Homework 3: Exploits

Due date & time: Due at 23:59 on April 5, 2023. This is a firm deadline. This project MUST be finished independently.

1. Introduction

It's time to show your web security skills. In this assignment, you will have a bunch of web functionalities to attack. Your mission is to find any 10 out of 12 categories of bugs we planted. Perhaps you may be able to find more than that.

More specifically, you are asked to locate bugs in our self-crafted web applications according to the categories of vulnerabilities listed in Table A. You are expected to find as many bugs as possible. Totally, there are 12 cases in our web application, each of which contains at least one bug. Some of the cases may have more than one bug. Just choose either one for mentioned bug category to exploit. If you claim and exploit more than one bug in a case, only one will be graded.

2. Environment

The VM can be downloaded from the following link.
https://drive.google.com/file/d/1JRa-cdbm92OuuIjrZnekRnMhItEKc79q/view?usp=share_link

2.1. VM information

The information for the VM is as follows:

- **OS account**:
  username: student
  password: student
- **Homepage (need to be accessed from the VM)**: http://www.wsb.com/Homework3
- **Source code**: `/var/www/html/Homework3/`
- **Database**: http://www.wsb.com/phpmyadmin
  - Management account:
    - username: root
    - password: student
  - Developing account:
    - username: cs5331
    - password: v29AcujVSDKWadxe

If you need to access the database, you are recommended to use root and access via phpMyadmin (http://www.phpmyadmin.net/) or other MySQL clients.

3. What to exploit

You will select to exploit 12 cases: case01-case12. Each case contains at least one bug. Table A shows the bug categories and your target for each bug. Note that the sequence of provided cases is not necessarily consistent with that of bugs. For example, case01 may not correspond to the first bug category, Reflected XSS.

A "flag" is a secret we embedded in the vulnerable application, which you need to find and output it using the exploit.

Table A. Bugs in our web application

No.	Bug Category	Target
1	Reflected XSS	Flag in cookie
2	PHP Injection	Flag in variable $flag
3	Remote Code Execution	cat /etc/passwd remotely
4	SQL Injection	Flag in database
5	Poor Password Management	Flag is the password of the user admin
6	Local File Inclusion	Flag in lfi.txt.php at Assigement3 directory
7	Cross-Origin Flaws	Flag will appear when coming from "authorized" origin
8	CSRF Predictable tokens	prepare a script with a simple form to submit a message to case09.php
9	Execution after Redirect	Flag output by server
10	Logic Authentication Flaw	Escalate your privilege to get the flag
11	Parameter Pollution	Flag on webpage when polluted
12	Unvalidated Redirect	Exploit url and redirect to www.google.com

4. Grading criteria

For each bug in Table A, to report and claim points, you will have to:

- identify the correct bug category for each case.
- show that the bug is exploitable by providing an attack script (a bash script) which will automatically exploit the bug. We already specify the target in each category in Table A, e.g., alert the flag, echo the etc/passwd, etc.
- Provide analysis for the bugs and explanation for your attack script.

A sample of the attacking script is in the directory `/var/www/html/Homework3/sample`. Please take a look to see how to structure yours.

For every bug reported, the TAs will execute your corresponding bash script and check if it achieves the target stated in Table A. When you create the script for attacks, you should not make any assumptions about the TA's prior knowledge. We simply execute your bash script and check whether you successfully achieve the target. We will check the validity of the script as well to see if you craft your attack correctly, e.g., you cannot manually copy the cookie from the browser and put it in an alert box.

You are only allowed to find bugs and create exploit for the mentioned categories without utilizing any other vulnerabilities in the system. For example, bugs pertaining to VM's user password guessing/cracking or using pre-configured secrets in the VM (e.g., SSL private keys, root passwords) are not considered in this assignment. Apache infrastructure exploits, browser compromise, etc., are all out-of-scope. Please focus on web application vulnerabilities instead of the underlying server-browser infrastructure.

5. Submission instruction

Your submission file (e.g. e19930314.zip) to Canvas should include:
1. Setup script, `setup.sh`. This script will install the packages used by your solution.
2. Exploit scripts (e.g., `exploit04.sh`, `exploit06.sh`, `exploit09.sh`, etc.)
At the beginning of each script, you need to include comments to specify the category of the vulnerability and how you identify the exploit. For example, you can use the following introduction to describe the exploit to the sample:

```
# Bug Category: Persistent XSS

# The logic of the sample.php is:
# 1. the user input their title and content at line 70
and 75 in sample.php
# 2. the php script will store it into the database at
line 18 in sample.php
# 3. the php script will retrieve the content from the
database and display to the user at line 51 in sample.php

# Exploit:
# We can see that there is no sanitization function for
the user input, so a user can insert javascript code into
the database and the next time when the web server
retrieve the content from the databsae and show it to the
user, the javascript will be executed and leading to a
persistent xss attack.
```

During grading, we will first run `setup.sh`. We then run the exploit scripts to check the results. Please explain your solution using the comments. Exploit scripts without sufficient comments will not receive full marks.

This assignment accounts for 15 marks of the final grade. You will select 10 out of the 12 marks to answer. The detailed breakdown for this assignment is:
1. Recognize the bug categories for 10 cases (5 marks)
2. Provide a prove-of-concept for bugs in 10 cases (10 marks)

6. Common questions

1. I found a bug and I was able to exploit the bug with an attack. However, I'm not able to automate the attack. Would it be possible for me to get the point?

Unfortunately, it is not. We are rather strict with grading in this assignment. Full marks will be given only if complete automation to launch an attack works. You will get the marks for bug category if you answer the bug category correctly in the comments. So even if the code may not fully work, you still need to submit the script and answer the category as comments.

2. Can we use any third-party libraries in the scripts?

Yes. Remember that we will run your script on a vanilla VM which possibly doesn't have the libraries you need. Therefore, you can set up the environment and install the dependencies in the setup script. We will run it before running the exploit scripts.

3. Can we assign one case to multiple different categories?

No, you can't. One case can be assigned to only one category.

4. Do all the vulnerabilities have to be exploitable?

Yes, all reported bugs have to be exploitable. You should verify it by a proof-of-concept attack.

5. How do you submit HTTP requests using script?

In our sample example, we use urllib2 library in Python to send requests. You are free to use any other library you prefer to perform the request submission.