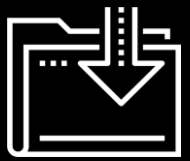
File Upload Vulnerabilities

Cybersecurity

Web Vulnerabilities 2 Day 2



Today’s Objectives

By the end of class, you will be able to:

- ☐ Test for LFI/RFI vulnerabilities
- ☐ Exfiltrate data from a server with webshells

Activity: SQL Injection Warm-UP

In this activity, you will review concepts covered last class about SQL injection.

Instructions sent via Slack.

Suggested Time:
15 Minutes

SQL Injection Warm-Up Review

Consider the following URL: `https://vulnerable.site?id=1`

How would you test this URL for a SQL injection vulnerability?

SQL Injection Warm-Up Review

Consider the following URL: https://vulnerable.site?id=1

How would you test this URL for a SQL injection vulnerability?

Start by injecting quotation marks (‘ and “) . If this produces errors, the server may be vulnerable to injection.

If you suspect there's a vulnerability, you should try an OR and AND injection.

SQL Injection Warm-Up Review

Consider the following URLs:

- `https://vulnerable.site?id=1%20AND%201=1`
- `https://vulnerable.site?id=1%20AND%201=2`

If the first URL works and the second causes an error, what can you conclude?

SQL Injection Warm-Up Review

Consider the following URLs:

- `https://vulnerable.site?id=1%20AND%201=1`
- `https://vulnerable.site?id=1%20AND%201=2`

If the first URL works and the second causes an error, what can you conclude?

If the first URL works and the second causes an error, we can conclude that the database ran the full SQL statement in the URL.

If the AND comparison is true, it will work as normal. If it is false, there will be an error. Only servers that are vulnerable to injection will work this way.

SQL Injection Warm-Up Review

Fill out the query below to select `login` and `password` information from `users`.

```
SELECT name, age FROM users
UNION
-- Your SQL here
```

SQL Injection Warm-Up Review

Fill out the query below to select `login` and `password` information from `users`.

```
SELECT name, age FROM users
UNION
SELECT login, password FROM users
```

Uploading Malicious Files

Uploading Malicious Files

Today we'll explore **uploading files** in order to turn an administrator's web server into a tool for running commands on the host.

- The file upload functionality we'll be looking at is designed for the site administrator to upload pictures, but we can use it to upload arbitrary files, including code.

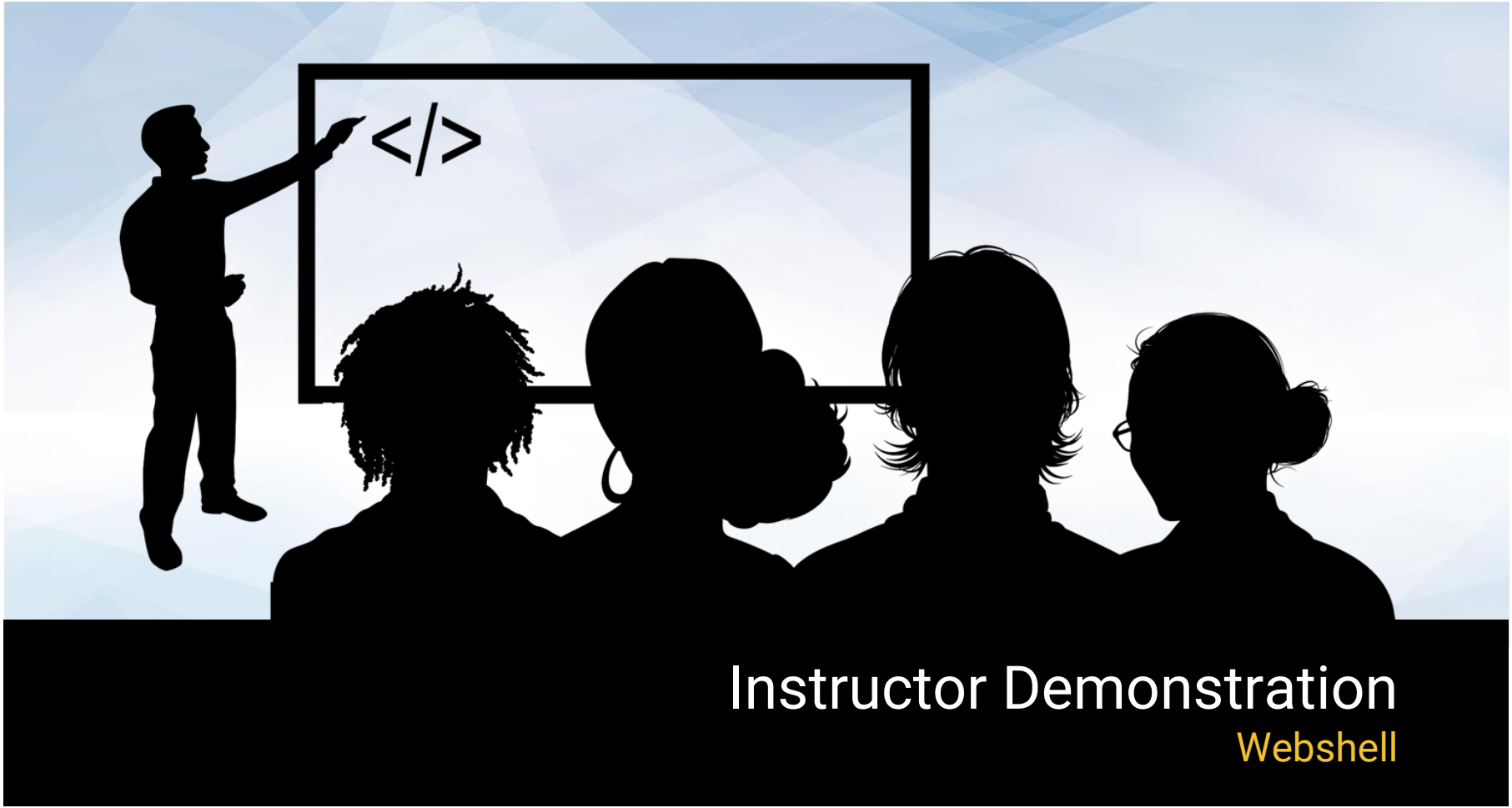
- We'll use a tool called **Webshell** to upload malicious PHP that can execute shell commands.

Webshells

Webshells allow attackers to run shell commands by sending them as parameters in HTTP requests.

Then, the script on the server parses the parameters, and runs them as code.

- For example, to use a webshell, an attacker might navigate to: `http://example.com/webshell.php?command=ls`.
  - The script would read the command parameter, whose value is `ls`.
  - The script would run `ls` as a shell command.
  - The HTTP response would contain the results normally printed to the command line.



Instructor Demonstration

Webshell

Activity: My First Webshell

In this activity, you will examine the form and function of a PHP webshell, upload a webshell to a vulnerable server, and use Burp Suite to perform reconnaissance against the compromised host.

Activities/Stu_My_First_Webshell/README

Suggested Time:

My First Webshell Review

First Steps with Webshell

While we do not need to know PHP in order to assess web applications, it is useful to be familiar with:

- How PHP is inserted into web pages.
- How user data is extracted from an HTTP request.
- How PHP runs shell commands from server side scripts.

My First Webshell Review

Open webshell.php and answer the following questions:

Look at the beginning and end of the file. How do you start/end a PHP script?

What does the variable $_GET refer to

Consider the URL: https://example.com?parameter=value. How would you get the value param in PHP?

Consider the following POST body: username=hacker&password=null. How would you get the value of username in PHP? What about password?

My First Webshell Review

Open webshell.php and answer the following questions:

Look at the beginning and end of the file. How do you start/end a PHP script?

PHP files look like: `<?php /* PHP Code Here */ ?>`. They start with <?php, and end with ?>.

What does the variable $_GET refer to

Consider the URL: https://example.com?parameter=value. How would you get the value param in PHP?

Consider the following POST body: username=hacker&password=null. How would you get the value of username in PHP? What about password?

My First Webshell Review

Open webshell.php and answer the following questions:

Look at the beginning and end of the file. How do you start/end a PHP script?

PHP files look like: `<?php /* PHP Code Here */ ?>`. They start with <?php, and end with ?>.

What does the variable $_GET refer to

The variable $_GET refers to the query string of an HTTP GET request.

Consider the URL: https://example.com?parameter=value. How would you get the value param in PHP?

Consider the following POST body: username=hacker&password=null. How would you get the value of username in PHP? What about password?

My First Webshell Review

Open webshell.php and answer the following questions:

Look at the beginning and end of the file. How do you start/end a PHP script?

PHP files look like: `<?php /* PHP Code Here */ ?>`. They start with <?php, and end with ?>.

What does the variable $_GET refer to

The variable $_GET refers to the query string of an HTTP GET request.

Consider the URL: https://example.com?parameter=value. How would you get the value param in PHP?

Use: `$_GET['param']`

Consider the following POST body: username=hacker&password=null. How would you get the value of username in PHP? What about password?

My First Webshell Review

Open webshell.php and answer the following questions:

Look at the beginning and end of the file. How do you start/end a PHP script?

PHP files look like: `<?php /* PHP Code Here */ ?>`. They start with <?php, and end with ?>.

What does the variable $_GET refer to

The variable $_GET refers to the query string of an HTTP GET request.

Consider the URL: https://example.com?parameter=value. How would you get the value param in PHP?

Use: `$_GET['param']`

Consider the following POST body: username=hacker&password=null. How would you get the value of username in PHP? What about password?

Use: `$_POST['username']` and `$_POST['password']`

My First Webshell Review

Use Repeater to extract the following info from the server:

- Run `id` and `whoami`
- Determine the operating system and kernel version with `uname`
- Read the contents of `/etc/passwd`
- List all running processes with `ps aux`
- Note: Recall that you must use a + or `%20` to send a space in a URL, e.g.: `ls -sail` becomes `ls+-sail` *or* `ls%20-sail`.

My First Webshell Review

Use Repeater to extract the following info from the server:

- Run `id` and `whoami`
- Determine the operating system and kernel version with `uname`
- Read the contents of `/etc/passwd`
- List all running processes with `ps aux`
- Note: Recall that you must use a + or `%20` to send a space in a URL, e.g.: `ls -sail` becomes `ls+-sail` *or* `ls%20-sail`.

To dump `/etc/passwd`: `GET /hackable/webshell.php?cmd=cat%20/etc/passwd`

For user info: `id`: `GET /hackable/webshell.php?cmd=id`

For OS info: `uname`: `GET /hackable/webshell.php?cmd=uname`

For process info: `ps aux`: `GET /hackable/webshell.php?cmd=ps%20aux`

My First Webshell Review

As a bonus, repeat the previous exercise with Burp Intruder instead of Repeater.

My First Webshell Review

As a bonus, repeat the previous exercise with Burp Intruder instead of Repeater.

Send the request from Repeater to Intruder by pressing Ctrl + I.

Add a position such that your request line looks like:

```
GET /hackable/webshell.php?cmd=$COMMAND$.
```

Add the commands you ran above as payloads in Intruder, and click **Start Attack**.

Receiving Requests with webhook

Activity: Serving an XSS Vector

In this activity, you will assess a site for XSS vulnerabilities.

Activities/Stu_Cookie_Phishing/README

Suggested Time:
20 Minutes

Today’s Objectives

By the end of class, you will be able to:

- Test for LFI/RFI vulnerabilities.
- Exfiltrate data from a server with webshells.