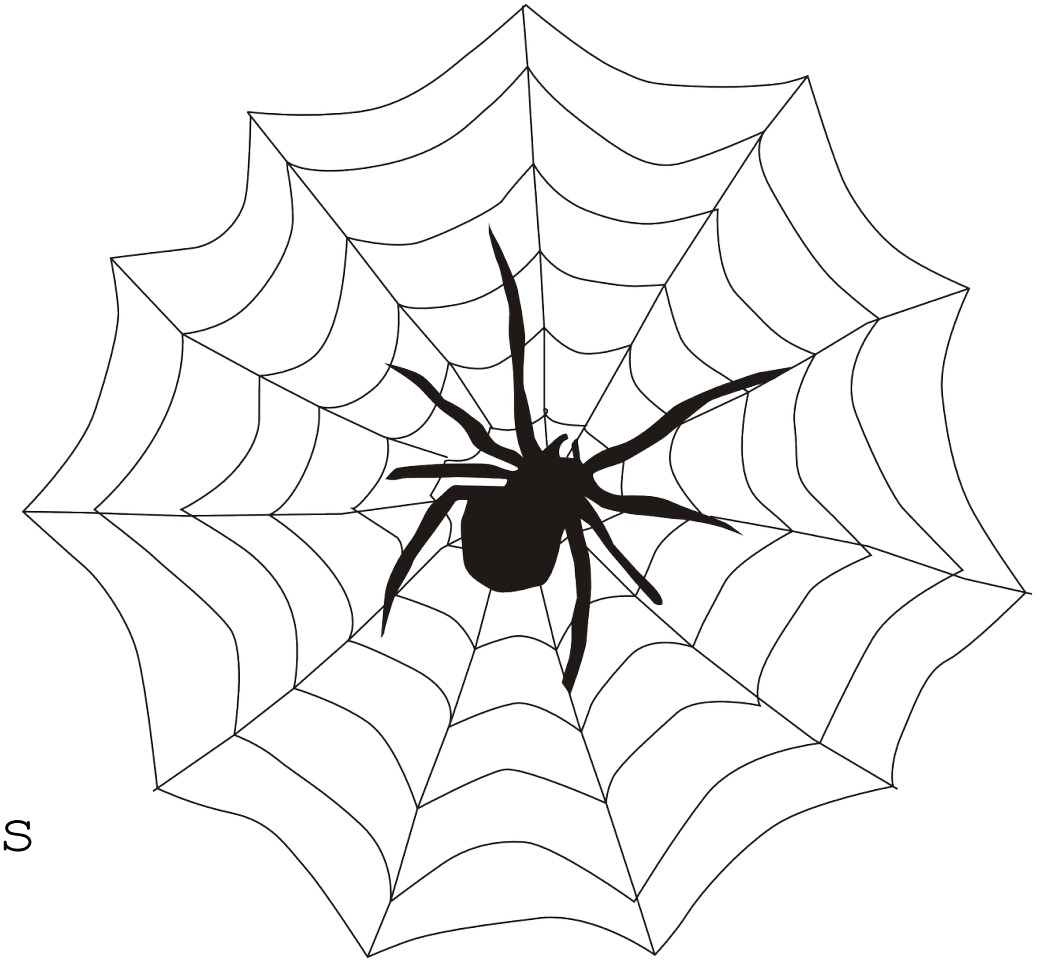
HackerFrogs Afterschool
Web App Hacking Basics: Part 4

Class:
Web App Hacking

Workshop Number:
AS-WEB-04

Document Version:
1.2

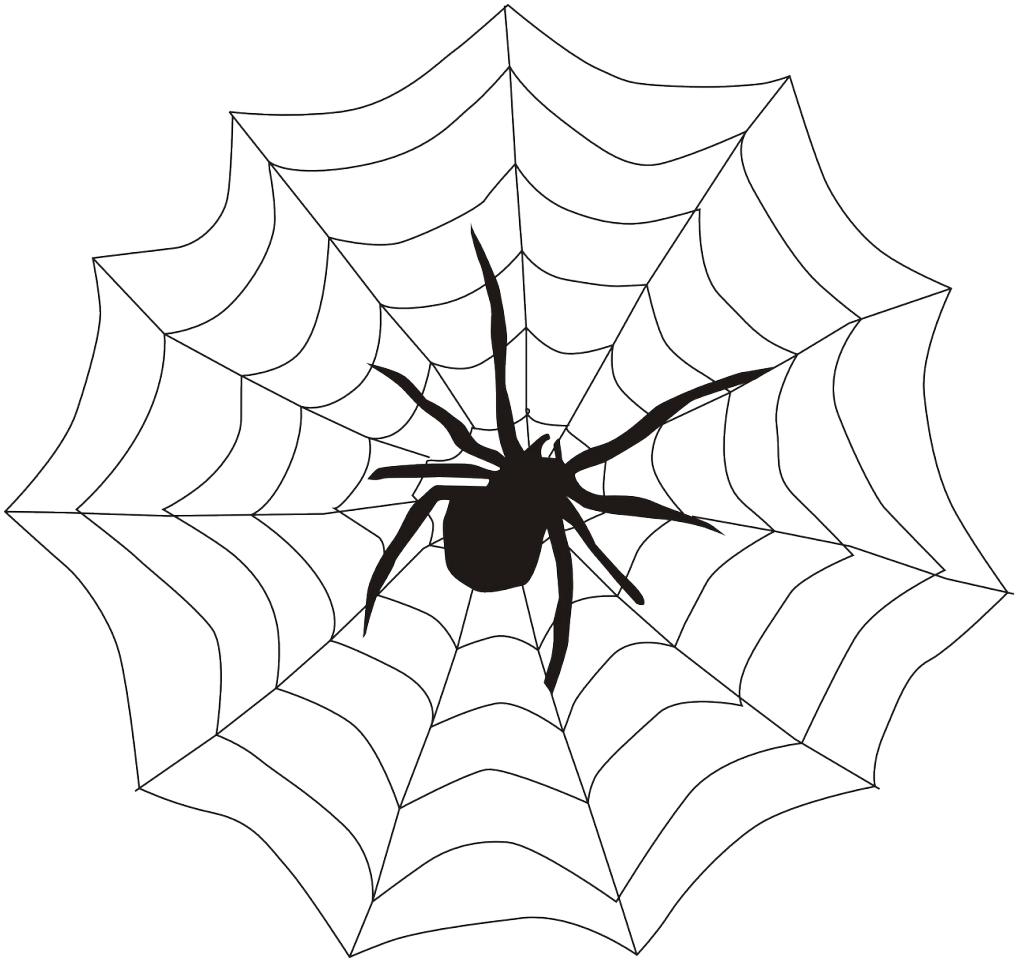
Special Requirements:
Completion of previous workshop, AS-WEB-03



What We Learned In The Previous Workshop

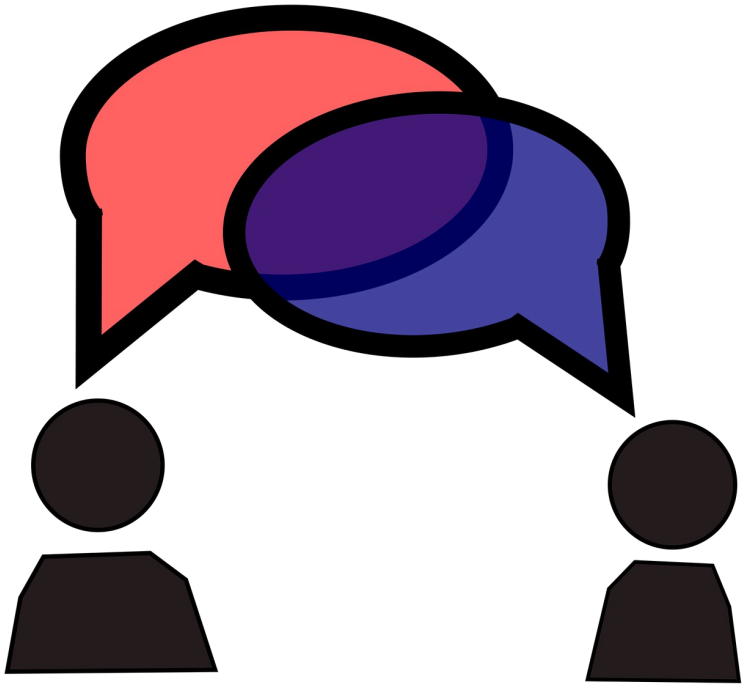
This is the fourth intro to web app hacking workshop.

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



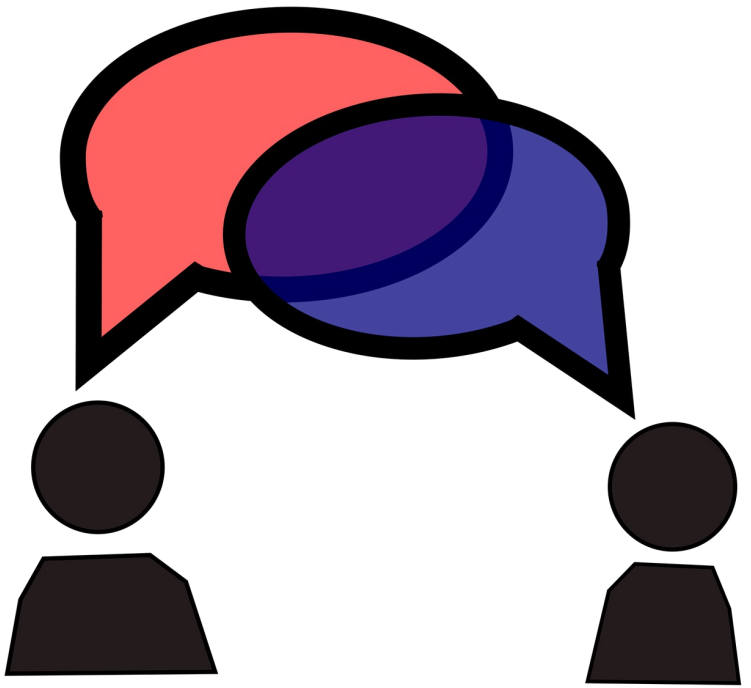
HTTP Referer Header

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

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.

Let's Continue Where We Left Off!

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

http://natas8.natas.labs.overthewire.org/

P-8 Sourcecode Analysis: Reverse Operations

```
$encodedSecret = "3d3d516343746d4d6d6c315669563362";

function encodeSecret($secret) {
    return bin2hex(strrev(base64_encode($secret)));
```

Since we have the value of the encoded secret, and we know which PHP functions were used to encode the secret, we can use the reverse PHP functions to obtain the decoded secret.

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

P-8 Sourcecode Analysis: Reverse Operations

We can use an online PHP interpreter website to assist us in this endeavor:

https://onlinephp.io/

P-9 OS Command Injection

```
if($key != "") {
    passthru("grep -i $key dictionary.txt");
```

The sourcecode shows that the PHP passthru function is used, which passes instructions to the underlying OS, and returns the results to the output stream.

P-9 OS Command Injection

```
if($key != "") {
    passthru("grep -i $key dictionary.txt");
```

Any user input that is passed directly to the webserver OS is potentially vulnerable to OS command injection.

P-9 OS Command Injection

Operating System (OS) Command Injection is a web app vulnerability where arbitrary OS commands can be performed on the webserver through a web interface.

P-9 OS Command Injection

OS Command Injection is a serious vulnerability, and can often lead to complete compromise of the webserver, and if so, the webserver can be used as a foothold to attack other machines on the network.

P-9 OS Command Injection
Injection String Anatomy 1

; cat /etc/natas_webpass/natas10 #

1) The semicolon terminates a command

2) The cat command reads files

3) This is the filepath to the natas 10 password file

4) The hash symbol nullifies anything that follows

P-9 OS Command Injection
Injection String Anatomy 1

; cat /etc/natas_webpass/natas10 #

1) The semicolon terminates a command

2) The cat command reads files

3) This is the filepath to the natas 10 password file

4) The hash symbol nullifies anything that follows

P-9 OS Command Injection
Injection String Anatomy 1

; cat /etc/natas_webpass/natas10 #

1) The semicolon terminates a command

2) The cat command reads files

3) This is the filepath to the natas 10 password file

4) The hash symbol nullifies anything that follows

P-9 OS Command Injection
Injection String Anatomy 1

; cat /etc/natas_webpass/natas10 #

1) The semicolon terminates a command

2) The cat command reads files

3) This is the filepath to the natas 10 password file

4) The hash symbol nullifies anything that follows

P-9 OS Command Injection
Injection String Anatomy 1

; cat /etc/natas_webpass/natas10 #

1) The semicolon terminates a command

2) The cat command reads files

3) This is the filepath to the natas 10 password file

4) The hash symbol nullifies anything that follows

P-9 OS Command Injection
Injection String Anatomy 1

grep -i ; cat /etc/natas_webpass/natas10 # dictionary.txt

The above command is what is sent to the webserver's OS for execution. Note: all content after the hash (#) is not executed, since is “commented out” .

P-9 OS Command Injection Input Validation

```
if($key != "") {
    passthru("grep -i $key dictionary.txt");
```

The reason why this code is vulnerable to OS command injection is because there is no input validation in place.

P-9 OS Command Injection
Input Validation

```
if($key != "") {
    passthru("grep -i $key dictionary.txt");
```

Input validation is the process of checking user input for any characters the input field is not designed to process.

P-9 OS Command Injection
Input Validation

```
if($key != "") {
    passthru("grep -i $key dictionary.txt");
```

In this case, the app should have done input validation for any characters that could be interpreted as OS commands, such as **; # & |** etc.

P-10 OS Command Injection Input Validation Bypass

```
if($key != "") {
  if(preg_match('/[;|&]/',$key)) {
    print "Input contains an illegal character!";
```

This version of the app implements input validation on these (; | &) characters.

P-10 OS Command Injection Input Validation Bypass

```
} else {
        passthru("grep -i $key dictionary.txt");
    }
```

Fortunately for us, the way the OS command is setup still allows us to read the password file.

P-10 OS Command Injection
Injection String Anatomy 2

a /etc/natas_webpass/natas11 #

1) An argument to the grep command, searching for the letter A

2) The file to be searched, the Natas 11 password file

3) The hash symbol nullifies what follows after

P-10 OS Command Injection
Injection String Anatomy 2

a /etc/natas_webpass/natas11 #

1) An argument to the grep command, searching for the letter A

2) The file to be searched, the Natas 11 password file

3) The hash symbol nullifies what follows after

P-10 OS Command Injection
Injection String Anatomy 2

a /etc/natas_webpass/natas11 #

1) An argument to the grep command, searching for the letter A

2) The file to be searched, the Natas 11 password file

3) The hash symbol nullifies what follows after

P-10 OS Command Injection
Injection String Anatomy 2

a /etc/natas_webpass/natas11 #

1) An argument to the grep command, searching for the letter A

2) The file to be searched, the Natas 11 password file

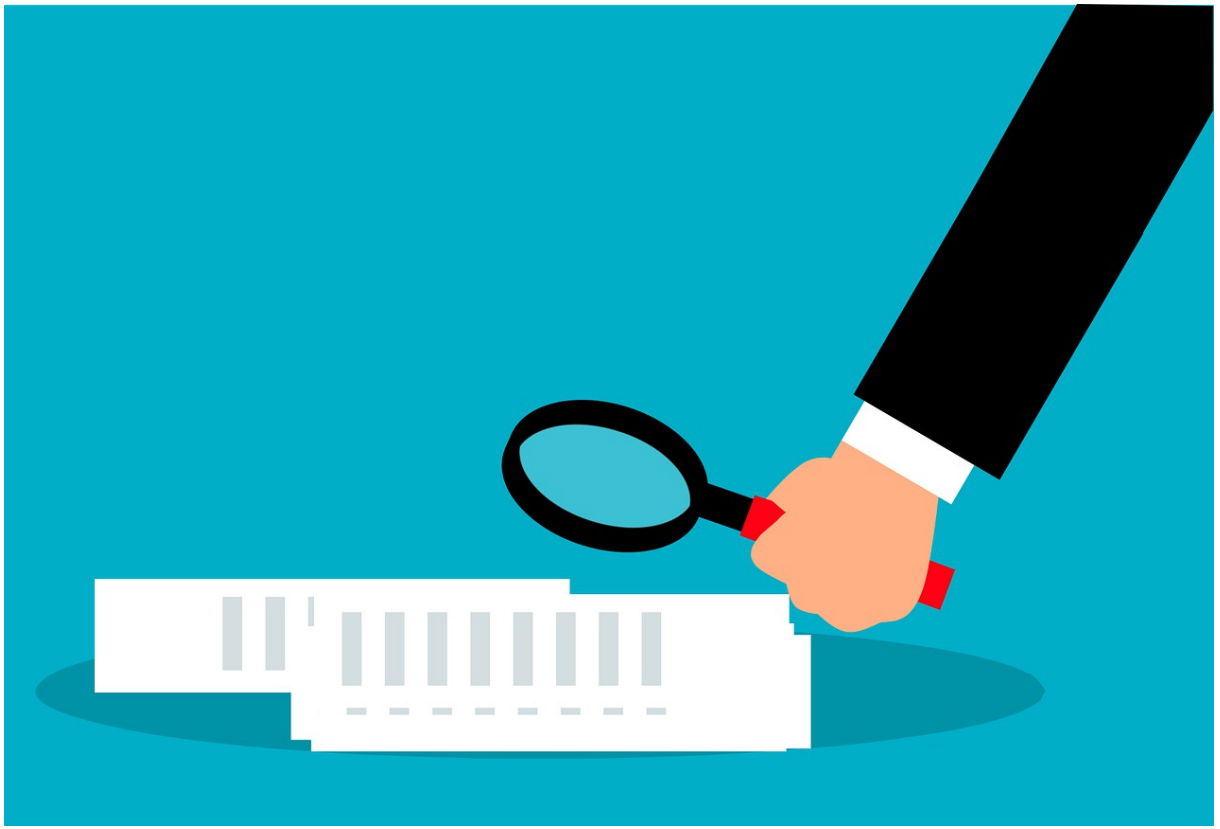
3) The hash symbol nullifies what follows after

P-10 OS Command Injection
Injection String Anatomy 2

grep -i a /etc/natas_webpass/natas11 # dictionary.txt

The above command is what is sent to the webserver's OS for execution. Note: all content after the hash (#) is not executed, since is “commented out” .

Summary



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

OS Command Injection

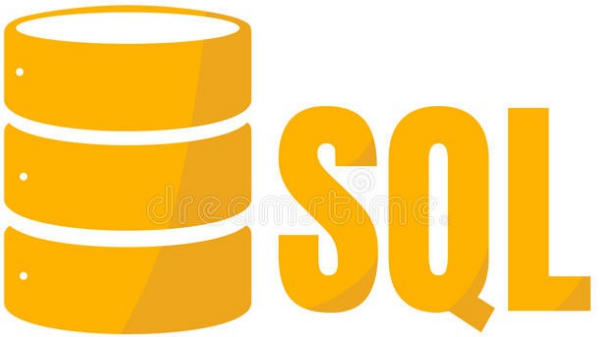
Operating System (OS) Command Injection is a web app vulnerability where arbitrary OS commands can be performed on the webserver through a web interface.

OS Command Injection

OS Command Injection can often lead to complete compromise of the webserver, and if so, the server can be used as a foothold to attack other machines on the network.

What's Next?

In the next HackerFrogs Afterschool web app hacking workshop, we'll learn about the SQL database language which most web apps use for user account credentials.



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!