CYBR 8410 – Distributed System Security – 2021 Spring

Lab 4

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NUID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Please follow the requirements and due date specified in the Syllabus to submit your work.

**Lab Description**:

Students will identify two types of Web attacks in this Lab. One is Command Injection attacks, and the other is Cross Site Scripting (XSS) attacks. You need to answer the specified questions in the following lab content and steps, and submit your answers in a PDF or Word file.

During the lab, if anything is not clear to you, please contact the instructor.

**Lab Content and Steps:**

1. Reuse your virtual machine created in previous labs.
2. Command Injection Attack
   1. Start your virtual machine and the Apache 2 Web server.

Question 1: (5 pts)

1. What are the users and groups pre‐created on your VM? (1 pts)
2. What is the main reason (in terms of security) for creating so many users and groups? (1 pts)
3. What are the running user and running group of those Apache 2 processes? (1 pts)
4. Which Apache 2 configurations are used to specify the running user and running group? (1 pts)
5. Why such configurations are used? (1 pts)

* 1. Host the extremeinsecure Web application on your Apache Web server by

1. Copy “extremeinsecure.tar.gz” to your VM
2. Extract the .tar.gz file to the appropriate directory of the Apache 2 Web server
3. Properly set the permissions of the extracted directory and files so that the nine .htm and one .php webpages in the “extremeinsecure” directory can be properly accessed by anyone through a Web browser, e.g., by typing: <http://192.168.65.128/extremeinsecure/> in which that IP address is for my VM.

Question 2: (5 pts)

1. What are the permissions that you have assigned to the “extremeinsecure” directory and the files in it (you can run “ls -l” and copy the results from the screen)?
   1. Identify and verify the command injection vulnerability in this Web application.

Question 3: (15 pts)

1. What is that vulnerability and how did you figure it out? (4 pts)
2. What are the risks due to this vulnerability? (Hint: you need to identify the vulnerability by either trying this Web application, doing a source code inspection, or running a third‐party Web application vulnerability scanning tool). (5 pts)
3. Demonstrate this vulnerability by using at least three attack examples. You need to describe the attack procedure, input, and the corresponding output in detail for each example. (6 pts)
   1. Perform a special attack by using the following steps:
4. Create a directory named confidential under the /var directory.
5. Create a file named bankinfo.txt in the confidential directory contains such a sentence:

*I have $1,00,000 in my bank account. I am so happy because NO ONE can know this.*

1. Change the owner of the confidential directory and bankinfo.txt to the current running user of those Apache 2 processes. (Hint: use chown command)
2. Set the permission of bankinfo.txt so that it is only readable and writable by the owner. Other users do not have any access right to this file.
3. Set the permission of the confidential directory so that it is only readable, writable, and executable by the owner. Other users do not have any access right to this directory.

Question 4: (15 pts)

1. What are the permissions that you have assigned to the confidential directory and the bankinfo.txt file (you can run ls -l and copy the results from the screen)? (1 pts)
2. Demonstrate that you can take advantage of the above‐identified vulnerability to read the content of bankinfo.txt by only using a browser on any computer. You need to provide the details about the attack procedure, input, and the corresponding output. (4 pts)
3. Demonstrate that you can take advantage of the above‐identified vulnerability to change the content of bankinfo.txt (still only using a browser on any computer) to:

*I have $2,00,000 in my bank account. I am so happy because NO ONE can know this.*

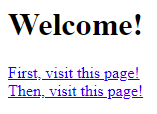
That is, only the amount of the money is doubled. The size of the bankinfo.txt file does not change. You need to provide the details about the attack procedure, input, and the corresponding output. (5 pts)

1. Analyze why such a confidential file outside of the Apache www directory can be read and modified. (5 pts)
   1. Propose a solution to fix the command injection vulnerability in this Web application. Also propose a solution to further reduce the potential risks that can be introduced by the Apache server to this Linux system (Hint: for this solution, you can either consider enforcing file create/access policies or consider changing Apache configurations).

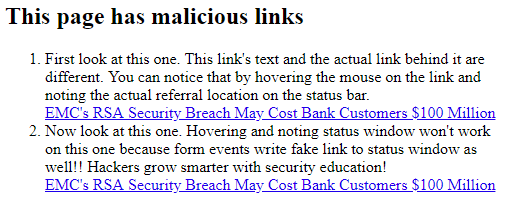
Question 5: (15 pts)

1. Describe your solutions and verifications.

1. Cross Site Scripting (XSS) attack
   1. Host the XSS Web application on your Apache Web server by
2. Copy “XSS.tar.gz” to your VM
3. Extract the .tar.gz file to the appropriate directory of the Apache 2 Web server
4. Properly set the permissions of the extracted directory and files so that webpages in the XSS directory can be properly accessed by anyone through a Web browser, e.g., by typing: <http://192.168.65.128/XSS/> in which that IP address is for my VM.
   1. Visit the XSS Web application using a browser
5. You will see two links as shown below. click the “First, visit this page!” link to visit setgetcookie.htm.



1. On setgetcookie.html, fill the username/password form, click “Set cookie” and then “Show cookie” buttons to make sure this website set a cookie to your browser.
2. Go back and click the “Then, visit this page!” link to visit malURL.htm. You will see two links on this malURL.htm page as shown below, but both of them do not work right now.



1. Change the malURL.htm of your XSS Web application, so that when you click the FIRST “EMC's RSA Security Breach May Cost Bank Customers $100 Million” link, the cookie stored on your browser for this website will be sent to an attacker’s server (that’s my VM) using an HTTP GET request with the following format:

<http://192.168.65.128/XSS/stealcookie.php?info=COOKIES>

Note: you need to use the above address and format with your own IP address but not mine, so that the stealcookie.php on your VM can receive and verify the submission record. Here, COOKIES means the complete cookie content save on your browser for this website (i.e., document.cookie), so it should include a pair of username and password values. The file stealcookie.php is also included in XSS.tar.gz.

Hint: there could be multiple solutions. One solution is to simply replace the current href value ("#") of the first link in malURL.htm with a piece of JavaScript code; nothing else needs to be changed. Note that if you use this solution and if you move the mouse over the link, you may notice that my VM address information is shown on the status bar of your browser. So, a security conscious user may notice this and may not be tricked to click the link.

Question 6: (15 pts)

1. What are your changes to the malURL.htm file? What are your verification results? Explain this XSS attack.
2. Make the attack stealthier, so that when you move the mouse over the second link, my VM address information will not be shown on the status bar of your browser. In this case, a security conscious user may also be tricked to click the second link. Still, when you click this second “EMC's RSA Security Breach May Cost Bank Customers $100 Million” link, the cookie stored on your browser for this website will be sent to an attacker’s server (that’s my VM) using an HTTP GET request with the following format:

<http://192.168.65.128/XSS/stealcookie.php?info=COOKIES>

Note: you need to use the above address and format with your own IP address but not mine just like what you did in the last question.

Hint: there could be multiple solutions. One simple solution is to simply add a

redirectpage.htm webpage in the XSS directory; you don’t need to change malURL.htm

because redirectpage.htm has been specified in it.

Question 7: (15 points)

1. Describe the details of your solution. If you use the redirectpage.htm solution, please provide the content of this file. What are your verification results?
2. Basically, the above two links illustrated the persistent (or stored) XSS attacks. That is, if an attack can store those links on your legitimate webpages or websites, they can perform attacks such as cookie stealing. This XSS Web application is also vulnerable to non-persistent (or reflected) XSS attacks. In this task, demonstrate that the above cookie stealing attacks can happen by using reflected XSS attacks. In other words, reflected XSS attacks can also send the cookie stored on your browser for this website to an attacker’s server (that’s my VM) using an HTTP GET request with the following format:

<http://192.168.65.128/XSS/stealcookie.php?info=COOKIES>

Note: you need to use the above address and format with your own IP address but not mine just like what you did in the last question.

Hint: you don’t need to make any change to this existing XSS Web application. You simply need to demonstrate the vulnerability by cleverly constructing input to this XSS Web application. Probably Google Chrome browser will detect your XSS attacks, then you will need to restart your Google Chrome browser with a specific flag or simply change to another browser such as Firefox.

Question 8: (15 points)

1. Describe the details of your attacks (including the input you used). What are your verification results?