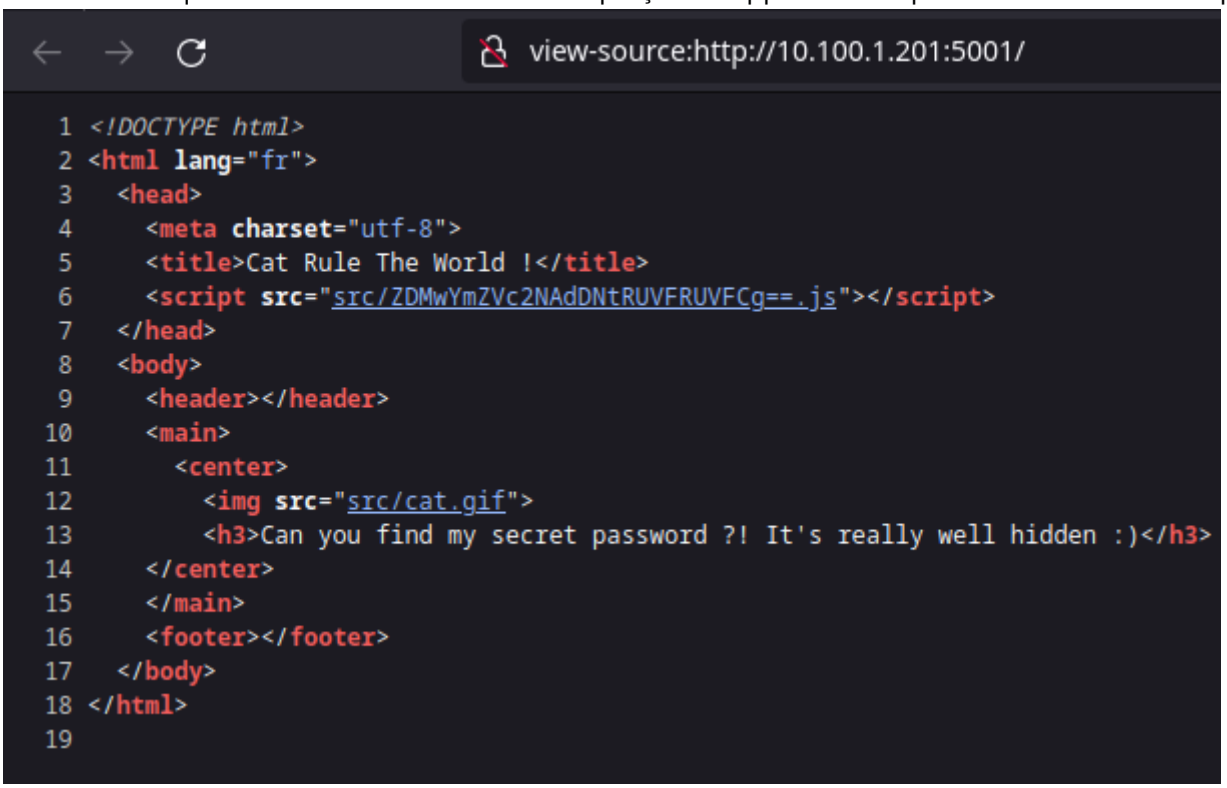
JavaScript ObfusCATion

Auteur: bWlrYQ

1. Recon

Dans un premier temps on se rend sur la page et on a le visuel ci-dessous

C'est très basique mais dans le code source on aperçoit un appel à un script dont le nom est assez particulier



Décodons-le

```
mika@bwlryq ~/D/ctf ./rw
$ echo "ZDMwYmZVc2NAdDNtRUVFRUVFCg==" | base64 -d
d30bfUsc@t3mEEEEEE
```

2. Recouvrement du code JavaScript

On retrousse ses manches et on déobfusque tout ça ! Voilà le code source du fichier JavaScript

```
ﾟωﾟ/= /｀m´）/ ~┻┻  //*´∇｀*/ ['_']; o=(ﾟｰﾟ)  =_=3; c=(ﾟΘﾟ) =(ﾟｰﾟ)-(ﾟｰﾟ); (ﾟДﾟ)
=(ﾟΘﾟ)= (o^_^o)/ (o^_^o);(ﾟДﾟ)={ﾟΘﾟ: '_' ,ﾟωﾟ/ : ((ﾟωﾟ/==3) +'_') [ﾟΘﾟ] ,ﾟｰﾟ/ :(ﾟωﾟ/
+ '_')[o^_^o -(ﾟΘﾟ)] ,ﾟДﾟ/:((ﾟｰﾟ==3) +'_')[ﾟｰﾟ] }; (ﾟДﾟ) [ﾟΘﾟ] =((ﾟωﾟ/==3) +'_')
[c^_^o];(ﾟДﾟ) ['c'] = ((ﾟДﾟ)+'_') [ (ﾟｰﾟ)+(ﾟｰﾟ)-(ﾟΘﾟ) ];(ﾟДﾟ) ['o'] = ((ﾟДﾟ)+'_')
[ﾟΘﾟ];(ﾟoﾟ)=(ﾟДﾟ) ['c']+(ﾟДﾟ) ['o']+(ﾟωﾟ/ +'_')[ﾟΘﾟ]+ ((ﾟωﾟ/==3) +'_') [ﾟｰﾟ] +
[........snipped.........] (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+ ((ﾟｰﾟ) + (o^_^o))+
(ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+ ((ﾟｰﾟ) +
(o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+
((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+
(o^_^o)+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[
ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (o^_^o)+ ((ﾟｰﾟ) +
(o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (o^_^o))+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟoﾟ]) (ﾟΘﾟ))
('_');
```

En faissant quelques recherche on tombe sur un encodage spécifique au javascript, `aaencode - Encode any JavaScript program to Japanese style emoticons (^_^)`. On peut encoder du code ici et on peut le décoder ici.

On obtient le code ci-contre:

```
_____ =
"Vm0wd2VFNUdWWGhVV0dSUFZsZFNXRll3Wkc5V2JHeDBaRWhrVlUxV2NEQmFWV2hyVm14S2MyTkVRbUZXV
jJoeVZtcEdZV015VGtsaFJtUlRaV3RGZUZadGVGWmxSbVJJVm10a2FWSXdXbGhXYlRWRFpWWmFjbFZyWkZ
wV01ERTBWa2MxVDJGV1NuVlJhemxXWWxSV1JGWkdXbUZqYkZaeVdrWndWMkpXU2xsV1Z6QXhWREZrU0ZOc
lpHcFNWR3hZV1d4b1UwMHhWWGhYYlVacVlrZDBObGxWV2s5VWJVWTJWbGhrVjFaRmIzZFpla1poVjBaT2N
scEdhR2xTTW1oWFZtMTBWMU14VWtkWFdHUllZbGhTY1ZSV1pGTk5SbFowWlVVNVZXSkZjRWhXTW5oelZqS
kdjbUo2UWxwbGEzQklWbXBHVDJNeFduUmlSazVwVmpKb1dWWXhaREJoTVZaMFZtNU9WbUpIVWxsWmJGWmh
Wa1pTVjFwR1RteGlSbkJKV2xWVk5XRkdTWGhYYm14WFRXNW9kbFl3V21GU2JFNTBZVVprYUdFeGNHOVdWM
0JIWVRKT2MxcElUbWhTTW1oeldXeG9iMkl4V25STldHUlZUVlpXTlZadE5VOWhWa3BIWTBaU1dtRXhWWGh
XTVZwWFkxWkdWVkpzVWxOaVIzY3hWa1phVTFVeFdYZE5XRXBYWVd4S1lWWnJWVEZrYkZweFUydDBXRll3V
2twV01uaDNZa2RGZUdOR1VsaGlSbHBvV1ZSS1QyTXhjRWxWYld4VFlYcFdWVmRYZEc5Uk1XUkhWMjVTVGx
OSGFGQlZiWE40VGtaVmVHRkhPV2hpUlhCWVZqSjRUMWR0U2xWU2JGSlhUVVp3YUZwRlpFOU9iRXB6V2tkc
1UySkhPVE5XTVZKUFpERkplVkpyWkZSWFIxSllXVmQ0WVdGR1duTmFSemxPVFZad2VGVnRkREJoTVVwelY
yNW9WMUl6YUhKWlZscGhVbXhrYzJKR1pHbFhSVXBKVmxod1MxUXhXWGhhU0ZaVllsVmFWRmxzWkc5V1Zsc
EhWV3QwYVUxV1ducFdNalZUVkd4YVJsTnNhRlZXTTA0MFZGUkdVMk14WkhSa1JtUnBWbGhDU2xac1pEUmh
NV1J6VjJ0YVdHRnJOVmhWYTFaaFpXeGFjMWRzVG1waVJrcDZWMnRrYzFVeVNrZGhNMlJYWWxob2NsUlZaR
VpsUm1SellVWlNhVkp1UWxwWFZ6QjRUa1pzVjJKR1ZsUmlWR3hYVlcxNGQyVnNXWGxsU0dScFVqQndSMVl
5TlhkWFIwVjVWV3RvVjJGcmNFeFZNVnBIWTIxS1IxcEdUazVOYldoMlZtcEdZVmxYUlhoYVJXUlZZbXR3V
1ZsclZrdFhSbXh6VjJ0MFYxWnNjREJaTUZVMVlVZEtWMWRyYUZkTmFsWk1WbTB4Um1WV1ZuTmFSbFpYWWt
oQmVsWkdWbUZXYlZaWVZXdG9VMkpHV25CVmFrWkxVMFphY1ZOcVVtbE5WbXd6VkZaV2IxWnNXa1pUYkdoV
```

```
1lURmFhRnBYZUdGalZrcDBaRWR3VGxkRlNraFdSRVpoWVRKR1YxTnNiRkpXUlhCWVdXeG9iMk5zVWxaV1d
HaFRUVlp3V2xsVldsTmhWa2w2WVVaU1dGWXpVbWhhUkVaYVpVWldjMXBHYUdoTk1VcFdWbGN4TkdReFRsZ
FZiR1JYWWxoU2IxbHNWbmRXTVZwMFkwZEdXRkl3VmpSWk1GcHZWakZKZW1GRmVGZGlSbkJvV1hwS1IxSXl
Sa2hpUms1cFlUQndXbFp0ZEdGWlZteFhZa1prV0ZkSGFGZFpiWE14WTBaV2NWTnFVbGRTYkd3elZqSjBNR
mRHV25OalJteGhWbGRSZDFaSGVFdFdWbHB5WVVaa1RtSnNTbmxXTVZwaFV6RktjMVJ1VGxoaVNFSndWVzE
0ZG1Wc1dsaGpSV1JXVFZac05WVnRkR0ZaVmtwMFZXczVWMkZyV2t4Vk1uaHJZekZhYzFkck9WZGlWa28yV
m1wS2QxbFdWWGxTYms1cVVteHdXRmxyV25kTk1WcFdWMjVPVDJKRmNIcFhhMlIzWVZaYVZWWnJhRmRTTTJ
ob1dWUkdXbVZHVG5OYVIyeHNZVEJ3V1ZkWGRHRlRNVTVIWWtaV1VsWkZXbFJVVmxwaFRWWmFXR1ZGT1doV
01Ga3lWbTE0YzFkR1duUlZWRUpZVm14d1lWcFZXbXRYVm5CSVVteE9VMkV6UWxwV2ExcGhZakZGZUZwRlp
GaGlhelZYV1ZSS1UxZEdVbGRYYm1Sc1ZteEtlbFl5Tld0WFIwcFdWbXBTVjJKSGFIWldha3BIWTJ4a2NtV
kdaR2xYUjJoNVZtMTRZVk15VFhoYVNFcFBWako0Y0ZacVNtOVdNVnAwWlVaT1ZFMXNXakJXYlhSclYwZEt
jbU5GT1ZkaVZFVXdWbXRhYzJOc1pISmtSbWhUWWtWd1YxWlVTWGRPVmxwelUyNVNWbUpIYUZsWlZFWkxWa
1phY1ZGWWFGTldiSEI2V1ZWYWEyRldaRWhoUkZwWVZteHdhRlY2U2s1bFZsSnlZa1pXYVZJeFNuZFdWekV
3WkRGa1YxZHVVazVXUmtwWVdXeGFZVk5XVm5Sa1NFNVhWakJ3U0Zrd1dsTlhiRnBHVGxWU1ZrMVdjR2hhU
lZWNFVsWktjMVpyTlZkaWEwcGFWbTF3UjJJeVVYaFdibEpVWVRKb1YxbHNaRzlYUm14ellVYzVXRkpzU25
sV01qVkxZVEF4V0ZWcVFsWmlXR2gyVm1wQmVGTldSbkpoUm1SVFVsWndiMWRXVWtkV2JWWkhZMFZhV0dKR
1NtOVVWM2hMVjFaYWRHUkdaR3RoZWtaSVZqSjRWMVV5UmpaaVNFWmFWMGhDU0ZZd1dscGxWMDQyVW14b1U
yRXpRbGxYVkVKaFdWWlplVkp1U2xSaWEzQmhXVmQwWVZVeGNGWlhiWFJyVm1zMWVsbFZaSE5WTURGV1kwU
lNWMkV4Y0doWFZscE9aVVp3UjFwSGFFNU5iRXBhVjFkNFYxbFhUa2RXYmxKc1UwZFNXRmxyV21GTlJsSnp
Xa1JDVjAxRVJubFpNRlp6VjJ4YVdHRkVUbGRoYTFwTVZXMTRhMlJIUmtkWGJXeFhVbFp3VWxac1kzaE9Sb
XhZVkZob1YySnNTbkJWYlhNeFlqRlNWVkZyZEZWaVJuQXdXa1ZrUjFkc1duTmpSRUpYWWxoU2RsWnNaRXR
UUmxaelYyeHdhRTFZUW05V2FrSmhZekpPYzFkdVNtdFNiVkpQVm0xMGQxZFdXbkZUYWxKclRWWktlbFl5T
lU5aGJFcFlaVWRvVjJKR2NETldWVnBoWTJ4d1JtUkdaRTVXTVVvMlZqSjBZV1F4VW5OVGJsWlNZa2Q0V0Z
sc2FGTmhSbVJYVjJ0MGFrMVlRa3BWTW5oRFZqSktjbE5zYkZkV00yaFlWakp6ZUZJeGNFZGlSM0JUVmpGS
1dGWkdXbXRWTVU1WFYydG9hMUl6VWxsVmFrSjNWMnhzVmxkdE9WZE5hMVkwVmpJeGIxWlhSWGhqUjJoYVR
WWndURmw2UmxOak1VNXlUbFprVjFKV1ZqTldiVEYzVXpBeFIySkdhRk5oTVhCUFZUQmtOR0l4Vm5SbFNHU
llVbTE0V1ZremNFZFZNVXB6VjFSS1ZtSlVWbEJaVnpGTFVqSk9SMk5HY0ZkV01VbDZWbTF3UzFNeFRsZFN
ibEpUWWtkb1dGUlVSa3RpTVZwWlkwVjBWRTFWYkRSWGEyaFhWbGRLU0dGR2FGWmlSbkF6VmpGYVlWZEhUa
1pQVjJ4T1ZtNUNTVlp0ZUc5ak1WSnpWMjVTVm1KSGFHRldNR2hEVTBaWmQxZHRSbXRTTVZwSFZERmFhMVJ
zV2xsUmJVWlhWbTFTTmxwVldscGxWazV6WWtaYWFWSXlhR2hXUm1SM1VqRmtSMVp1UmxOaVIxSnhWRmQwW
VZOV2EzZFhhemxZVW10d1Yxa3dVazlXTWtwSVZWUkNWMVpGV2t0YVZscGhZMnh3UjJGSGJHaGxiRm95Vm1
4U1ExWnJNVmRYV0docFUwVTFXVmxzVm1GWFZsWjBaVWhrVGxKdGRETldNakV3VmpBeFYySkVUbHBOUmxwM
lZtMXplR1JXVmxsYVJtaFhZa2hDYjFkWWNFZFpWbVJZVW10a1dHSlhlRlJVVmxaM1UxWlplV1ZIUm1oTlZ
tdzBWVEkxUzFReFduUmhSemxXWVd0dk1GWnJXbk5qYkhCRlVXczVVMkpJUVhkWGExWnJVakpHUmsxWVNrN
VdSa3BZVkZkd1YxWkdXbkZUYTNSVFRXczFTRlZYZUd0aFZtUklZVWM1VjJKWWFHaFpla1poVmpGT2RWTnR
kRk5pUm5CVlYxZDBiMUV3TlZkWGJsSk9Va1ZhVlZSWGRIZFRSbFY1VGxVNVYxSXdjRWxhVldSSFZsWmFWM
k5IYUZWV1JWcG9WVEJrVjFKdFVrZGFSbVJPVTBWS1NWWnRNSGhPUjBWNVVteGtWRmRIZUc5VmExcDNWMFp
zVlZKcmRGUlNiVkpZVmpKME1HRXhTbkpPV0d4WFlsaG9jbGxXV2t0amJVNUpXa1prVjJWclZqTlhhMUpIV
lRGSmVGcElTbUZTYldod1ZXMDFRMlZzV2xoalJVcE9WakZhZWxZeWVHOWlSa28yWWtoQ1ZtSllVak5XYlh
oaFpFVXhSVlpzYUdsV1Zsa3dWMVJDYTFJeFdsZFhiazVxVWpKb1YxbFhkR0ZOTVZWNFYyNWtVMDFXU2pCW
mExcHJZVlpPUmxOcmNGZGlXRkpvVjFaYVdtVkdXbGxoUm1oWVVqSm9iMVpYZUd0aU1rMTRWbTVTYTFKWVV
sbFZiVEUwVm14V2RFMVZaR2xTTUhCSVZUSTFkMVl3TVhWVVZFWmFWak5PTkZacVJtdGtWMHBIWTBVMVUyS
kdXVEJXYlRGM1VqRnNWMU51VG1GVFJWcFlXV3RrVTJJeFVsZGhSVTVzVm14d1NWcEZaRWRXYXpGelUydG9
WazFxVmxoV2JHUkxVMFpXZEU5V2NHaE5XRUY2Vm0xd1IxbFhVa2hVYTFwUVZtdHdUMWxyV25kWFZscHpXa
1JTYUUxV2JEVlZNalZUVm0xS1NHRkdhRnBpUjJoUFdsVmFZV1JIVmtkYVIzUlRUVVJSZVZaWE1IaGlNVlY
zVFZWa1dGWkZXbGxaYTFwaFpHeHdSVkpzY0d4U2F6VXhWbGQ0WVdGSFJqWldiR3hZVmpOb1ZGVnFSbXRTT
VdSMVVteE9hRTB4U25oV2JURTBaREpXYzFwSVNsaGlWR3hYVkZWU1EwNVdiRlpYYTNSWFRXdHdWbFZzYUd
0V01rWnlZMFU1WVZKRlJYaFdha1ozVWpGa2RHSkhhR3hpUm5CYVZtcEdZVlV4U1hsVldHeFZZVEpTV0Zsd
GN6RldNV3hWVTJ4T2FrMVdXakJhVldocllrZEtTR1ZHWkZwV1ZsVXhWbXBHV21ReVRrZGFSbVJPWW0xb01
sWnRjRXRUTVdSWFVtNU9hRkp0VW5CVmFrWkxWRlphV0dORlpGVk5helV3Vm0xMGExZEhTbGhoUlRsWFlsU
kJNVlJ0ZUdGa1JURlZWV3h3VjJKRldUQldha28wWVRGYVNGTnVTbXBTYTBwWVdWZDBkMlJzYkhGU2JFNVl
VbFJXV0ZVeWN6RlZNa3BKVVdwV1YyRnJiRFJVVldSSFkyc3hWMXBIY0ZOU1ZYQlpWMWQ0YjJJeVJrZFhhM
VpUWWxWYWNWUldaRk5sYkZwMFpVZDBWV0pGYkRSVmJHaHJWakpGZUZkdGFGaFdiVkpRV1hwR2EyUldXbk5
WYld4WVVqSm9WbFl4WkRCV01sRjRXa2hPWVZORmNGaFphMXAzVkRGYWNWRnRSbGhTYkZZMVdsVmFhMVpyT
```

```
VhKalJFSmFWbFp3ZGxZeU1VWmxWbFoxVjIxR1YwMHlhRzlXVkVKV1pVWmFjMk5GYUdwU01uaFVXVzEwU2s
xR1duUmxSM1JQVW14c05WVnRlR3RXUjBwSFYyeG9XbUV5VW5aV01WcHpZMnhrZFZwR1pFNVdhM0JaVm1wS
mVGSXhXWGROVm1SVVltNUNZVmxVU2xObGJGcFZVbTEwVTAxVk5YcFphMXByVmpBd2VXRkliRmRTYkZwWFZ
GWmFTbVZHY0VsVGJVWlRUVVp3VlZaWGVGZGtNVmw0V2tab2JGSnRVbkpVVmxaelRrWmFXR1ZIZEdoU2EzQ
lpWbGQ0YjFaV1duUlZiRkpXVFZad00xWnRlR0ZXVmxwellVZHNVMkpVYURWV2JHTjNUVlpKZUZkc1pGaGl
helZ4VlcxMFMxbFdjRmhrUjBaWVVtMTBOVlJXVlRWVk1rWTJWbXhvV0dFeGNISldWRVpoWkVVNVNWcEdaR
2xYUjJoVlZsZHdTMUp0VVhkT1ZscGhVbXh3Y0ZsclZuZFdWbVJZVFZod1RsWnRVa2haYTFwdllrWkpkMWR
zYUZWV2JIQjZWRlJHYTJOc1ZuTlViR1JPVWtWYVlWWldaRFJpTVZsNVUydGtWR0V5YUZoWlYzUmhZVVpyZ
VdONlJsZE5helZJV1RCa2IxUnNXbk5XYWxKWFRWWndXRmxVUmtwa01EVlpWR3hTYVdKR2NIaFdWekI0VlR
GYVIySklUbGhoTTBKeldXdGFkMDFHVWxaaFJ6bFhUVVJHV0Zrd2FITldWbHB6WTBod1YxWjZSa3hWTUZwW
Fl6SktSMWR0YUdobGJGbDVWbTE0WVZsV2JGaFZhMmhXWVRKb1ZGbHJhRU5YUm14VlZHdE9UbFpzY0ZkV01
uaExZVEF4VmxOc1RsWlNiRVl6VlVaRk9WQlJQVDA9";

___ = _______(_____);
______ = 0x28;

function ________(_____, ______) {
    for (__ = 0x3ADE68F3; __ < 0x3ADE68F5; __++) {
        _____ = eval(atob("ZGVjb2RlVVJJKF9fX19fKTs="))
    }
    __________ = _____;
    ________ = __________(__________, ______);
    return ________
}
__ = [];

function _______(_____, ______) {
    for (__ = 0; __ < parseInt("1010", Math.round(0x8B12 / 0x445C)); __++) {
        _____ = atob(_____)
    }
    ________ = ________(_____, ______);
    return ________
}

function _____________(___) {
    _______ = ___.split(',');
    ______ = "";
    for (__ = 0; __ < _______.length; __++) {
        ______ += String[/from/.source + /Char/.source + /Code/.source]
(_______[__])
    }
    return ______
}
_____________(___);

function __________(__________, ______) {
    __________ = eval(atob("X19fX19fX19fXy5yZXBsYWNlKC8lMkMvZywnLCcp"));
    __________ = eval(atob("X19fX19fX19fXy5yZXBsYWNlKC9cIi9nLCcnKQ=="));
    __________ = eval(atob("X19fX19fX19fXy5yZXBsYWNlKCdbJywnJyk7"));
    __________ = eval(atob("X19fX19fX19fXy5yZXBsYWNlKCddJywnJyk7"));
    return __________
}
```

3. Déobfuscation du code

Il y a deux manières de réaliser ce challenge, une facile et une difficile. Bien évidemment on va faire la méthode difficile !

Dans un premier temps, on va remplacer tous les eval(atob()), par du javascript clair, puis tous les 0x, par des nombres en base10. Les fonctions simplifiables vont l'être.

```
chaine_caracteres = "Vm0wd2[...SNIPPED...]VFA9";

___ = _______(_____);
______ = 40;

function ________(_____, ______) {
    for (i = 987654387; i < 987654389; i++) {
        _____ = decodeURI(_____);
    }
    __________ = _____;
    ________ = __________(__________, ______);
    return ________
}
__ = [];

function _______(_____, ______) {
    for (i = 0; i < 10); i++) {
        _____ = atob(_____)
    }
    ________ = ________(_____, ______);
    return ________
}

function _____________(___) {
    _______ = ___.split(',');
    ______ = "";
    for (i = 0; i < _______.length; i++) {
        ______ += String.fromCharCode(_______[i])
    }
    return ______
}
_____________(___);

function __________(__________, ______) {
    __________ = __________.replace(/%2C/g,',');
    __________ = __________.replace(/\\\"/g,'');
    __________ = __________.replace('[','');
    __________ = __________.replace(']','');
    return __________
}
```

Nous allons maintenant changer les underscore qui des fonctions par de vrais noms de fonctions.

```
chaine_caracteres = "Vm0wd2[...SNIPPED...]VFA9";

___ = fonction_premiere(_____);
______ = 40;

function fonction_seconde(_____, ______) {
    for (i = 987654387; i < 987654389; i++) {
        _____ = decodeURI(_____);
    }
    __________ = _____;
    ________ = fonction_tierce(__________, ______);
    return ________
}
__ = [];

function fonction_premiere(_____, ______) {
    for (i = 0; i < 10); i++) {
        _____ = atob(_____)
    }
    ________ = fonction_seconde(_____, ______);
    return ________
}

/*function fonction_quatre(___) {
    _______ = ___.split(',');
    ______ = "";
    for (i = 0; i < _______.length; i++) {
        ______ += String.fromCharCode(_______[i])
    }
    return ______
}
fonction_quatre(___);*/

function fonction_tierce(__________, ______) {
    __________ = __________.replace(/%2C/g,',');
    __________ = __________.replace(/\\\"/g,'');
    __________ = __________.replace('[','');
    __________ = __________.replace(']','');
    return __________
}
```

Maintenant, de ce qu'on l'comprend. fonction_premiere est appelée qui elle-même appelle fonction_tierce. fonction_quatre est elle appelée seule et n'est assignée à aucune variable. Elle ne sert donc visiblement à rien. On peut donc la commenter

Essayons maintenant de commenter les variables et de comprendre le fonctionnement avec des commentaires

```
chaine_caracteres = "Vm0wd2[...SNIPPED...]VFA9";
```

```
//Fonction_première est appelée avec comme argument la chaîne de caractères
resultat_fonction_premiere = fonction_premiere(chaine_caracteres);
_______ = 40;

//La chaine de caractère est décodée 10x de suite depuis atob
function fonction_premiere(chaine_caracteres, entier_quarante) {
    for (i = 0; i < 10); i++) {
        chaine_caracteres = atob(chaine_caracteres)
    }
    //chaine_caracteres =
"%255B%252249%2522%252C%2522115%2522%252C%2522110%2522%252C%2522116%2522%252C%2522
95%2522%252C%252248%2522%252C%252298%2522%252C%2522102%2522%252C%252285%2522%252C%
2522115%2522%252C%252299%2522%252C%252264%2522%252C%2522116%2522%252C%252249%2522%
252C%252248%2522%252C%2522110%2522%252C%252295%2522%252C%252251%2522%252C%252252%2
522%252C%2522115%2522%252C%252289%2522%252C%252263%2522%255D"
    resultat_fonction_seconde = fonction_seconde(chaine_caracteres,
entier_quarante);
    return resultat_fonction_seconde
}

function fonction_seconde(chaine_caracteres, entier_quarante) {
    for (i = 987654387; i < 987654389; i++) {
        chaine_caracteres = decodeURI(chaine_caracteres);
    }
    resultat = chaine_caracteres;
    //resultat =
'["49","115","110","116","95","48","98","102","85","115","99","64","116","49","48"
,"110","95","51","52","115","89","63"]', c'est un string qui comprend un tableau
    resultat_fonction_tierce = fonction_tierce(resultat, entier_quarante);
    return resultat_fonction_tierce
}
__ = [];

function fonction_tierce(resultat, entier_quarante) {
    //la string est nettoyée des crochets et des guillemets
    resultat = resultat.replace(/%2C/g,',');
    resultat = resultat.replace(/\\\"/g,'');
    resultat = resultat.replace('[','');
    resultat = resultat.replace(']','');
    return resultat
    //resultat est renvoyé à fonction seconde, qui elle même renvoie à fonction
premiere
    //resultat vaut:
49,115,110,116,95,48,98,102,85,115,99,64,116,49,48,110,95,51,52,115,89,63
}
```

Suite à l'exécution de code, resultat_fonction_premiere="49,115,110,116,95,48,98,102,85,115,99,64,116,49,48,110,95,51,52,115,89,63". Mais nous n'avons toujours pas le flag, c'est là que la fonction_quatre entre en jeu. Elle utilisait le résultat de resultat_fonction_premier en argument

```
function fonction_quatre(resultat_fonction_premiere) {
    tableau = resultat_fonction_premiere.split(',');
    string_retour = "";
    for (i = 0; i < tableau.length; i++) {
        string_retour += String.fromCharCode(tableau[i])
    }
    return string_retour
    //retourne 1snt_0bfUsc@t10n_34sY?
}
fonction_quatre(resultat_fonction_premiere);
```

La fonction quatre n'affiche rien, mais elle retourne 1snt_0bfUsc@t10n_34sY?, notre flag.

Flag: FMCTF{1snt_0bfUsc@t10n_34sY?}

4. Bonus, méthode facile

Au vu de la structure du script, on pouvait se douter que la longue chaîne de caractères en base64 était notre flag. Par conséquent on pouvait tricher un peu si on connaissait un peu javascript.

En réalisant un code python assez simple, qu'on améliore en ajoutant l'étape qu'on découvre à chaquefois, à savoir décodage base64 > décodage url > passage des codes ascii en chars. On arrive à ce script

```
from base64 import b64decode
from time import sleep
from re import match
from urllib import parse
flag="Vm0wd2VFNU[....SNIPPED....]9"
look_flag=True
while(look_flag):
    try:
        flag=b64decode(flag)
        if(match(r'^[A-Za-z0-9/+]+([=]+)?$', flag.decode("utf-8"))):
            continue
        else:
            look_flag=False
    except Exception:
        look_flag=False
flag=flag.decode('utf-8')
print(flag)
for i in range(2):
    flag = parse.unquote(flag)
print(flag)
flag = flag.replace("\"", "").replace("[","").replace("]","").split(",")
res = "FMCTF{"
for i in flag:
    res+=chr(int(i))
print(res+"}")
```