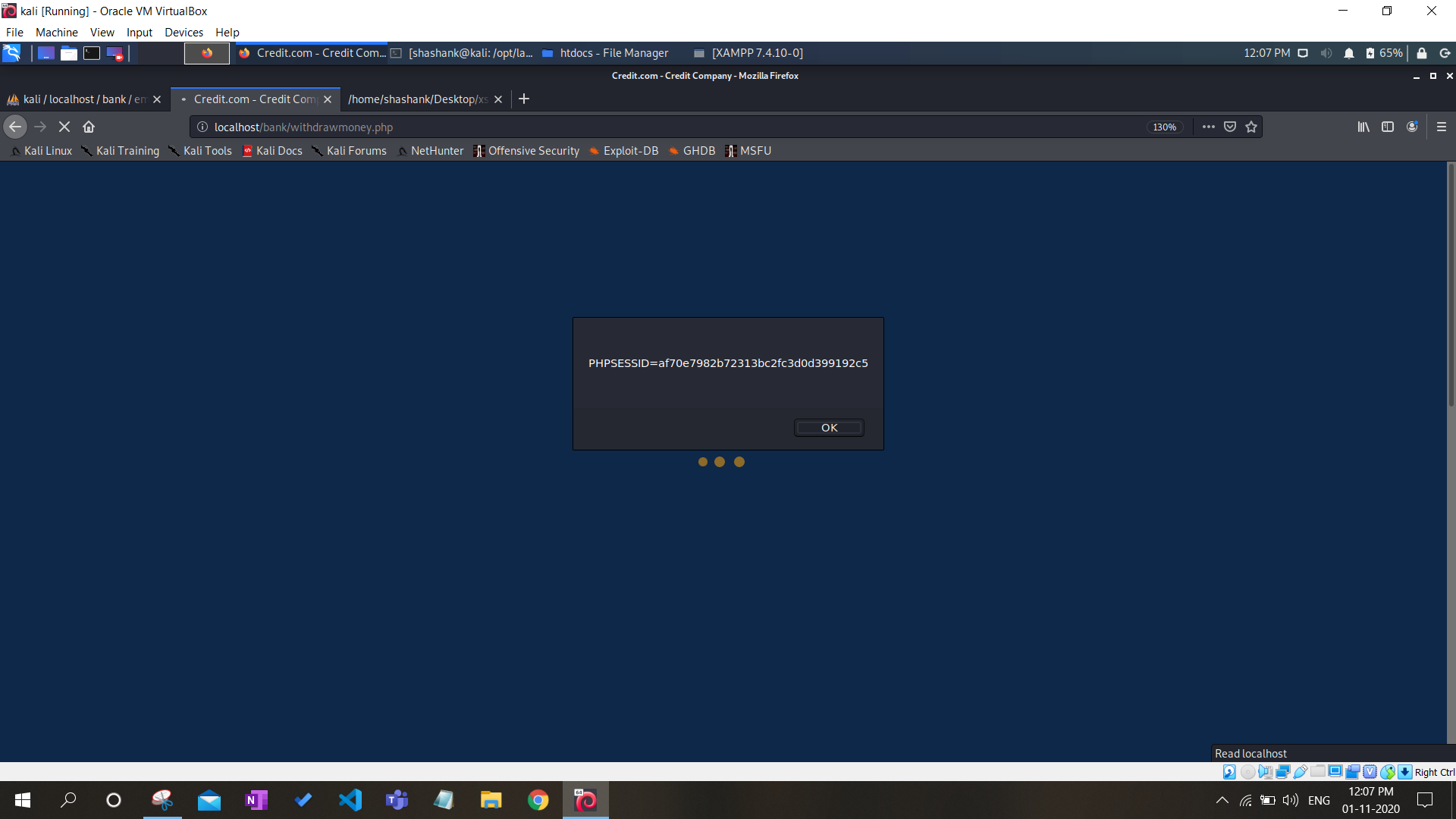
**Session hijack-**

Getting Session cookie through injecting malicious script-

Inserting script-

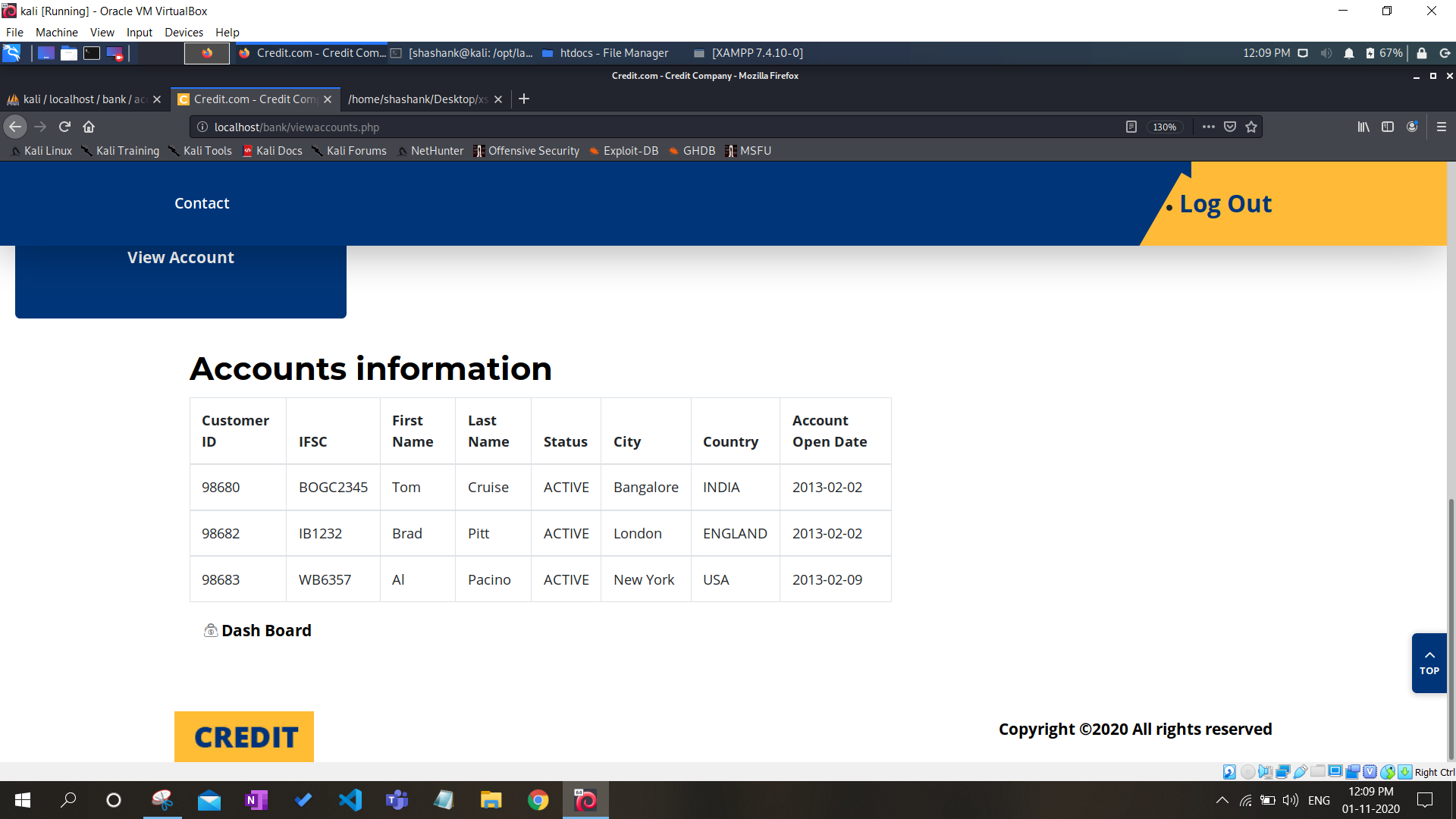


Got session cookie as reflected output due to XSS vulnearbility-



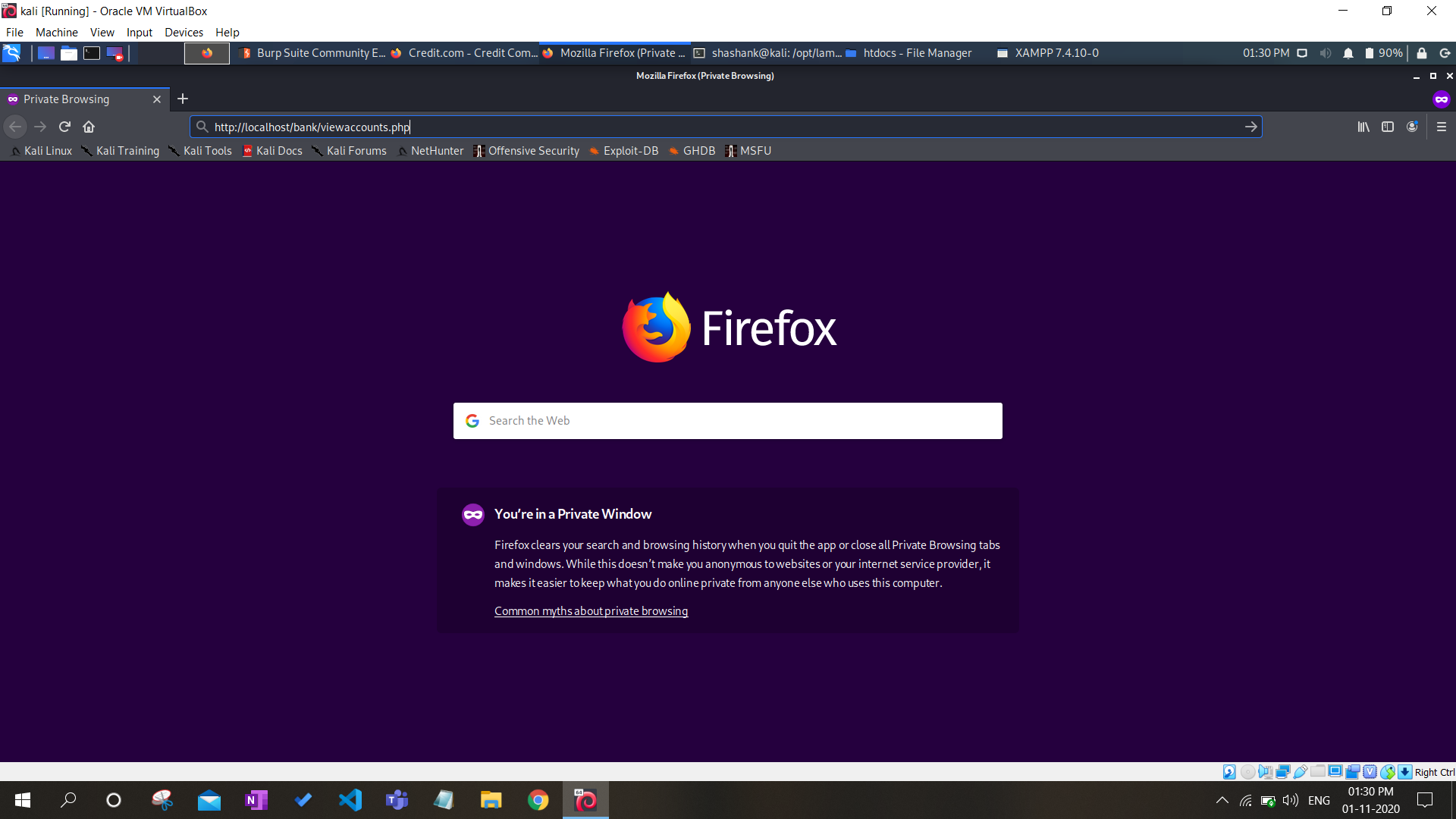
Hijacking the continuous session using cookie id-

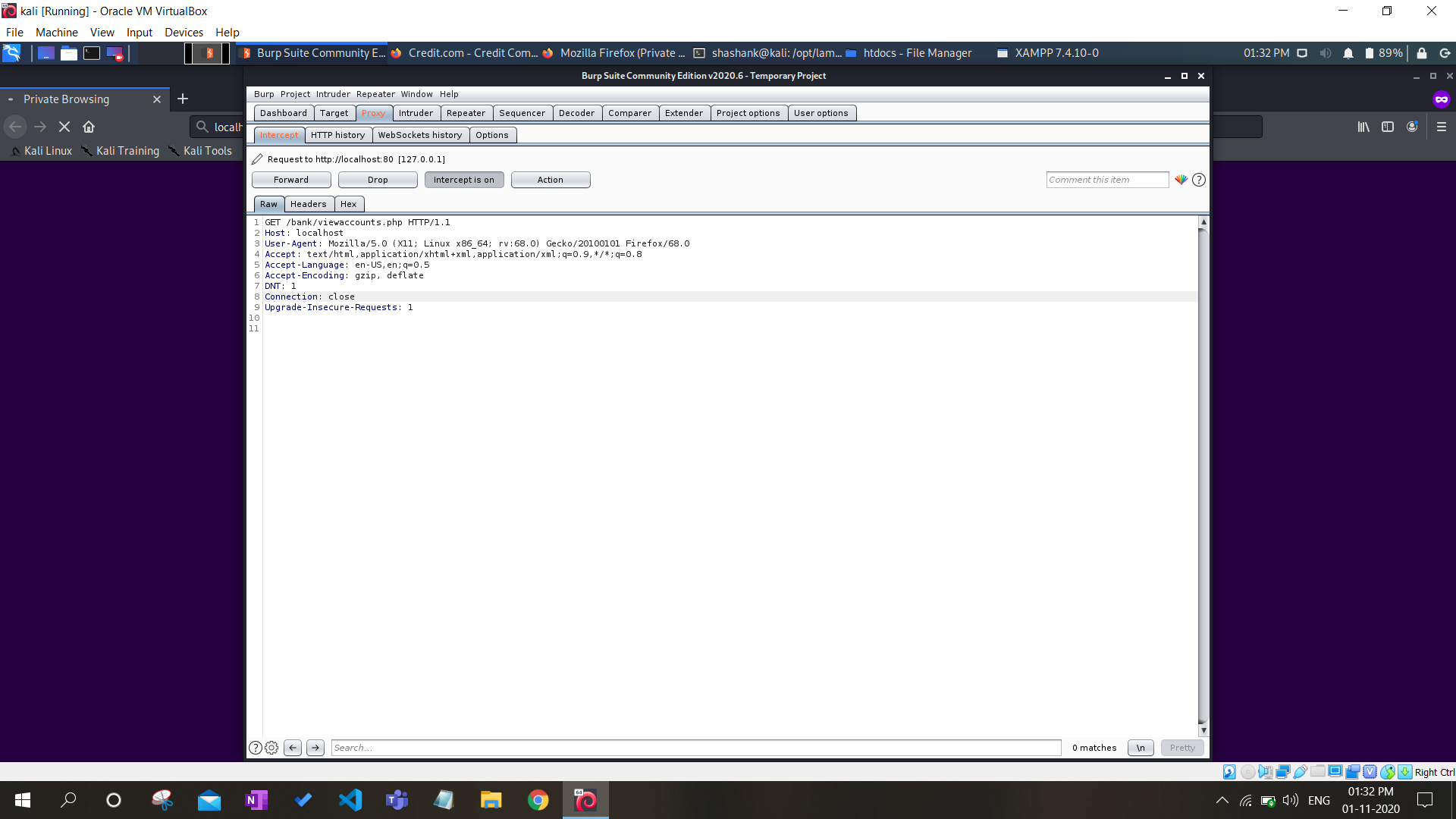
View account page for normal user-



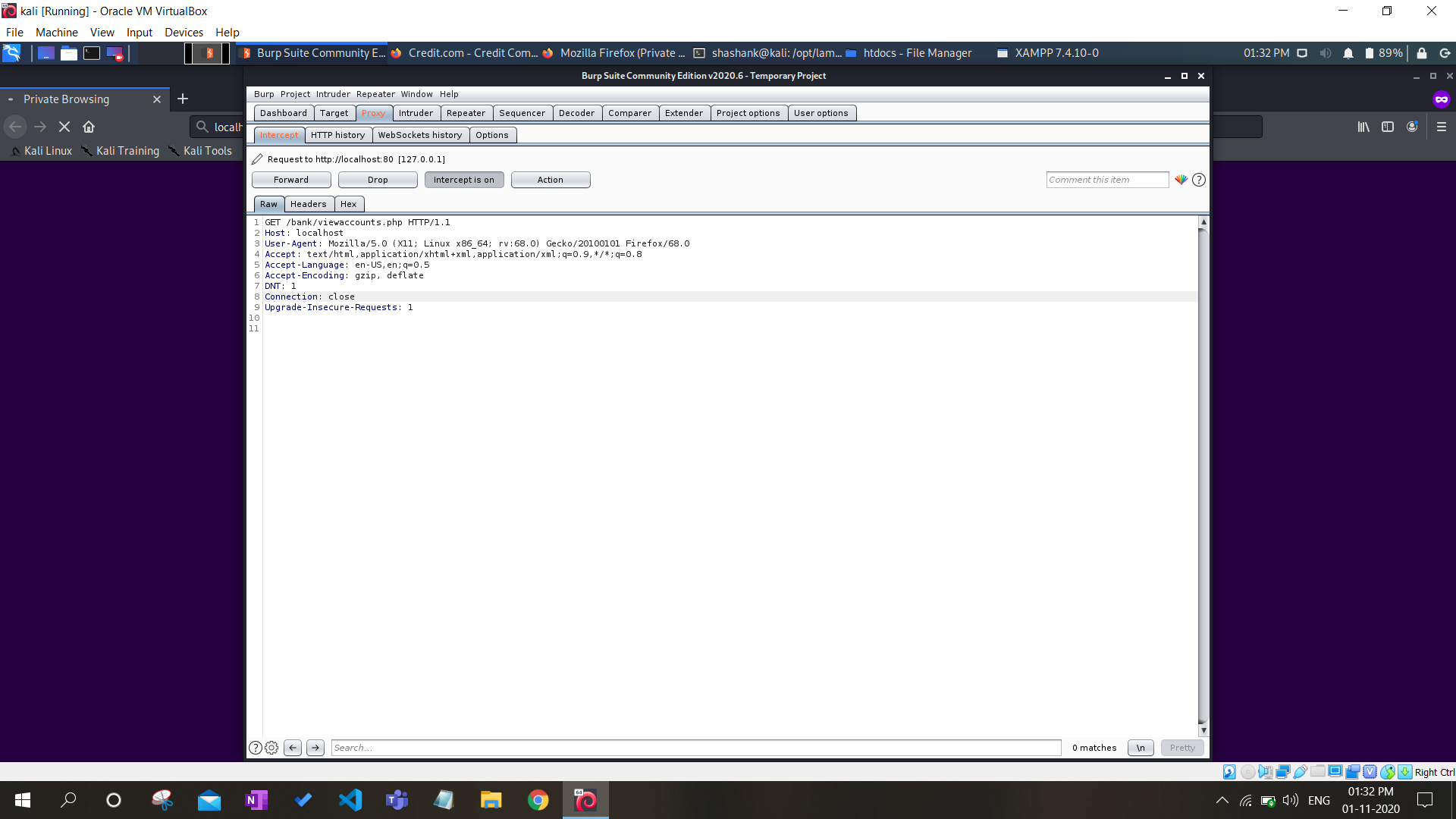
View account page for session hijacked page-

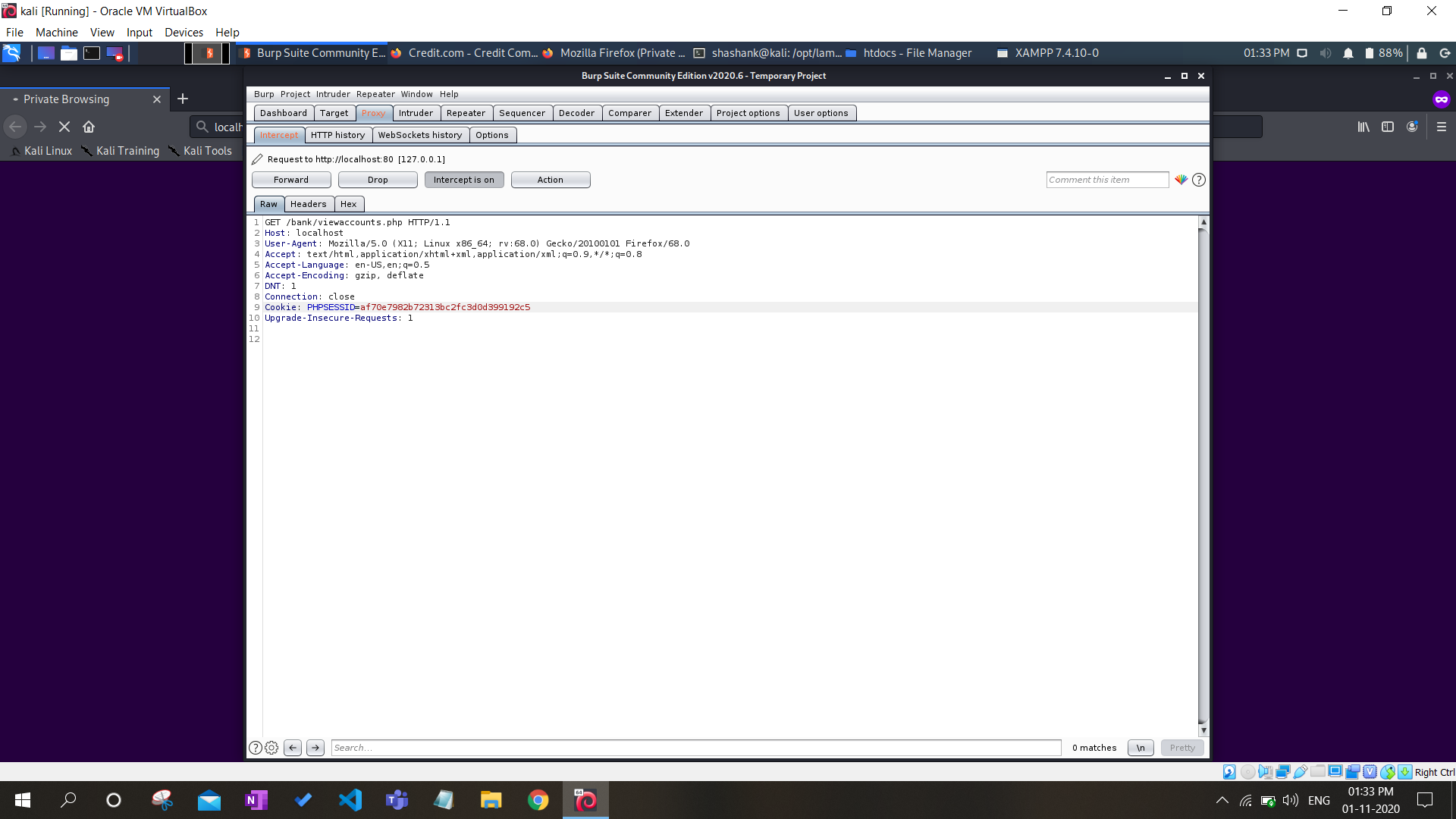
Copying link of page to be viewed in private window-



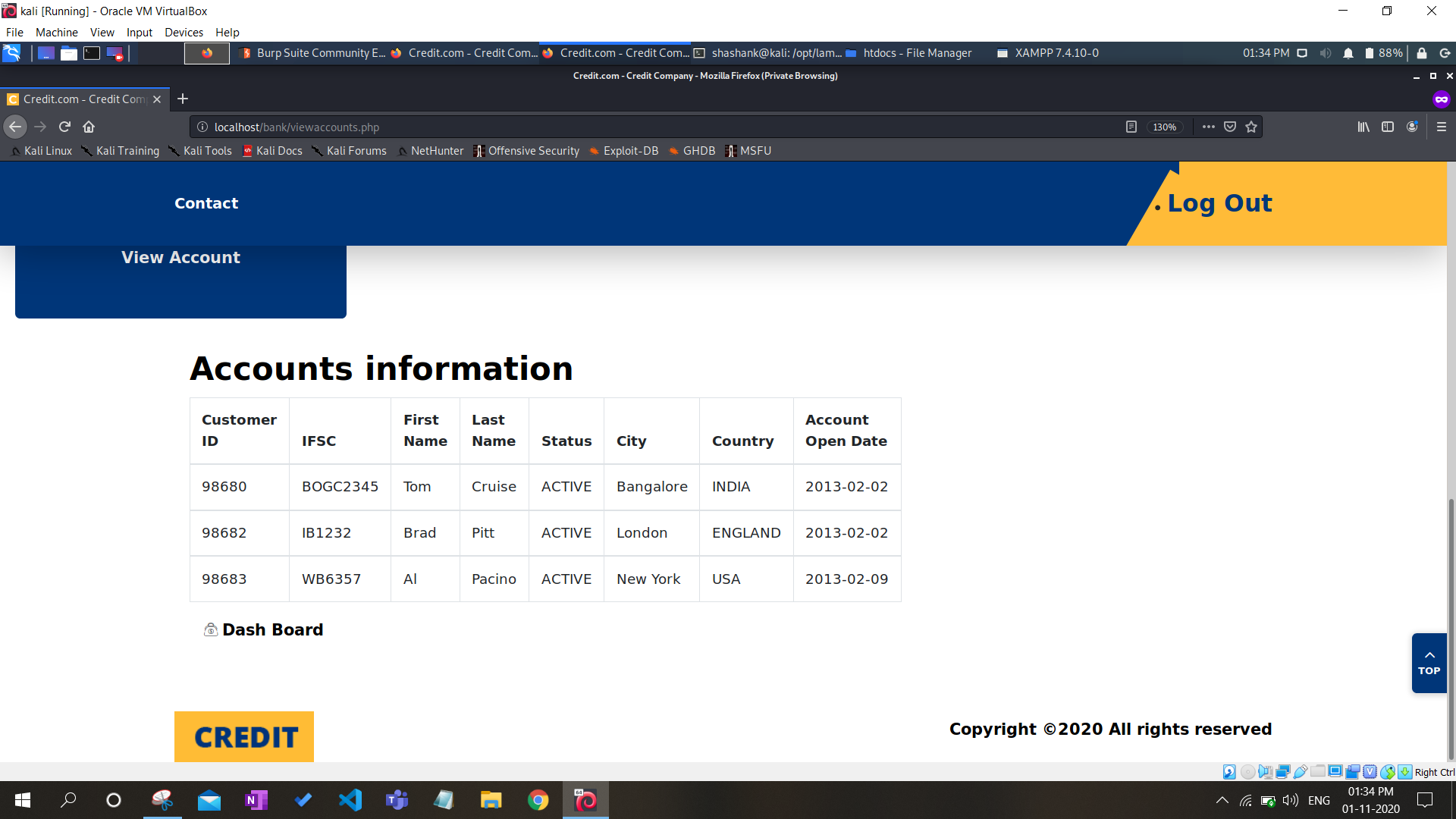


Pasting Cookie id or session id in burp suite-

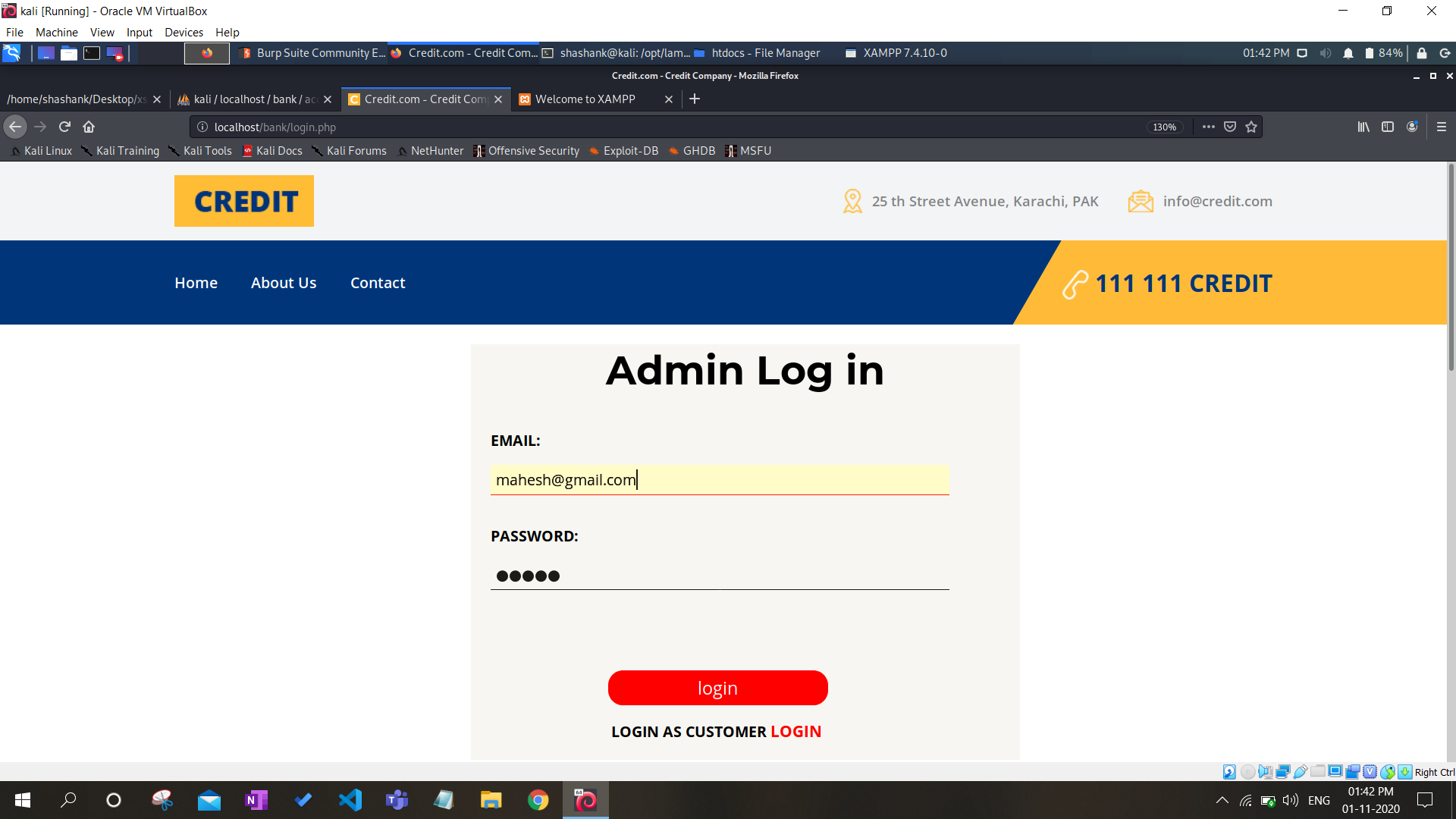


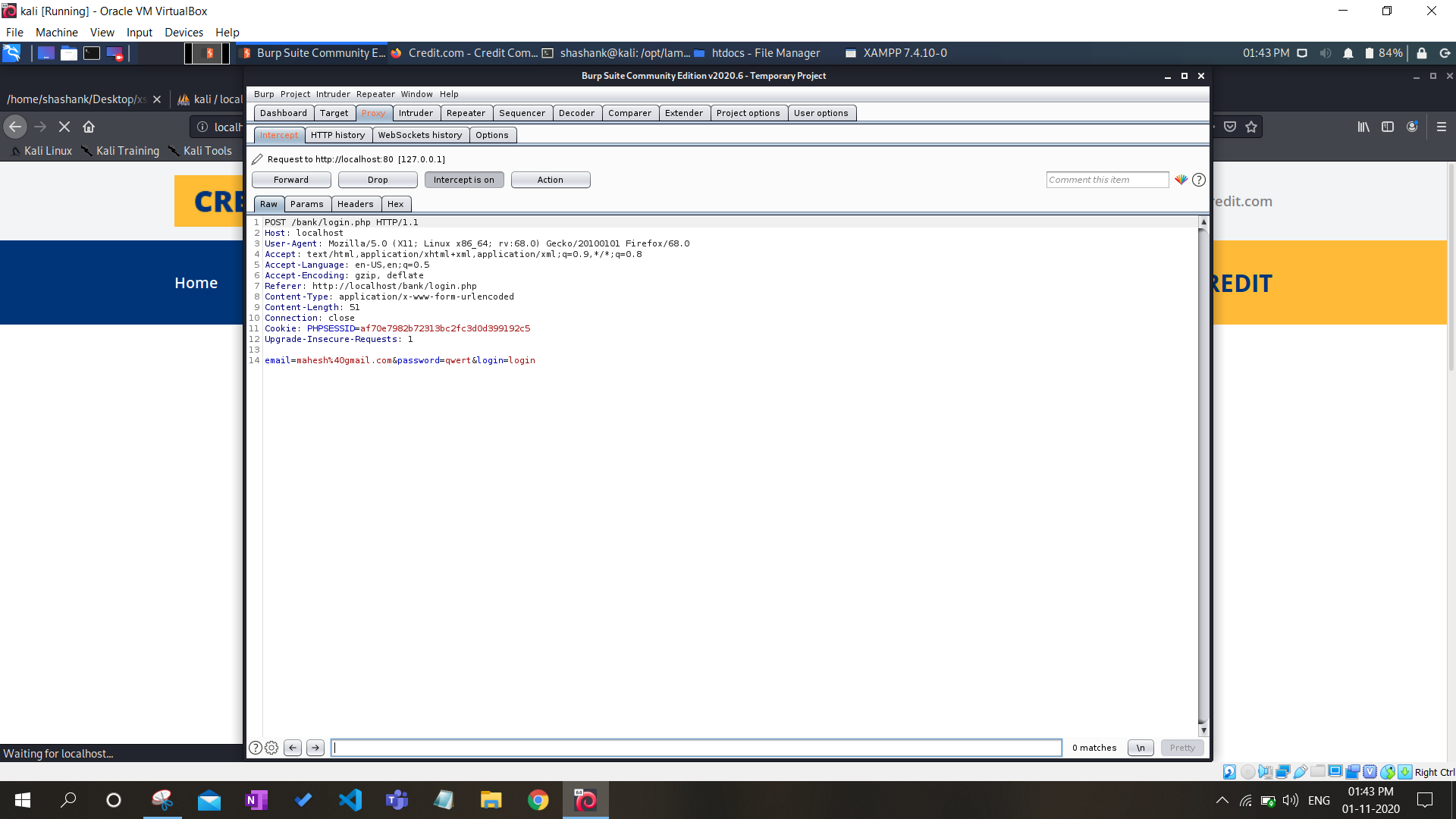


Session hijacked page-



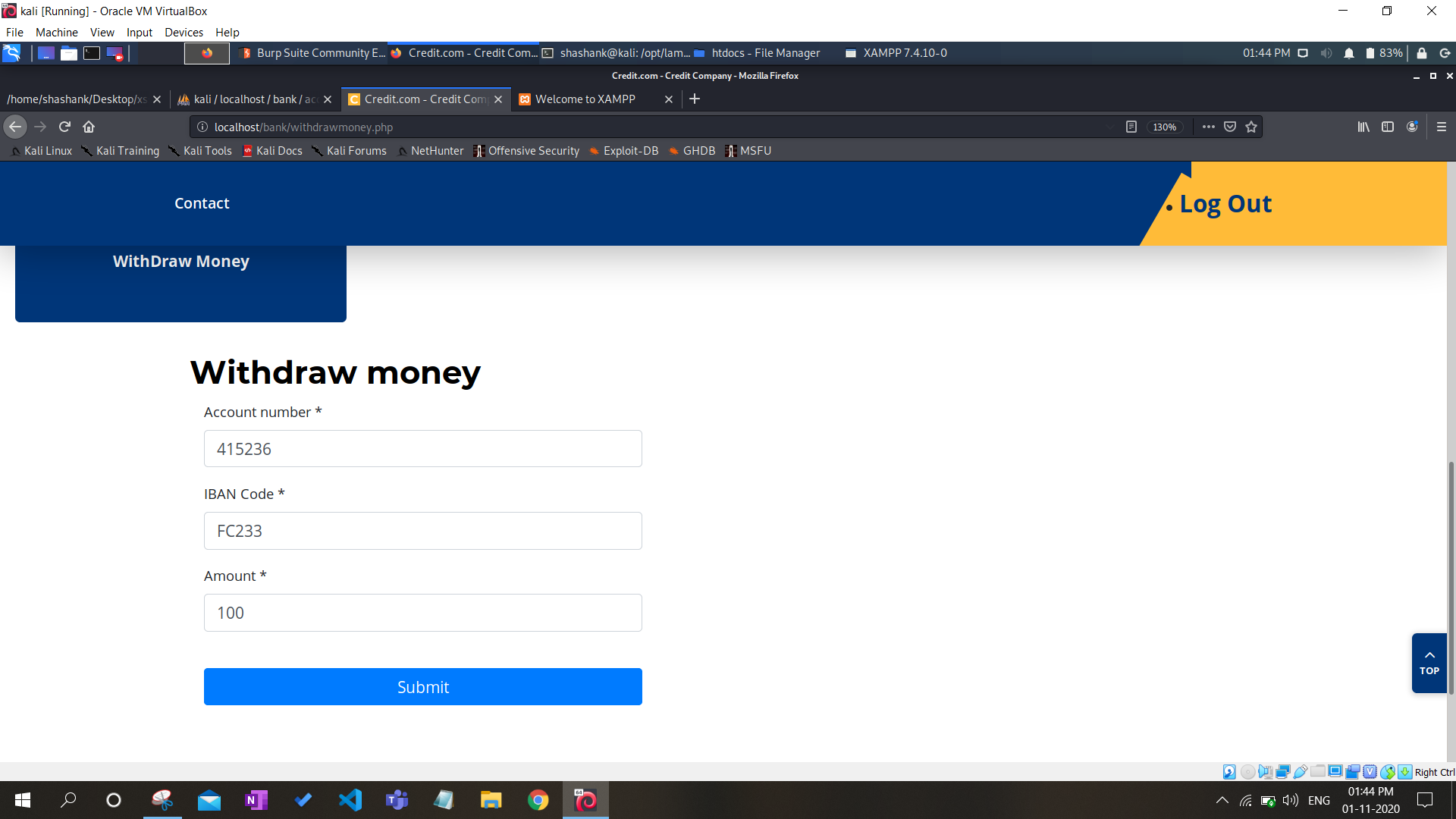
Stealing username and password using BurpSuite-



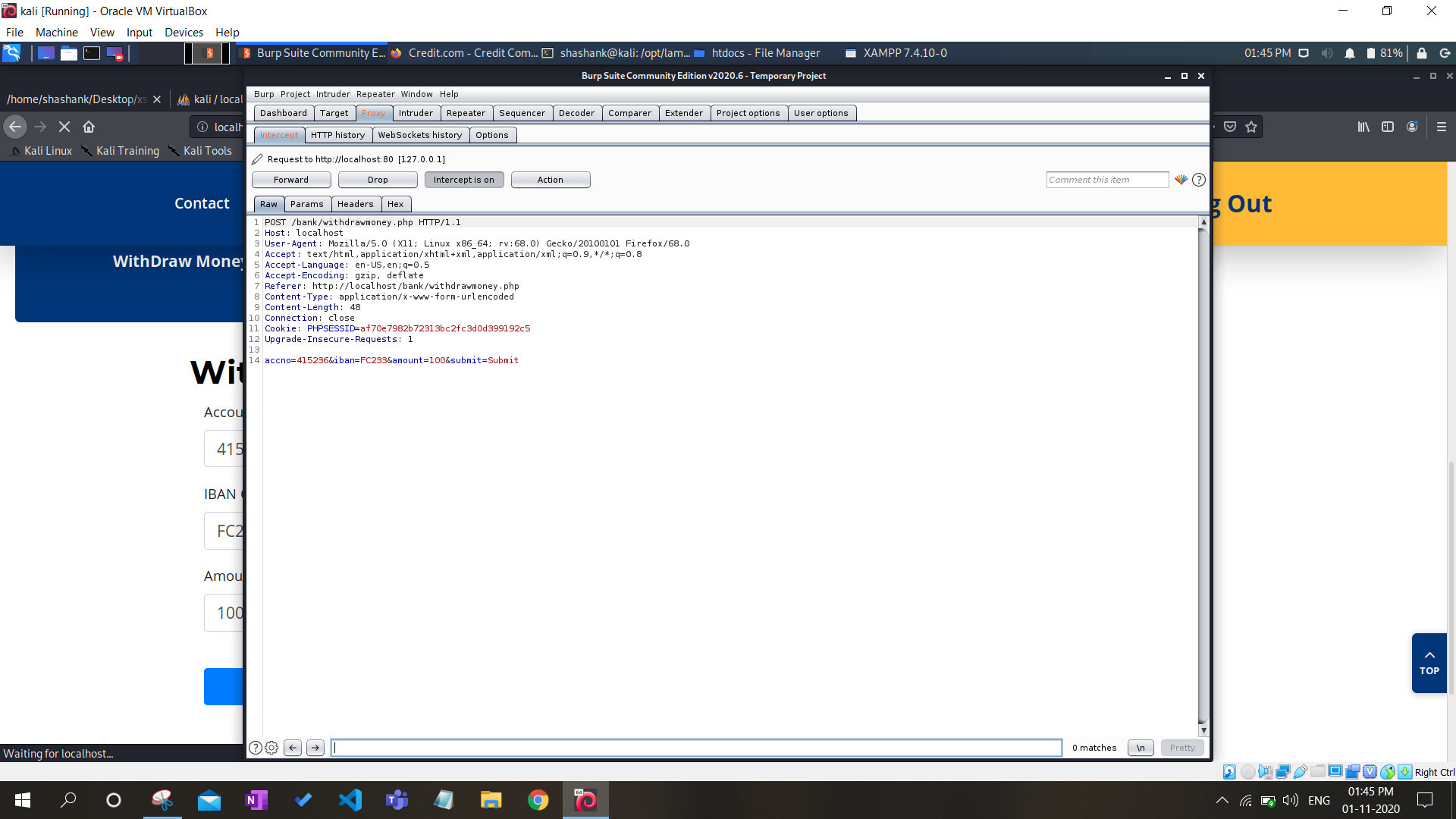


**Detection and Prevention using BurpSuite-**

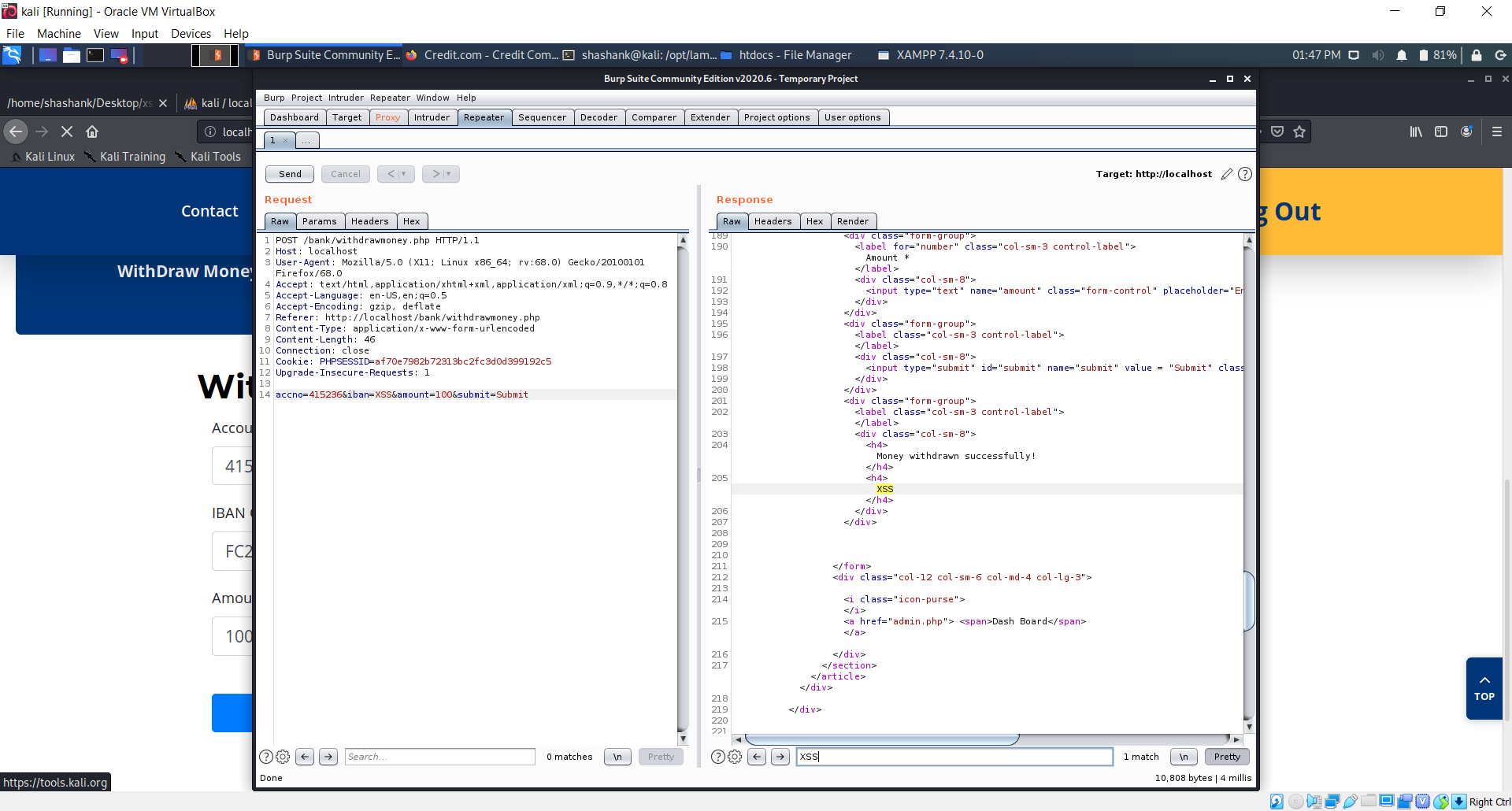
Checking XSS vulnerability on web page-



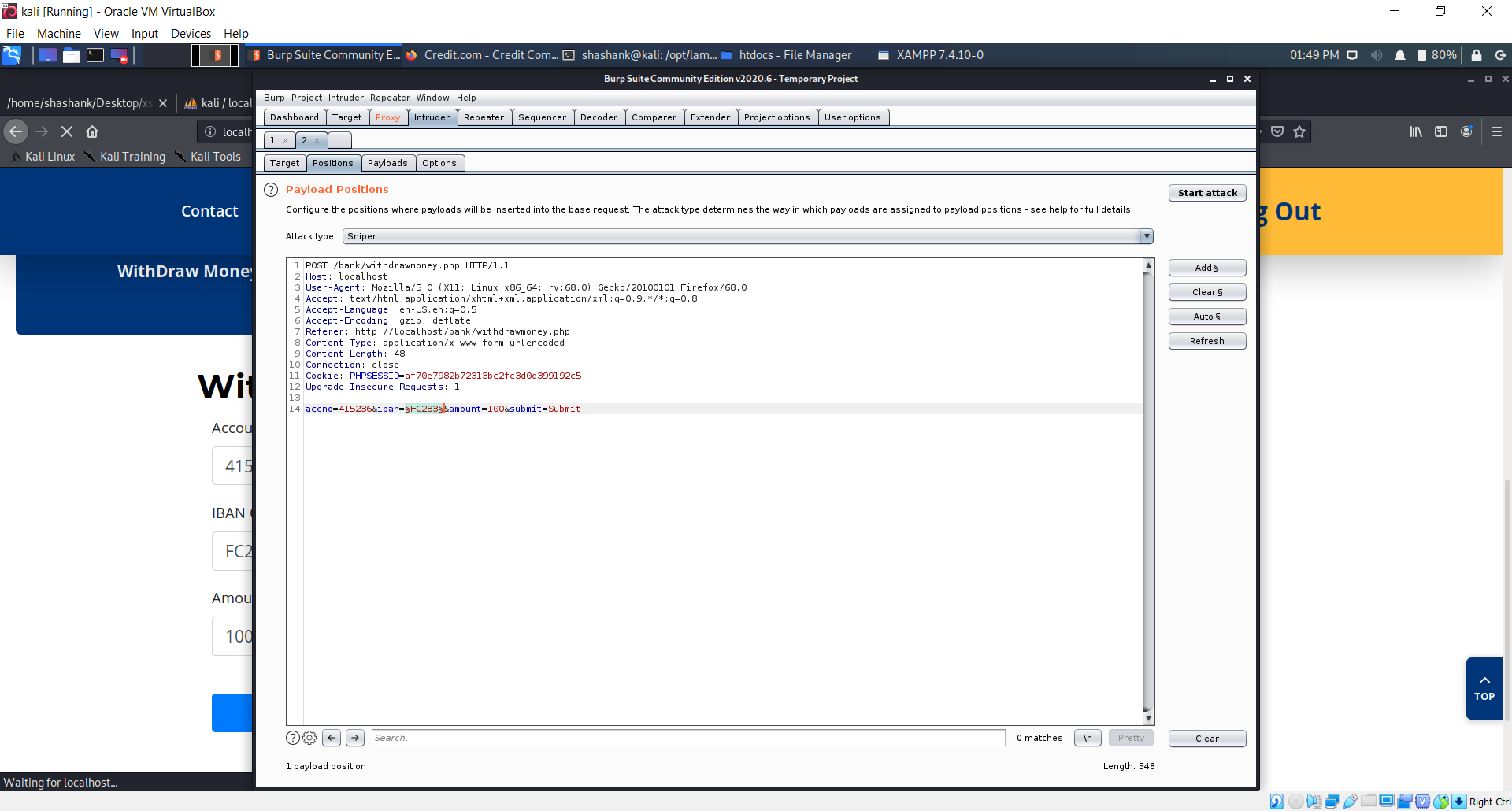
Intercepting request on BurpSuite-



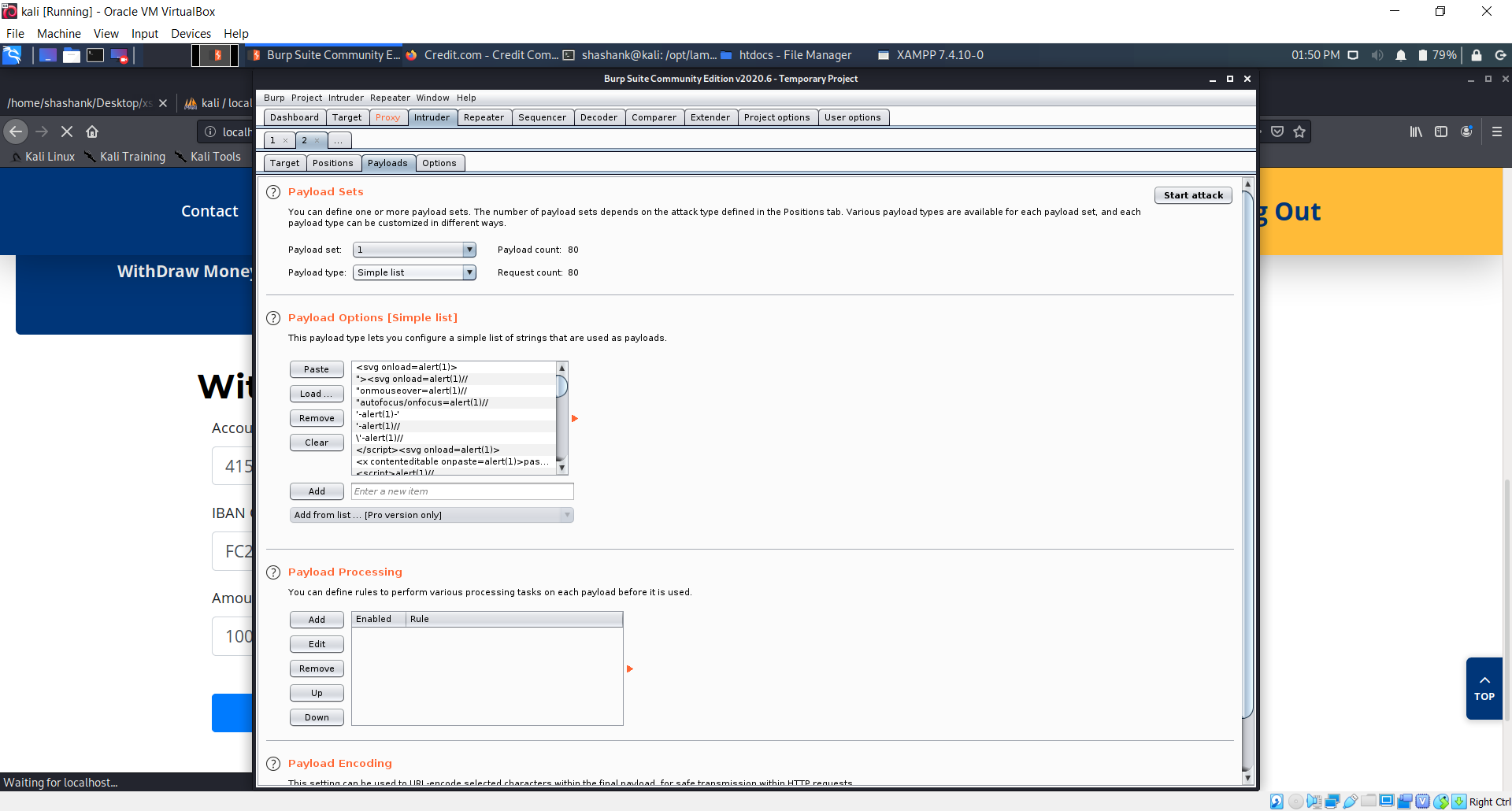
Found XSS vulnerability-



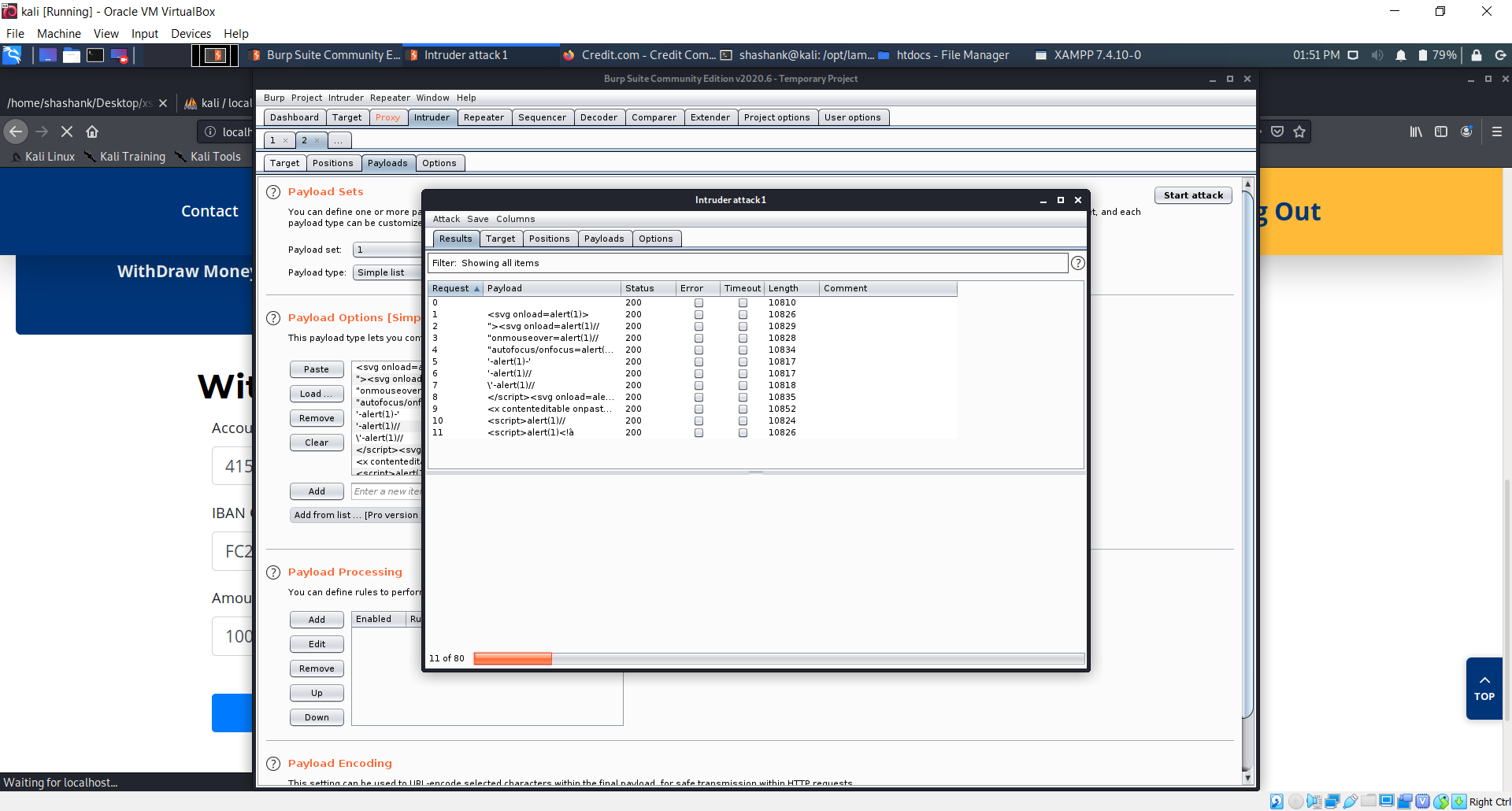
Sending request to intruder to check all possible XSS attacks-



Entered some payloads or malicious scripts-



Attack started and status 200 shows attack successful-



So this is how we detect that which form entry is vulnerable, then to prevent it from future attacks, we will just have to remove that line of code from javascript which returns same output to web page as entered into the page.

We will remove following highligted code for prevention-

