Name: ______________________________________ Student ID:______________

You will need:
A computer with internet access

COS80013 Internet Security

Lab 6 (week 6)

In this lab you will bypass authentication using a variety of techniques.

1. Start **a web browser – Firefox of Chrome** will be suitable.
   The browser must have an Inspect console which lets you **edit cookies** and alter HTML form variables.
2. In the browser, go to https://www.hackthissite.org

If you can right-click on part of an HTML page and select ***Inspect***… you should be OK

If you don’t already have an account; create one.
Register: choose a suitable user ID (for example your student number preceded by an ‘s’). Choose a suitable password and either remember it or write it down – don’t re-use your SIMS password!
Use a real e-mail that you can check in class – you will need to respond to it before you can log in.

Note: Swinburne e-mails may be delayed by Swinburne’s automated spam checking software.

What are the minimum requirements for a password on this site?

Go to your e-mail client and respond to the registration activation message

3. Log in to HackThisSite
   Go to Challenges
   Click on Basic missions

4. Click on Basic 1

You are presented with an HTML page with a form.
You could try: `1' or '1' = '1` (SQL injection) or
`<script> alert(document.cookie);</script>` (XSS)
but before you even consider those, just look at the page source.

How do you view the source of the page in your browser?

Viewing a web page source is the easiest and first thing you should do when testing a web site. It's like listing a directory or running *strings* on a binary (or learning "Chopsticks" on a piano).

Name: _______________________________________ Student ID:______________

There will be a lot of template code from the menus and navigation panels in the source. Search for the code around the Level 1 challenge text.

What is the HTML for a comment? (marked in Green text)

What is the password? (inside the comment)

Enter the password into the form field and move on to the next level.
Next challenge (Basic 2)

5. After checking the source, go back to the clue. The description explains that the web developer planned for the password to be in a file on the server. A php script would compare the user input with the contents of the file. The PHP code may look something like this:

```
$lines = file("./randomname.php");
if (strcmp($lines[$0],$_REQUEST["password"] == 0)
//continue to next level
```

You may recall from your web studies that PHP does not always send error messages to the browser when things go wrong.

What will file( ) return if it fails? Look it up on the web.

What will be added to $lines if file( ) fails?

Back in HackThisSite, try submitting a blank password.
Does Nothing == Nothing?

Move to the next level. Next challenge (Basic 3)

Name: _______________________________________ Student ID:______________

6. Check the page source. Find the hidden variable that has the name of the password file.
   What is it?

   Try adding the name of the password file to the URI.

   What is the password?

   Sometimes this is referred to as "URL hacking"

   How could you prevent web users from reading a file on a web server? Do a search for this There are at least 2 ways of doing this.

   Move to the next level. Next challenge (Basic 4)

7. After checking the page source, you will notice that the "script" referred to is in level4.php. You can't load this without it executing. However, you can see Sam's e-mail address stored in a hidden variable.

   What is the name of the hidden variable and what is Sam's e-mail address?

   In ancient times, you could copy the page source to a local file, edit the e-mail address, save it your own web server, and load it into a browser to re-direct the email reminder to yourself.

   Alternatively, you could install a web proxy such as Burp or OWASP Zap (Burp is included in the Kali distribution), load the page into your browser, edit the DOM in Burp and substitute your e-mail address...

   But with modern browsers... It's all built into the DOM inspector.

Name: ______________________________________ Student ID:______________

In Firefox or Chrome, right-click on the Send password to Sam button and select *Inspect* or *Inspect Element*.

Find the element containing the e-mail address. Double-click on it and change it you your own e-mail address. Click elsewhere on the Inspector console and click on the button above.
Check your e-mail for the password.

What is the password?

Go back in the Browser, and submit the password.

Move to the next level. Next challenge (Basic 5)

8. The same approach will work for this level.

What is the password?

Go back in the Browser, and submit the password.

Move to the next level. Next challenge (Basic 6)

9. This level is a simple crypto challenge.

A form is provided to allow you to encrypt plain text to see what the cipher test looks like. This allows you to experiment and deduce the key used to encrypt the password. You then have to decrypt the provided cipher text using the key you derived.

Entry-level crypto challenges tend to rely on encoding (e.g. base-64), Caesar, ROT or Vigenere ciphers. Base-64 is a keyless translation from one bit-width to another. You can experiment here: https://www.base64decode.net/
The simple ciphers are *substitution* ciphers - where each character is changed individually to a ciphertext character. Caesar uses the same key for every character, the key being an integer which is added to the ASCII value of the plain text character to encrypt it. Decryption is the reverse - subtract the key from the ASCII value of the ciphertext character. In ROT-13 the Caesar key is fixed at 13.
In Vigenere, each character has a different key, sometimes determined by a table or square grid. In an OTP the key length is infinite and perfectly ransom. In Vigenere the key is repeated and systematic. The only problem is that Vigenere only works with letters of the alphabet...

Name: ____ Student ID:______________

To figure out the key, try encrypting a simple string such as 00000000 or aaaaaaaa.
Use 8 characters because the ciphertext we need to decrypt is only 8 characters.
.

Can you see a pattern in the ciphertext? Describe it.

Is the key the same for all characters? If not that rules out Caesar, ROT.

What is the key?

Find an ASCII table on-line and convert each ciphertext character to its int value.

Cipher test characters							
0	f						
ASCII int values for each one							
48	102						
Subtract key value for each position							
0	1						
48	101						
Convert back to ASCII characters							
0	e						

What is the plaintext password?

Go back in the Browser, and submit the password.

Move to the next level.

(Basic 7)

Name: ______________________________________ Student ID:______________

10. Level 7 has a UNIX/Linux bash command passthrough vulnerability. These turn up in CTFs often, and also appear in web sites when the developer is used to *bash* scripts, and wants to offer functionality normally provided by the operating system. ASP, PHP, MySQL/MariaDB and other scripted systems have functions which pass bash command strings through to the OS. These can be manipulated to execute arbitrary code on the server.
    On this page we are informed that the UNIX *cal* command will return results through the web page. There are two vulnerabilities here:
    1. The user can input the year as a string
    2. The output of the system call is redirected through the web server to the user.

    Look up the UNIX **cal** command.

    What are the inputs? How are inputs separated?

    There are at least 4 different ways of executing bash commands using PHP scripts. Read about them here:
    https://www.binarytides.com/execute-shell-commands-php/

    bash allows command stacking - where multiple commands are issued on the same line, separated by ; //or other characters (which you can look up).

    Time to experiment.
    click on "view" and leave the input blank.
    Try inputting 1999
    Try inputting 1999;ls
    Try inputting ;ls

    Appropriate validation would be restricting input to integers. This can be done using regex filtering or php char filtering functions:
    https://stackoverflow.com/questions/15723663/using-regex-to-filter-year-of-fixed-length-0-or-4-digit

    What is the name of the file containing the password?

    Append the name of the password file to the URI.

    What is the password?

    Go back in the Browser, and submit the password.

    Move to the next level.

(Basic 8)

11. In this level (8) there is a facility to input a string, and have it added to a file on the server, followed by sending the file contents back to the browser.

Name: _______________________________________ Student ID:______________

Experiment a bit. You could try some XSS, but it won't work as the < and > characters are sanitised (converted to HTML entities) and other characters are filtered out.

Try a string of characters which would not normally be in a name:
!@#$^&*()-_=+[]{};:'"

Which of these characters are filtered or sanitised? (use source view to check for HTML entities)

Look up Server Side Includes here:
https://web.archive.org/web/20190225032018/https://httpd.apache.org/docs/1.3/howto/ssi.html

What are SSI directives used for? Why are they not often used this century?

What is the general format of an SSI directive?

What are some common SSI directives elements (i.e. verbs or commands)?

In the HowTo document referred to above, find the SSI directive that lists files in a directory.

What is it?

Try inputting it in to the *"Enter your name*" field in the HackThisSite page.

The directory list (unformatted) contains the names of .shtml files in the https://www.hackthissite.org/missions/basic/8/tmp/ folder.

Change your injected ls command to one which shows the contents of the parent (..) folder.

i.e. `ls ..` or
`ls ./..`

What is the name of the .php file containing the password?

Append the name of the password file to the URI. Omit the /tmp part of the URI. Add it after .../basic/8/

What is the password?

Go back in the Browser, and submit the password.

Move to the next level.

(Basic 9)

12. This level has the same challenge - find the php file containing the password. It is in .../basic/9/. However there is no input script to use to get there, so use the one from level 8. This (and level 8) is a kind of directory traversal attack. To get to level .../basic/9/ to /basic/8/tmp/ you have to include directory identifiers (e.g. . .. / /name) to travel *up* the tree (towards root) and then *down* a different branch.

    How many /.. would you need to go from https://www.hackthissite.org/missions/basic/8/tmp/ to https://www.hackthissite.org/missions/basic/ ?

    What would you have to add (to ls) to get from https://www.hackthissite.org/missions/basic/ to https://www.hackthissite.org/missions/basic/9/ ?

    Therefore, what will be the combined ls command?

    Try it

    What is the name of the .php file containing the password?

Name: ______________________________________ Student ID:______________

Append the name of the password file to the URI. Omit the /tmp part of the URI. Add it after .../basic/9/

What is the password?

Go back in the Browser, and submit the password.

Move to the next level.

(Basic 10)

13. This level (10) has no hints or text input links, so start with the source, and then look at the DOM with **inspect.**

    How many hidden variables are in the HTML form?

    Using the DOM inspector, look for other methods of passing data from the browser to the server. Check Memory, Storage tabs (Firefox), or Application/Storage (Chrome). You should be able to find two cookies for www.hackthissite.org, one for the **PHPSESSID** and one for **level_10_authorized.**

    What is the value stored in level_10_authorized?

    Double-click on it and change it to ***yes***. Back on the web page, submit (any) password.

14. There is no link to level 11. Go there by changing the URI, replacing 10 with 11.

    The page source doesn't show much. The DOM inspector only shows the level_10_auth cookie from the previous level.

    For this level try brute forcing the URI.
    First, try .../basic/11
    Refresh the page a few times.

    What do the "songs" have in common?

    Try appending single characters to the URI:
    a, b, c, d, e, f,
    Stop when you find a directory

Name: _______________________________________ Student ID:______________

What is the first valid subdirectory you find?

Follow the subdirectories...
As you go, try appending robots.txt or .htaccess to each URI
If you get bounced back to the home page, click on the browser back button.

Note: file names starting with a . are hidden in UNIX/Linux

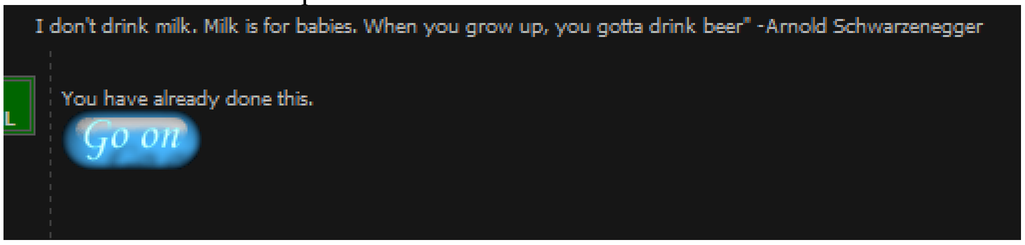
Eventually there will be no more subdirectories.
check for robots.txt or .htaccess again.

What does the htaccess directive: IndexIgnore do? Look it up.

What is the full URI of the DaAnswer file?

What is the string in the file?

This is a cryptic clue. Go back to the form submission page:
https://www.hackthissite.org/missions/basic/11/index.php
Try strings which are variations on the clue...
e.g. *right in front of you, easy to see, in this place...*
Your reward will be a quote and button:


I don't drink milk. Milk is for babies. When you grow up, you gotta drink beer" -Arnold Schwarzenegger

You have already done this.

End of Lab.