

Nome: Yuri Medeiros da Silva

---

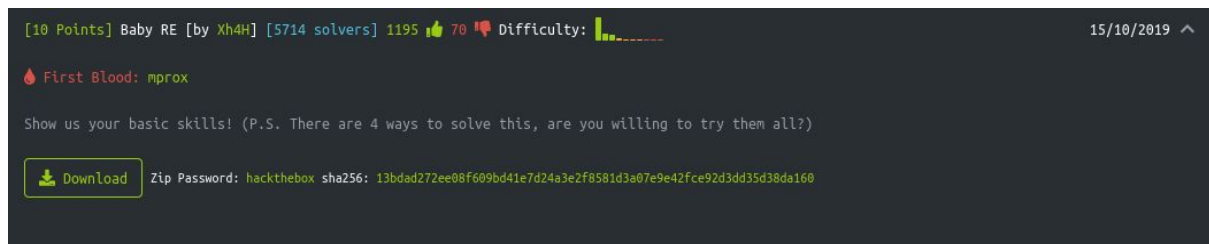
Nesta Tag foi pedido para escolhermos um desafio do HTB para resolver. os dois que eu resolvi foram de Engenharia reversa.

Esse arquivo contém :

- Baby RE
- DSYM RE
- Como criar conta

BABY RE [<https://www.hackthebox.eu/home/challenges/Reversing> --  
BABY RE ]

Esse primeiro foi bem básico, eu apenas descompilei com o ghidra e obtive a flag procurada.



Basicamente baixei o arquivo e abri no ghidra. ao abrir no ghidra procurei a função main e nela tinha uma comparação de strings.

```
undefined8 main(void)
{
    int iVar1;
    undefined8 local_48;
    undefined8 local_40;
    undefined4 local_38;
    undefined2 local_34;
    char local_28 [24];
    char *local_10;

    local_10 = "Dont run `strings` on this chall
    puts("Insert key: ");
    fgets(local_28,0x14,stdin);
    iVar1 = strcmp(local_28,"abcde122313\n");
    if (iVar1 == 0) {
        local_48 = 0x594234427b425448;
        local_40 = 0x3448545f5633525f;
        local_38 = 0x455f5354;
        local_34 = 0x7d5a;
        puts((char *)&local_48);
    }
    else {
        puts("Try again later.");
    }
    return 0;
}
```

apenas rodei o programa com o input "abcde122313" (que aparece na linha do strcmp) e obtive a flag

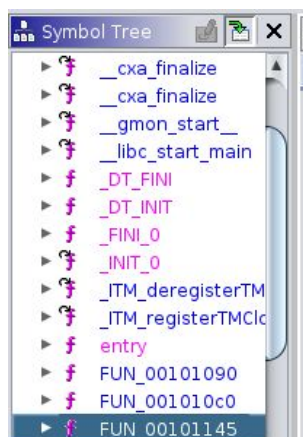
A terminal screenshot showing the output of the program. It says 'Insert key:' followed by the input 'abcde122313' and then the flag 'HTB{B4BY\_R3V\_TH4TS\_EZ}'.

Outra forma seria ver o que tem no endereço 0x594234427b425448 (local\_48) .

Como esse era muito simples, fiz outro abaixo um pouco mais chatinho.

DSYM [<https://www.hackthebox.eu/home/challenges/Reversing> --  
DSYM]

Baixei o executável e abri no ghidra. Esse não tinha nenhuma função já com o nome de “main”, então tive que ir procurando as funções uma por uma.



Com o disassembly de `FUN_00101145` no ghidra temos

```
local_68[0] = 0x2cf;
local_68[1] = 0x2dd;
local_68[2] = 0x2d5;
local_68[3] = 0x2e1;
local_58 = 0x2f6;
local_54 = 0x2aa;
local_50 = 0x2f2;
local_4c = 0x2c5;
local_48 = 0x2ff;
local_44 = 0x2a9;
local_40 = 0x2ae;
local_3c = 0x2e3;
local_38 = 0x2e3;
local_34 = 0x2f6;
local_30 = 0x2c5;
local_2c = 0x2ee;
local_28 = 0x2aa;
local_24 = 0x2fd;
local_20 = 0x2c5;
local_1c = 0x2e0;
local_18 = 0x2a9;
local_14 = 0x2e7;
printf("You almost got me :D\nHere is small price for you: ");
local_c = 0;
while (local_c < 0x16) {
    auStack200[local_c] = local_68[local_c] ^ 0x29a;
    printf("%x", (ulong)auStack200[local_c]);
    local_c = local_c + 1;
}
```

Essa é a mensagem que recebemos quando damos um **strings** no binário ( “You almost ... you :”). Como ele diz small price, e só pintar algumas variáveis, vamos pegar todas elas e ver no que dá.

```
local_68[0] = 0x2cf;  
local_68[1] = 0x2dd;  
local_68[2] = 0x2d5;  
local_68[3] = 0x2e1;  
local_58 = 0x2f6;  
local_54 = 0x2aa;  
local_50 = 0x2f2;  
local_4c = 0x2c5;  
local_48 = 0x2ff;  
local_44 = 0x2a9;  
local_40 = 0x2ae;  
local_3c = 0x2e3;  
local_38 = 0x2e3;  
local_34 = 0x2f6;  
local_30 = 0x2c5;  
local_2c = 0x2ee;  
local_28 = 0x2aa;  
local_24 = 0x2fd;  
local_20 = 0x2c5;  
local_1c = 0x2e0;  
local_18 = 0x2a9;  
local_14 = 0x2e7;
```

Para isso copiei todos esses valores e escrevi um programa que faz a mesma coisa que o binário fazia, pega cada um desses valores e faz o xor com 0x29a.

```
package main  
  
import "fmt"  
  
func main() {  
    src := []uint16{0x2cf, 0x2dd, 0x2d5, 0x2e1, 0x2f6, 0x2aa, 0x2f2, 0x2c5, 0x2ff, 0x2a9,  
        0x2ae, 0x2e3, 0x2e3, 0x2f6, 0x2c5, 0x2ee, 0x2aa, 0x2fd, 0x2c5, 0x2e0, 0x2a9, 0x2e7}  
    var ans string  
    for _, v := range src {  
        ans = ans + string(int(v^0x29a))  
    }  
  
    fmt.Println(string(ans))  
}
```

```
The Go Playground  Run  Format  Imports  Share

1 package main
2
3 import "fmt"
4
5 func main() {
6     src := []uint16{0x2cf, 0x2dd, 0x2d5, 0x2e1, 0x2f6, 0x2aa, 0x2f2, 0x2c5, 0x2ff, 0x2a9, 0x2ae,
7     var ans string
8     for _, v := range src{
9         ans = ans + string(int(v ^ 0x29a))
10    }
11    fmt.Println(ans)
12 }
13 |
14 }
15
16
17
18
19
20
21
22
23
24
25
26
```

UGO{10h\_e34yyl\_t0g\_z3}

UGO{10h\_e34yyl\_t0g\_z3}

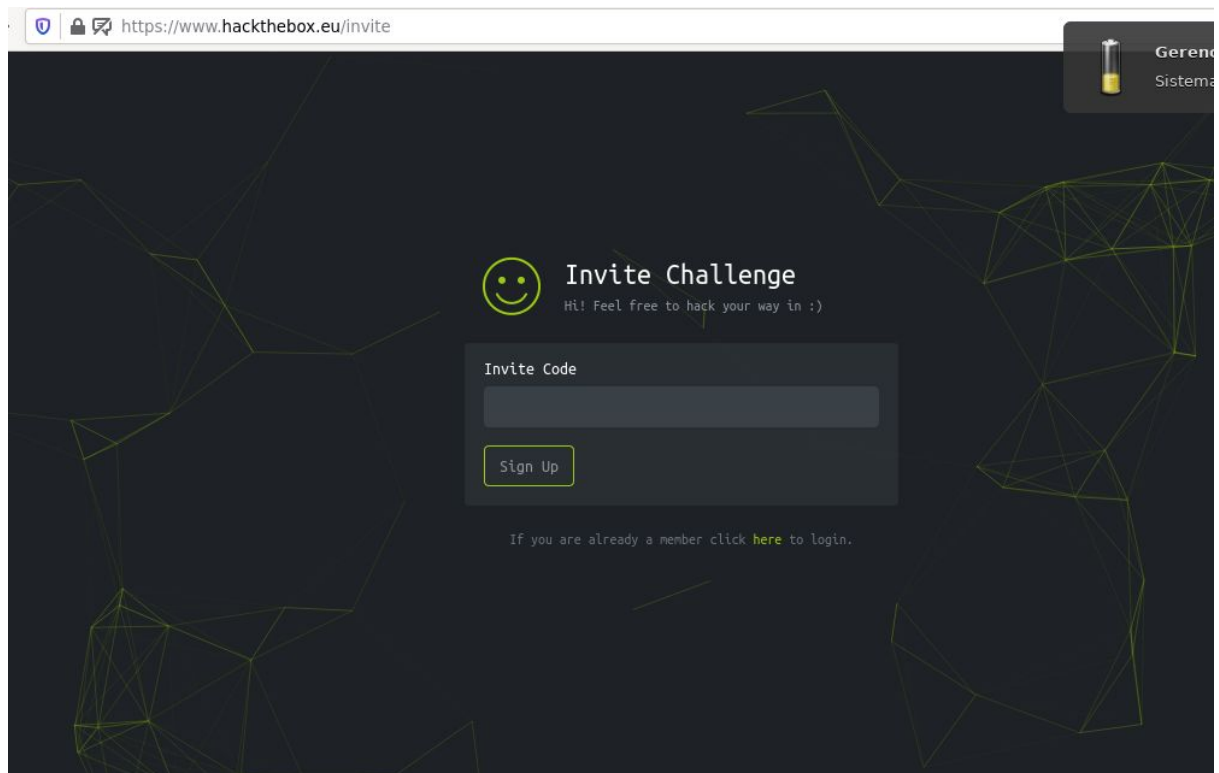
Parece muito com o formato de flag do HTB, e  $U + 13$ ,  $G + 13$ ,  $O + 13 = HTB$ , o que é uma cifra de César.

E aí eu utilizei o site <https://cryptii.com/pipes/caesar-cipher> com UGO{10h\_e34yyl\_t0g\_z3}

VIEW	ENCODE DECODE	VIEW
Ciphertext	Caesar cipher	Plaintext
BNV{s0o_l34ffs_a0n_g3}	SHIFT -6 a→u	HTB{y0u_r34lly_g0t_m3}
	ALPHABET abcdefghijklmnopqrstuvwxyz	
	CASE STRATEGY Maintain case	FOREIGN CHARS Include Ignore
	→ Decoded 22 chars	

## COMO CRIAR CONTA NO HTB

Primeiro acessamos o link do invite.



Claramente parece uma chall web. Então, vamos vasculhar o código.

```
<div class= "native-du "></div>
  <script>...</script>
</div>
  <div id="particles-js" class="particles_full">...</div>
</section>
</div>
<script src="https://www.hackthebox.eu/js/htb-frontend.min.js"></script>
<script defer="" src="/js/inviteapi.min.js"></script>
<script defer="" src="https://www.hackthebox.eu/js/calm.js"></script>
</body>
```

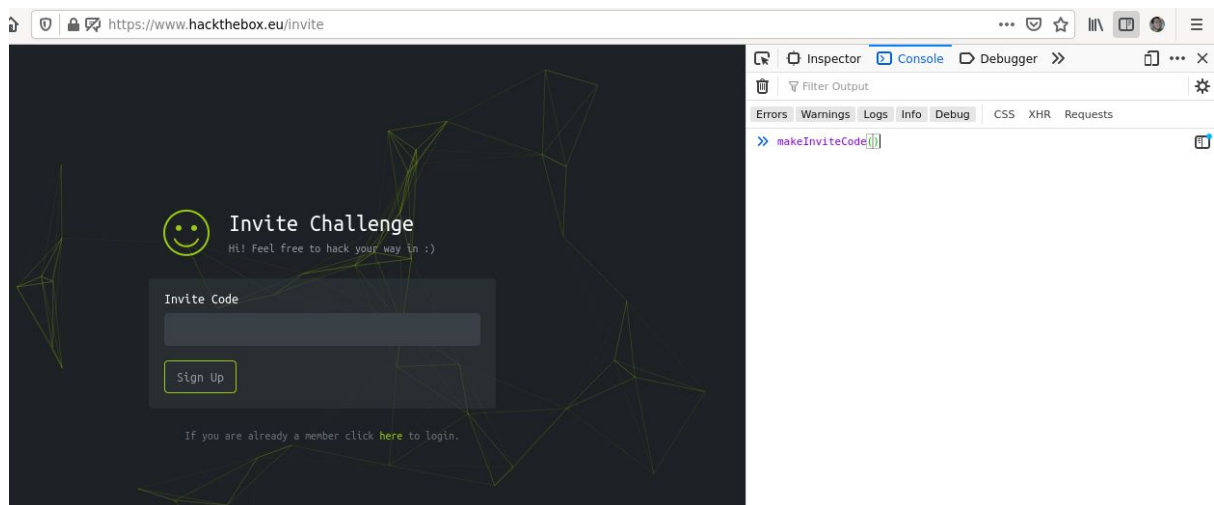
Esse inviteapi, pelo nome, parece suspeito.

← → ↺ 🏠 ⓘ 🔒 🗨️ https://www.hackthebox.eu/js/inviteapi.min.js

```
eval(function(p,a,c,k,e,d){e=function(c){return c.toString(36)};if(!''.replace(/^/,String)){while(c--){d[c.toString(a)]=k[c]||c.toString(a)}k=[function(e){return d[e]}];e=function(){return '\\w+'};c=1;while(c--){if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}return p}('1 i(4){h 8={\"4\":4};$.9({a:\"7\",5:\"6\",g:8,b:\"/d/e/n\\\",c:1(0){3.2(0)},f:1(0){3.2(0)}}}1 j(){$.9({a:\"7\",5:\"6\",b:\"/d/e/k/l/m\\\",c:1(0){3.2(0)},f:1(0){3.2(0)}}}')',24,24,'response|function|log|console|code|dataType|json|POST|formData|ajax|type|url|success|api|invite|error|data|var|verifyInviteCode|makeInviteCode|how|to|generate|verify'.split('|'),0,{}))
```

```
function verifyInviteCode(code) {  
    var formData = {  
        "code": code  
    };  
    $.ajax({  
        type: "POST",  
        dataType: "json",  
        data: formData,  
        url: '/api/invite/verify',  
        success: function(response) {  
            console.log(response)  
        },  
        error: function(response) {  
            console.log(response)  
        }  
    })  
}  
  
function makeInviteCode() {  
    $.ajax({  
        type: "POST",  
        dataType: "json",  
        url: '/api/invite/how/to/generate',  
        success: function(response) {  
            console.log(response)  
        },  
        error: function(response) {  
            console.log(response)  
        }  
    })  
}
```

Analisando, vemos que tem uma rotina que cria um invite code. vamos chamar essa rotina e ver o que ela nos retorna.



```
> echo "Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFg erdhrfg gb /ncv/vaivgr /trarengr" |tr '[A-Za-z]' '[N-ZA-Mn-za-m]'
In order to generate the invite code, make a POST request to /api/invite/generate
```

Ela nos retorna uma string em rot13, utilizamos do comando tr pra decifrar e recebemos outra rota para fazer um post.

```
> curl -X POST https://www.hackthebox.eu/api/invite/generate
{"success":1,"data":{"code":"RkhVQVEtVlVXSkitUEZXQlUtT0lKTFMtR0hXSkw=","format":"encoded"},"0":200}%
```


```
> echo "RkhVQVEtVlVXSkitUEZXQlUtT0lKTFMtR0hXSkw=" | base64 -d
FHUAQ-VUWJB-PFWBU-OIJLS-GHWJL%
```

```
> █
```

Fazendo o post conseguimos o invite-code.



→ ↻ 🏠 🔒 🗨️ https://www.hackthebox.eu/register

 Hack The Box  
PEN-TESTING LABS

Individuals Companies Universities Store Gift Cards Log

**Username**


**E-Mail**

**Password**

☐ I accept the [Terms of Service](#).

☐ Product Updates

☐ I'm not a robot

  
reCAPTCHA  
[Privacy](#) - [Terms](#)