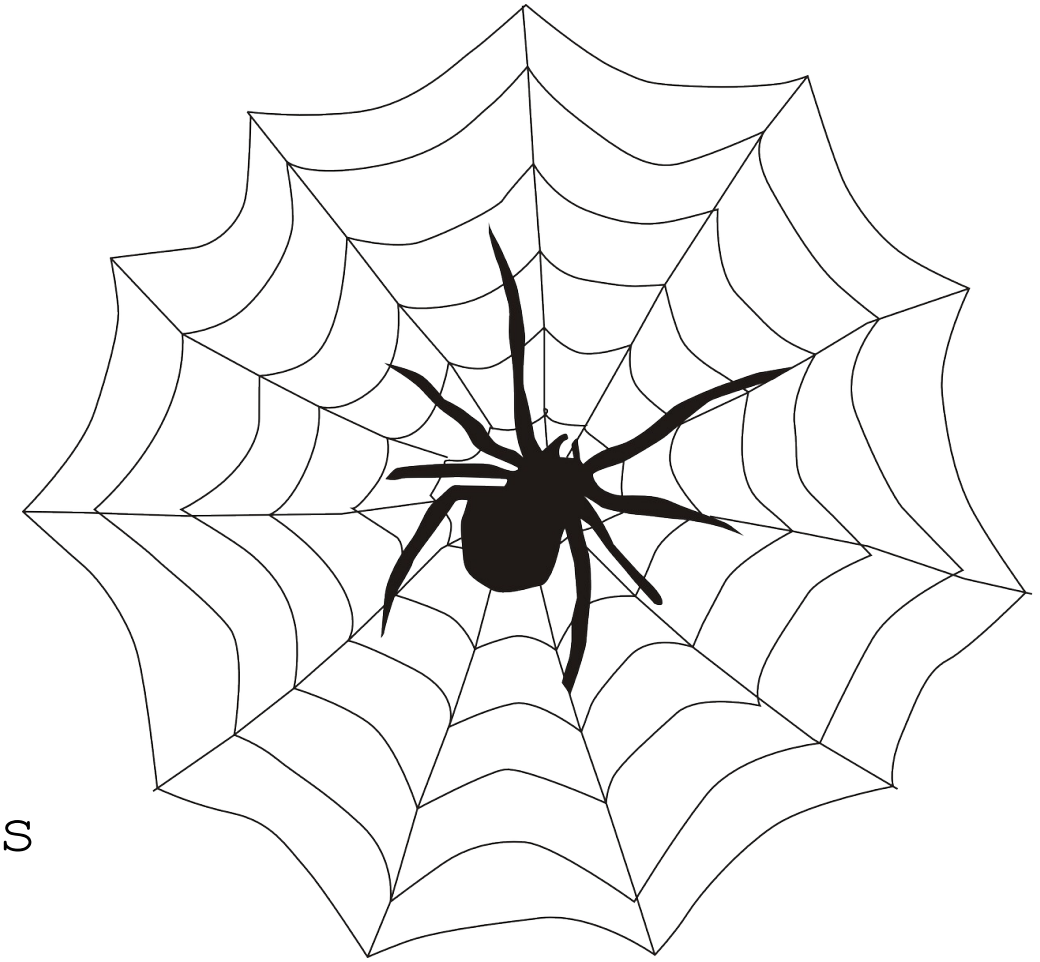
# HackerFrogs Afterschool
# Web App Hacking Basics: Part 3

Class:
Web App Hacking

Workshop Number:
AS-WEB-03

Document Version:
1.5

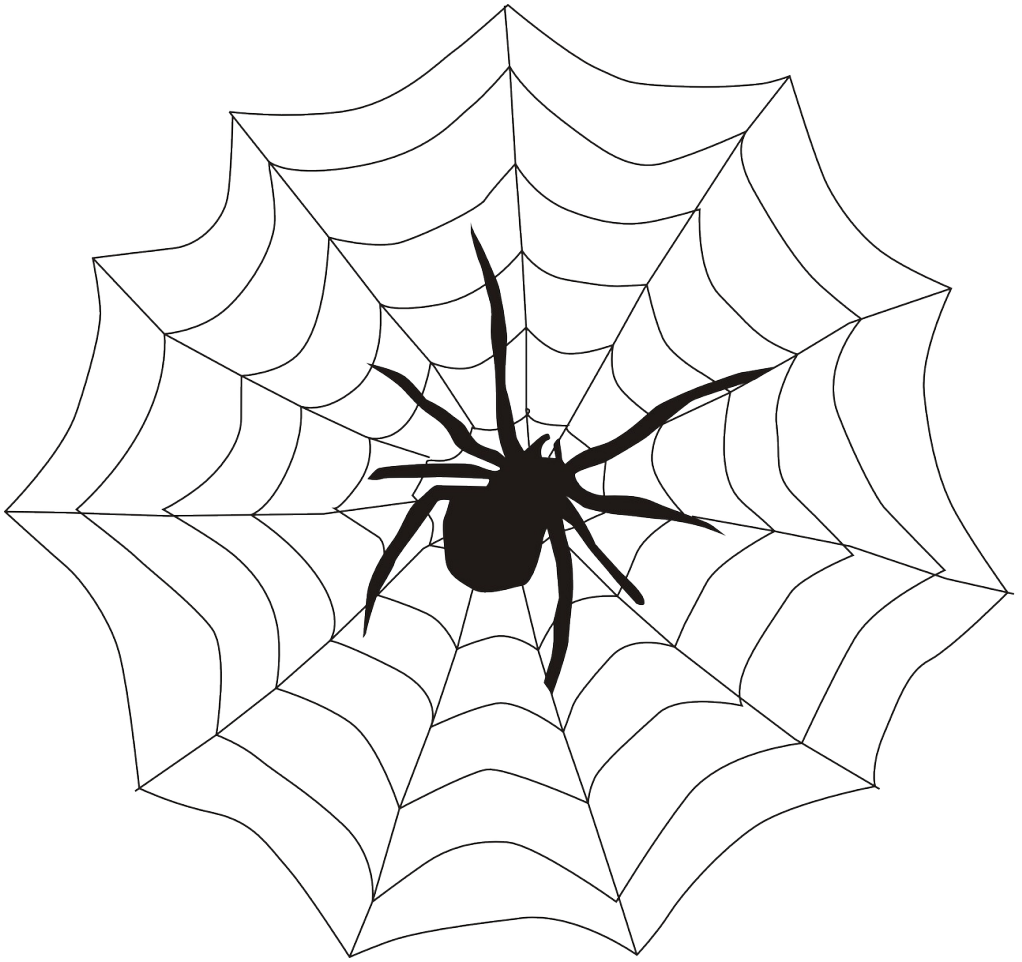
Special Requirements:
Completion of previous workshop, AS-WEB-02



# What We Learned In The Previous Workshop

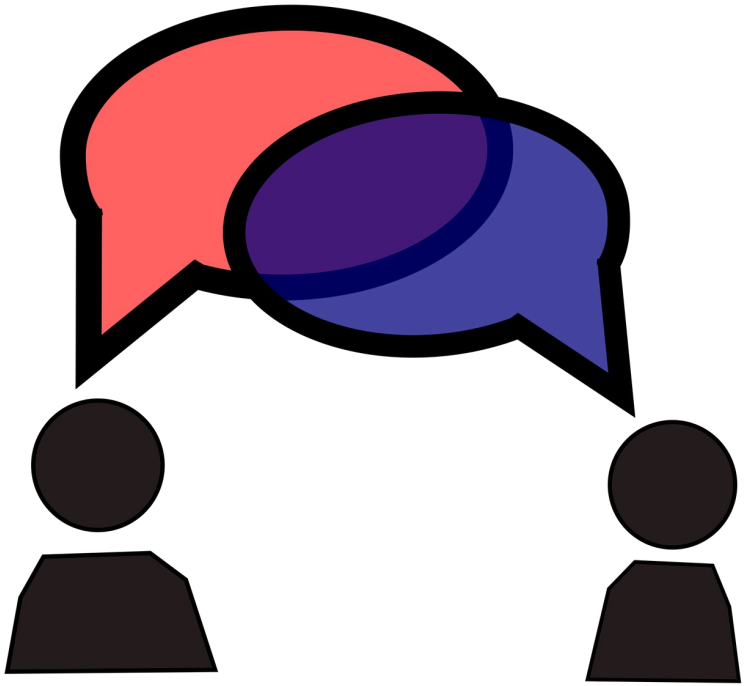
This is the third intro to web app hacking workshop.

In the previous workshop we learned about the following web app hacking concepts:



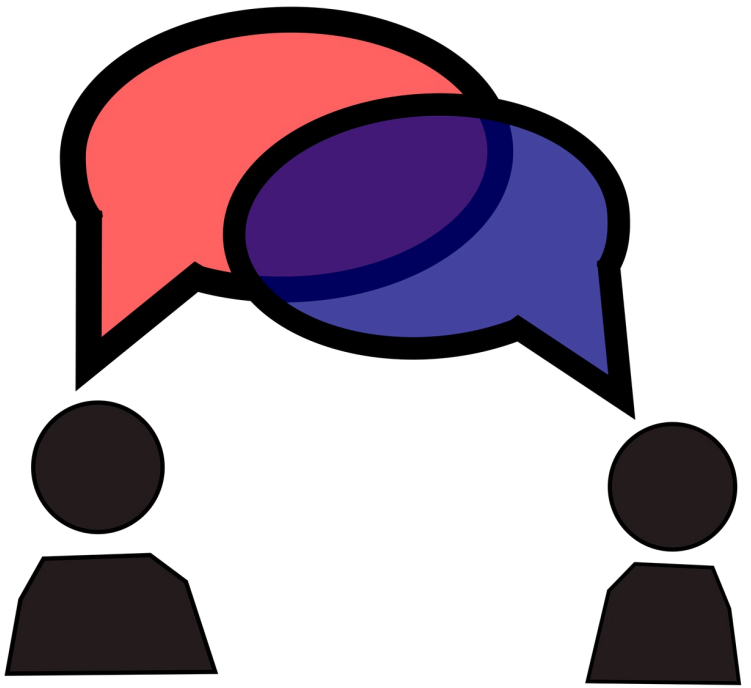
# HTTP Referer Header

The HTTP Referer header (which is misspelled on purpose) contains the value of a complete or partial address of the webpage that is making the request. It's often used for analytics or logging.



# HTTP Referer Header

The HTTP Referer header is occasionally used by web app developers as a security mechanism, but this is not a good practice, as it was never intended to be used in such a way



# HTTP Cookie Header

HTTP Cookies are commonly used as a security mechanism that allows users to maintain an authenticated user session on a website.

# HTTP Cookie Header

In short, HTTP cookies allow users to “login” to websites

# HTTP Cookie Header

But since cookie values can be modified by the user, proper care should be taken to ensure that valid cookie values are not predictable or easily guessed

# Let's Continue Where We Left Off!

Let's pick up where we left off in the Natas CTF:

http://natas6.natas.labs.overthewire.org/

# URL Parameters


URL parameters are variables attached to the end of URLs. They can be identified by the ? (question mark) directly after the webpage or directory name, followed by the parameter key name, then the = (equals) sign, then the value.

# URL Parameters


If there are multiple parameters included in the same URL, then they are separated by the & (ampersand) symbol.

# URL Parameters Use Cases


There are a few different reasons why webpages use URL parameters. The most common one is for search queries.

# URL Parameters Use Cases


However, another common, and potentially dangerous use of URL parameters is to instruct the webserver on which webpage to display.

# URL Parameters Use Cases


The use of URL parameters which reference other files on the webserver could potentially be exploited in an attack called Local File Inclusion (LFI).

# Local File Inclusion

Local File Inclusion (LFI) is a web app vulnerability where arbitrary local webserver files can be accessed through a web interface.

# Local File Inclusion

LFI vulnerabilities can lead to sensitive data exposure, and can also be used as the first step in a chain of exploits.

# Local File Inclusion


The inclusion of file names in URL parameters is a typical method through which a potential LFI vulnerability is identified.

# Local File Inclusion: Filesystem Structure

Each **../** indicates an elevation of one level in the filesystem, traveling from the web app's working directory ( /natas7 ) up to the top-level directory ( / )

/
/var
/var/html
/var/html/labs
/var/html/labs/natas
/var/html/labs/natas/natas7

# Local File Inclusion: Filesystem Structure

From the top-level directory, we can provide a filepath to the file we want to access.

A typical test file for LFI on Linux / Unix webservers is the **/etc/passwd** file, since it is publicly readable by default, and gives info regarding usernames on the webserver.

# P-8 Sourcecode Analysis: Reverse Operations

**Original Operation** **Reversed Operation**

# P-8 Sourcecode Analysis: Reverse Operations

| **Original Operation** | **Reversed Operation** |
| --- | --- |
| bin2hex | |

# P-8 Sourcecode Analysis: Reverse Operations

| **Original Operation** | **Reversed Operation** |
| --- | --- |
| bin2hex | hex2bin |

# P-8 Sourcecode Analysis: Reverse Operations

| **Original Operation** | **Reversed Operation** |
| --- | --- |
| bin2hex | hex2bin |
| strrev | |

# P-8 Sourcecode Analysis: Reverse Operations

| **Original Operation** | **Reversed Operation** |
| --- | --- |
| bin2hex | hex2bin |
| strrev | strrev |

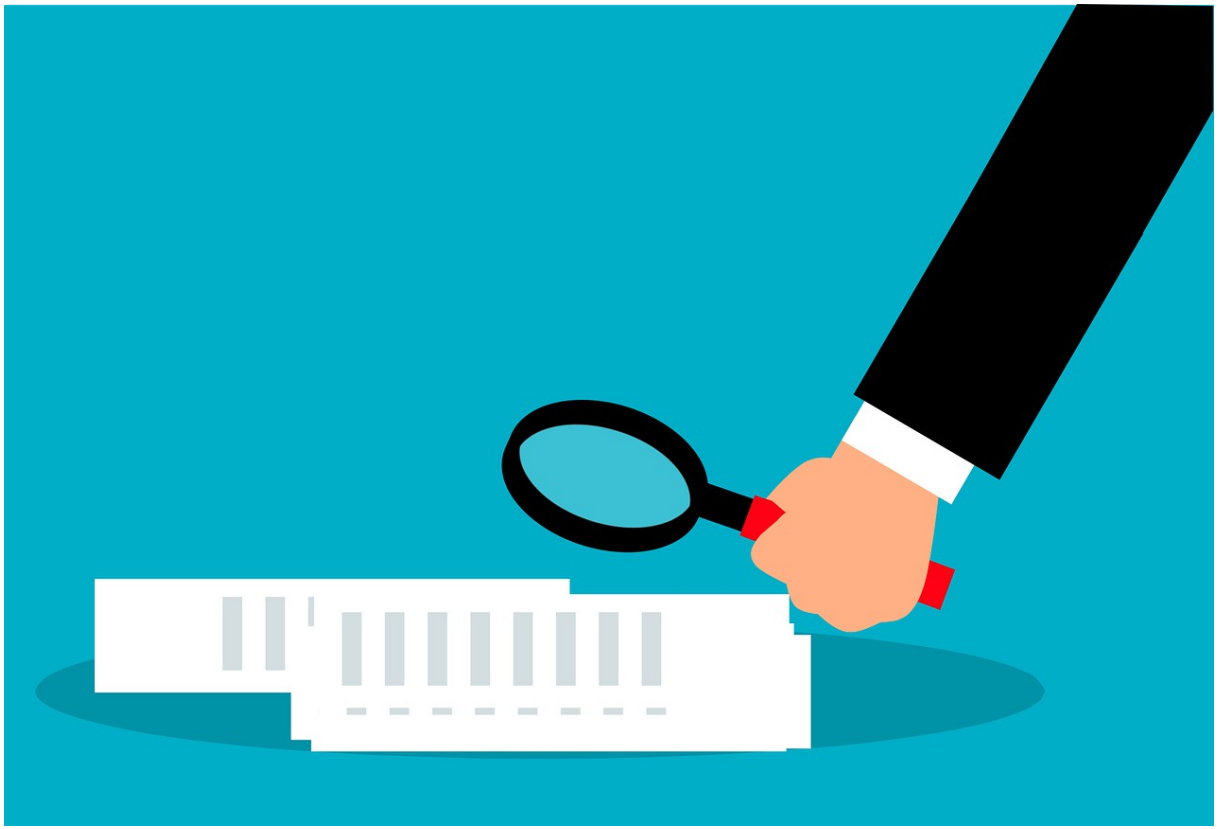
# P-8 Sourcecode Analysis: Reverse Operations

| **Original Operation** | **Reversed Operation** |
| --- | --- |
| bin2hex | hex2bin |
| strrev | strrev |
| base64_encode | |

# P-8 Sourcecode Analysis: Reverse Operations

| **Original Operation** | **Reversed Operation** |
| --- | --- |
| bin2hex | hex2bin |
| strrev | strrev |
| base64_encode | base64_decode |

# Summary



Let's review the web exploitation concepts we learned in this workshop:

# Sourcecode Analysis

Sourcecode analysis is the process of analyzing the code of a piece of software with the goal of deeper understanding regarding its function.

```
<?

include "includes/secret.inc";

    if(array_key_exists("submit", $_POST)) {
        if($secret == $_POST['secret']) {
        print "Access granted. The password for
    } else {
        print "Wrong secret";
    }
    }
?>
```

# Sourcecode Analysis

It is a common technique used in software security testing, but it requires the software sourcecode be readily available.

```
<?

include "includes/secret.inc";

    if(array_key_exists("submit", $_POST)) {
        if($secret == $_POST['secret']) {
        print "Access granted. The password for
    } else {
        print "Wrong secret";
    }
    }
?>
```

# URL Parameters


URL parameters are variables attached to the end of URLs. They are often used to send search queries to webservers, but they can be used to retrieve other data as well.

# Local File Inclusion

Local File Inclusion (LFI) is a web app vulnerability where arbitrary local webserver files can be accessed through a web interface.

# Local File Inclusion


LFI vulnerabilities are typically identified when filenames appear in URL parameters when accessing webpages.

# What's Next?

In the next HackerFrogs Afterschool web exploitation workshop, we'll continue learning web exploitation skills with Natas CTF.

# Extra Credit

Looking for more study material on this workshop's topics?

See this video's description for links to supplemental documents and exercises!

# Until Next Time, HackerFrogs!