

Reversing Challenge #1

One of the monkeys lit a beacon for the others to follow, but we are having a hard time understanding the message. Can you translate this for us?

```
JHM9TmV3LU9iamVjdCBjTy5NZW1vcn1TdHJlYW0oLFtDb252ZXJ0XT06RnJvbUJhc2U2NFN0cmLuZygiSDRzSUFCQlU4R0VBLz
ZWV1hXOGFPeEi5Wm4rRkg1QjJWMTBRQ2IwYFyeEZLaUdRRUpWY2xLV2hEVVhJN0E3Z3htc1QyMHZndhZudkh1OEhKVzJpUmlv
djdJZG5mT2JNbWVNTndkUkNmWWhrQmpJR1Vyc0JwWmtVNU5oeHFtZX1iOGdKZWVjNmKxUkV4ajYyRjdNbG10bGF5V2hHNDfpQj
F1U3JVeGxTU1JQaVZUZlV6UklacHh3Q2t0M1loUkNuQ3Z4S3hbbGtqMUTOlFKbWdocTJnVmtDwmlWampSdDvR1Lo2Z1NZVH1z
UzAxZXFrU29FdytYmZlIRXhiYTbQbW5JSDJmUeT0akZlZ29QYmYvQXRFaG53bDFWbjluTXM1NWNXeVhZZEdLeXlVTFdMNzdyMk
1jRHNWnNvVHYU0rTzVueis3L3FSMk5LMTM3MVBldGVlR08yMGdxY2VjdXo1NTlPMkdvOTBhUEhmQUlpvZFYSm02bU1ubWnmMURo
djRqXQovSXNidCtVZGx5VGJHT2w0dTBXZk1ZejhYTElYTFR6amwwQXpLeCswMm1VL0p1aitZnkZZWwXVTzhMQTBxdVExQWJGb0
d1WDFBumM3aUdCWWE1R3RzbmxxN1BJQlNZVklrY0FHEj1STI4QTY4cVZNER6RHQ1YmQ2cGR3VVBKYm12RGZJT2czRFYwQ2cv
S0RUeEdqcFFnYWliUEIyVzh3djZBM0g1K1B0R1lMN3o2RHdqMVJnNExLbUJtVUYrRDdUcVZDcVQ3Qkt3SG04b05jdm1Ua2dqSU
FNRVFZMvVPOXZPa1VyQm4vN29UNzV0R2FtREZ4TWrsVkJGVE42ZUhhNY0ptZHHJRmsrZG110FU2ckhQWi9PVThSaVVMZi95TkP6
QmdnazQyd21hc0tnVS9CUDZ5NTdCZ3VPdEZYnJfU3QXB4ZW03eEF1S3pnaDBrSENYNGExZzNZV1lmZTVxRGewZ1lkNDJvVUJMK1
V6QjVEejIzTHdhUU1ILzVQY3EwdXNBGeGczSjFNvNE3Y25kN2I3WGM0V1RyZ0F4VG5QTW9JQ0ZRRG5GQTJrs3o0bFU3T1RLN3RC
TlR3QjJrM0xDSWfsT21tnVlEZVvocHNYVkhDch1ZTk1MdUlnMmpjQTBsbz15eUvWQUxGc1BwTG1UTEVzTEJkZ2VjZENqbk9IS1
1hWU05d1N1V2k5Q116U2hFKzVNKy9EcWFiRD1aYzBod2R1WkNQVTZYNkRuR1JHvnlvMHVJOXo3eURPeHlUdktoc0Z5VkpCMkFS
Z0dFWepxQTNEQmwwTmNzN3o4Sjc4L2dQYldZSnpBN0NvcEdldGtnVGs1M3hvnUX0akt5aHd0S3V1QX1ZMDRawksyblpISkt0Zn
o5MXA1Q111bTVvN3Z0VmrpOXBaOHVZajQrd1IxM2p5NWxUMXkxUDEzYzNIz1B1OWU5NUdZd2F0Nk9Qb1I1U011WG85dng1Y041
ODNMMWNYbH1ZaEV1cEVMUDJkbzUvcGZnZjQyYkhLM0ZnRHRqVWZiNW16ZS5sWUw5bTBsMU95Mj1lMz1mbTI4eFcvTXZXMCM8yWm
1QOS9YY3VPcUJLcn1qSDZ0RFVtA24ycE9vVjFqU1V6RVo0M3ZPbjZSMG9BUhQR2p5THlrYTJJPWmVSTmRNWfHBMnRQVGZjs1Fy
MkExNDJqNSs5OHZHUUXSYWlnK1kxemRQRk1qT2Nvc0xTZDh1RnJkXWRsbGZvSm1QeFBZawXUvdrclcwMkd1aVdqZTNiaHUrOG
5wYU9YtZ15Tjdmce1MNUlUaU01M01sbk8yRlhjeG1uSWptVTBaODI0TW1tdjZmV2twZDU5cDY2RE5EemZQbU8rODV4K29zZmhM
ZGFtdjJQWHL1Sd1QvN0p0S2NOVmFiM1JjN3g4eVdiUjY5S2ZkTHZmaVJWU2g1SkRjdHI2K114ZnNpb1pXcUhrK1NmWk4vSUEyVj
U0RGR5RFJIZ2tWcTdsSE5VS2FESDJ0UlpFcNZZWZ3TzRodm1IK01KQUFBPSIpKTtJRVggKE5ldy1PYmplY3QgSU8uU3RyZWft
UmVhZGVyKE5ldy1PYmplY3QgSU8uQ29tcHJlc3Npb24uR3ppcFN0cmVhbSgkcyxbSU8uQ29tcHJlc3Npb24uQ29tcHJlc3Npb2
5Nb2RlXT06R6GVjb21wcmVzcykKpKS5SZWFkVG9FbmQoKTS=
```

Solution

Overview

Contestants will perform reverse engineering on Base64 encoded PowerShell in the same format you would find a Cobalt Strike BEACON payload.

Using a tool like CyberChef, contestants need to identify the text above is Base64 and logically decode the next step using the instructions presented to them. At the very end of the decoding process they will find the flag.

monkeyCTF{5u0qK5PcCVI3B25FolUpsjqJApFAS8QB}

TECHNICAL DETAILS

Visit <https://gchq.github.io/CyberChef/> to perform the actions below.

The text presented in the challenge is Base64 encoded. Using CyberChef, decode the text with the From Base64 option.

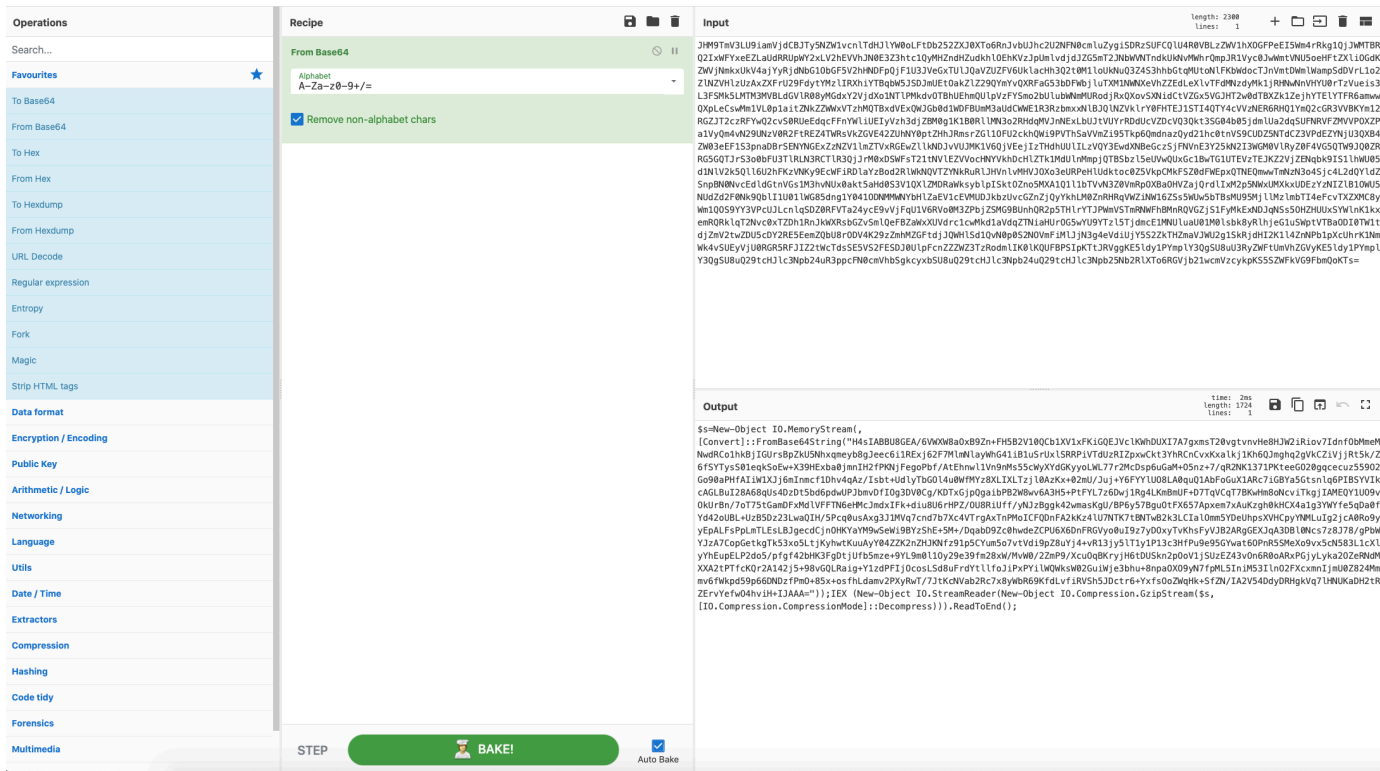


Figure 1: Base64 decode the original text

The contestants will find PowerShell with another layer of encoded text.

```
$s=New-Object
IO.MemoryStream(,[Convert]::FromBase64String("H4sIABBU8GEA/6VWxW8a0xB9Zn+FH5B2V10QCb1XV1xFKiG6EJVC
lKWhDUXI7A7gxmsT20vgtvnnvHe8HJW2iRiov7Idnf0bMmeMNwdRCo1hkBjIGUrsBpZkU5NhXqmeYb8gJeec6i1RExj62F7M1mN
layWhG41iB1uSrUxLSRRPiVTdUzRIZpxwCkt3YhRCnCxvKxalkj1Kh6QJmghq2gVkcZiVjJrt5k/Z6fSYTysS01eqkSoEw+X39
HEXba0jmnIH2fPKNjFegoPbf/AtEhnw11Vn9nM55vcYxyYdGKyyoLWL77r2McDsp6uGaM+O5nz+7/qR2NK1371PKteeG020gqc
ecuz55902Go90aPHFAi1W1XJj6mInmcf1DhV4qAz/Isbt+UdlyTbG014uWmHYz8XLIXL7zj10AZKX+02mU/Ju+Y6FYU08L
A0quQ1AbFoGuX1ARc7iGBYa5Gtsnlq6PIBSYVikcAGLBUi28A68qUs4DzDt5bd6pdwUPJbmVdFi0g3DV0Cg/KDtTgXjPQgaibPB
2W8wv6A3H5+PtFYL7z6Dwj1Rg4LKmBmUF+D7TqVCqT7BKwHm8oNcviTkgjIAEQY1U09v0kUrBn/7oT75tGamDFxMdlVFFTN6e
HMcJmdxIFk+diu8U6rHPZ/OU8RiUff/yNJzBgk42wmasKgU/BP6y57BguOtFX657ApXem7xAuKzgh0kHCX4a1g3YWyfe5qDa0f
fYd42oUBL+UzB5Dz23LwaQIH/5Pcq0usAxxg3J1MVq7cnd7b7Xc4VTrgAXtNPMoICFQDnFA2kKz41U7NTK7tBNTWb2k3LCIa1Om
m5YDeUhpSxVHCpyYNMLuIg2jCA0Ro9yyEpALFsPpLmTLEsLBjgecdCjnOHKYaYm9wSewi9BYzShE+5M+/DqabD9Zc0hwdZCPU
6X6DnFRGVyo0uI9z7yD0xyTvKhsFyVJB2ARgGEXJqA3DB10Ncs7z8J78/gPbWYJzA7COPGetkgTk53xo5LtjKyhwTkuuAyY04Z
ZK2nZHJKNfz91p5CYum507vtVdi9pZ8uYj4+vR13jy51T1y1P13c3HfPu9e95GYwat60PnR5SMExo9v5cN583L1cXlyYhE43vO
LP2do5/pfgf42bHK3FgDtiJufb5mze+9YL9m0110y29e39fm28Xw/MvW0/ZZmP9/Xcu0QbKryjH6tDvXk2n2p0oV1jSUZEZ4upOn
6R0oARxPGjyLyka20ZErNDMXXA2tPTfckQr2A142j5+98vGQLRaig+Y1zdPFIj0CocLSd8uFrdYt1lfoj1pXPyiLWQkwsW02Gu
iWje3bhu+8npaOX09yN7fpML5IniM53I1n02FXcxmIjmU0Z824Mmmv6fWkpd59p66DNDzfPmO+85x+osfHLdamv2PXYRwT/7J
tKcNVab2Rc7x8yWbR69KfdLvfiRVSh5JDctr6+YxfsOoZWqHk+SfZN/IA2V54DdyDRHqkVq71HNUKAhD2tRZErVYefw04hviH+
IJJAA="));IEX (New-Object IO.StreamReader(New-Object
IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
```

Figure 2: Decoded Base64

The text shown in Figure 2 presents PowerShell code that requires two steps to get to the next step. Contestants should identify `FromBase64String` and `GzipStream::Decompress` highlighted in red above. The code highlighted in yellow above needs to go into CyberShef, adding the `From Base64` and `Gunzip` options.

Operations

Search...

Favourites

Data format

Encryption / Encoding

Public Key

Arithmetic / Logic

Networking

Language

Utils

Date / Time

Extractors

Compression

Hashing

Code tidy

Forensics

Multimedia

Other

Flow control

Recipe

From Base64

Alphabet
A~Za-z0-9+/=

Remove non-alphabet chars

Gunzip

STEP

BAKE!

Auto Bake

Input

start: 1532

end: 1532

length: 1532

length: 0

lines: 1

H4sIABBU8GEA/6VW0W0a0x89Zn+FH5B2V1BQCb1XV1xFK1GQEJVC1W0hDUX17A7gmsT2bvgtnvH8HJW2IRlov71dnf0BmnefNdRCo1hK8JIGUrsBp2kX5Nhxqneyb8gJeece61lRExj6Z7F7M1nLaymH411B1u5rUX1SRRP1VTDZRTZpxwCk13YHRCnCxKaa1kj1Kh60Jmghq2gVkcZVJjR1t5k/26fSYTYS501eqk5eW+X39HExba0jenn1H2fPmNjFegofb7fAtEhm1Vn9Hm55CkyXYdGkyoLW77f2Mcdsp6p6Gm+05nz+7/gk2Nk1371PKteeG02Bqgcecu2559026g98aPHfAI1W1XJj6nEnncf1Dhv4qAz7zbtVdly7dG014ubHmYmRDXL7zj10Aks+02mUJuj+YFYYU08LAmqu01AafFouXJARC7fGBY5G5tn1qBP7BSYV1kAGL8uJ2B8A8RqJs4DZDf5DdfpawPjbmVf1Dg3DVKCg/KDTx6jpdga1bP82W0w6A3H5+P1FYL7z6DwJ5Rg6Lk0bLUF+077qCq7B8K6Hm8bKcV1TkgJ1AMEQY1U09v0kUrBn/7oT75tGambF4dLVFFTN6eHMcJmdx1Fk+d18U6fHPZ/U8R1Uf7fANJzBggk42masKgu/BP6y578gu0fX657Apem7AuKzgh0KHC4a1g3YWf5eQba0fYd42uBL+UzB50223LwaQ1H/SPC0usAxg3J1Mvq7cnd7b7Xc4VTrgAxTnPMo1CFQDnFA2Kkx4LU7N7K7tBN7u62K3LC1a10m5YDeUhpXVHCpyYmLuI2j/cABR09yEpaLFsPpLntLEs1B3gecdCjNOHcyayM9u5ew19B7r5He+9M+/Dqab09ZCdhudeZCPu6X0DnFRGyobu19z7yD0kyfVkhfYVJ32A8gGEXJq3D8LWmcs7z87B/gPmW7JzA7Cpogetkg7K53x0SLtjKyhetKuuY104ZKzn2bXWm7r5p3Cym507vVd19p2Bu74wV13j51T1yP13C3HfPue505wa60Hr5ShoKobv55n583L1cYhEupELP23u5jprf42bWCFjd1Uf55n2e+9YL9m81lDy29e39fnd28W/NvWb/2ZaP9/Xcu0qBKryJH61DUSkn2p0vU15JuzE243v0n6R0aArxPGjYkA202Z6N9M00A2LPTtCkQr2A142j5+98vGQLRa1gY1zdPFIj0cos1Sd8uFrDy1llfoJ1PxPY1LWQksW02Gu1Wje3bhu+8npa0X09yN7fPmLS1n1M5311n02FXcxm1jnu08249mmv6fmpds9p66NDzFPm0+85x+osfhlDamv2PxyRw77J1KckNvab2Rc7x8yWbR69KfDLv1RVSh53Dctr6+Yxfs0zWqHk+S7Zn/TAZV54DdyDRHgKvq71UNUKaDH21RZErvYefu04hv1H-IJAAA="

Output

time: 4ms

length: 2536

lines: 45

Set-StrictMode -Version 2

\$DoIt = @'

function func_get_proc_address {

Param (\$var_module, \$var_procedure)

\$var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { \$_.GlobalAssemblyCache -And \$_.Location.Split('\')[1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')

\$var_gpa = \$var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]])

@('System.Runtime.InteropServices.HandleRef', 'string')

return \$var_gpa.Invoke(\$null, @([System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr), (\$var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke(\$null, @(\$var_module)))), \$var_procedure))

}

function func_get_delegate_type {

Param (

[Parameter(Position = 0, Mandatory = \$True)] [Type[]] \$var_parameters,

[Parameter(Position = 1)] [Type] \$var_return_type = [Void]

)

\$var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate'))),

[System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', \$false).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass',

[System.MulticastDelegate])

\$var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public',

[System.Reflection.CallingConventions]::Standard,

\$var_parameters).SetImplementationFlags('Runtime, Managed')

Figure 3: Second stage decode in cyberchef

The following text will be presented

```

Set-StrictMode -Version 2

$DoIt = @'
function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object {
    $_.GlobalAssemblyCache -And $_.Location.Split('\')[1].Equals('System.dll')
    }).GetType('Microsoft.Win32.UnsafeNativeMethods')
    $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]])
    @('System.Runtime.InteropServices.HandleRef', 'string')
    return $var_gpa.Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-Object
    System.Runtime.InteropServices.HandleRef((New-Object IntPtr),
    ($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke($null, @($var_module)))),
    $var_procedure))
}

function func_get_delegate_type {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )

    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object
    System.Reflection.AssemblyName('ReflectedDelegate')),
    [System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule',
    $false).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass',
    [System.MulticastDelegate])
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public',
    [System.Reflection.CallingConventions]::Standard,
    $var_parameters).SetImplementationFlags('Runtime, Managed')

```

```

    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual',
$var_return_type, $var_parameters).SetImplementationFlags('Runtime, Managed')

    return $var_type_builder.CreateType()
}

[Byte[]]$var_code =
[System.Convert]::FromBase64String('TkxNSEZaYHd1WBZWE1JoFnNAYHVqEGERFmVMT3ZTUE1SaWJTZWJwG3JhXg==')

for ($x = 0; $x -lt $var_code.Count; $x++) {
    $var_code[$x] = $var_code[$x] -bxor 35
}

$var_va =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address
kernel32.dll VirtualAlloc), (func_get_delegate_type @([IntPtr], [UInt32], [UInt32], [UInt32])
([IntPtr])))
$var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)

$var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer,
(func_get_delegate_type @([IntPtr]) ([Void])))
$var_runme.Invoke([IntPtr]::Zero)
'@

If ([IntPtr]::size -eq 8) {
    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
    IEX $DoIt
}

```

Figure 4: Second stage decoded PowerShell

Inside the second stage of the decode, contestants are presented with PowerShell commonly found in Cobalt Strike BEACON. While most of the text above serves a function, the important information is to identify the last two decoding steps required for this challenge.

The PowerShell contains an additional Base64 string highlighted in yell. Attempting to decode this in CyberChef gives the following

The screenshot shows the CyberChef interface. On the left is a sidebar with various operation categories. The main area displays a recipe with two operations: 'From Base64' and 'Gunzip'. The 'From Base64' operation is selected, and its settings are visible. The input field contains the Base64 string: 'TkxNSEZaYHd1WBZWE1JoFnNAYHVqEGERFmVMT3ZTUE1SaWJTZWJwG3JhXg=='. The output field shows the result of the operations: 'NLMHFZ' weX.,V.,Rh.s@'uj'.a...eLdvSPIRiBsebp.ra^'. The output is garbled, indicating that the decoding steps were not sufficient to reveal the original message.

Figure 5: Third stage decode with bad results

The text decoded does not give a flag. The missing step is highlighted in red in Figure 4. Contestants need to identify that after Base64 decoding the text, it needs to be XOR'ed with a Key of 35. Entering the From Base64 and XOR option with a decimal key of 35, they will find the challenge flag.

Recipe	Input	start: 57 end: 58 length: 1
From Base64 Alphabet A-Za-z0-9+/=	'TkxNSEZaYHd'WBZWE1JoFnNAYHVqEGERFmVMT3ZTUE1SaWJTZWJwG3JhXg==	
<input checked="" type="checkbox"/> Remove non-alphabet chars		
XOR		
Key 35	DECIMAL	
Scheme Standard	<input type="checkbox"/> Null preserving	
Output start: 43 end: 43 length: 0 monkeyCTF{5u0qK5PcCVI3B25Fo\UpsjqJApFAS8QB}		

Figure 6: Decoding the last stage for the flag

The flag identified is **monkeyCTF{5u0qK5PcCVI3B25Fo\UpsjqJApFAS8QB}**.