MANDIANT
YOUR CYBERSECURITY ADVANTAGE

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?

```
JHM9TmV3LU9iamVjdCBJTy5NZW1vcnlTdHJlYW0oLFtDb252ZXJ0XTo6RnJvbUJhc2U2NFN0cmluZygiSDRzSUFCQlU4R0VBLz
ZWV1hXOGFPeEI5Wm4rRkg1QjJWMTBRQ2IxWFYxeEZLaUdRRUpWY2xLV2hEVVhJN0E3Z3htc1QyMHZndHZudkhlOEhKVzJpUmlv
djdJZG5mT2JNbWVNTndkUkNvMWhrQmpJR1Vyc0JwWmtVNU5oeHFtZXliOGdKZWVjNmkxUkV4ajYyRjdNbG1ObGF5V2hHNDFpQj
F1U3JVeGxTUlJQaVZUZFV6UklacHh3Q2t0M1loUkNuQ3Z4S3hhbGtqMUtoNlFKbWdocTJnVmtDWmlWampSdDVrL1o2ZlNZVHlz
UzAxZXFrU29FdytYMzlIRXhiYTBqbW5JSDJmUEtOakZlZ29QYmYvQXRFaG53bDFWbjluTXM1NWNXeVhZZEdLeXlvTFdMNzdyMk
1jRHNwNnVHYU0rTzVueis3L3FSMk5LMTM3MVBLdGVlR08yMGdxY2VjdXo1NTlPMkdvOTBhUEhmQUlpVzFYSmo2bUlubWNmMURo
djRxQXovSXNidCtVZGx5VGJHT2w0dTBXZk1ZejhYTElYTFR6amwwQXpLeCswMm1VL0p1aitZNkZZWWxVTzhMQTBxdVExQWJGb0
d1WDFBUmM3aUdCWWE1R3RzbmxxNlBJQlNZVklrY0FHTEJ1STI4QTY4cVVzNER6RHQ1YmQ2cGR3VVBKYm12RGZJT2czRFYwQ2cv
S0RUeEdqcFFnYWliUEIyVzh3djZBM0g1K1B0RllMN3o2RHdqMVJnNExLbUJtVUYrRDdUcVZDcVQ3Qkt3SG04b05jdmlUa2dqSU
FNRVFZMVVPOXZPa1VyQm4vN29UNzV0R2FtREZ4TWRsVkZGVE42ZUhNY0ptZHhJRmsrZGl1OFU2ckhQWi9PVThSaVVmZi95Tkp6
QmdnazQyd21hc0tnVS9CUDZ5NTdCZ3VPdEZYNjU3QXB4ZW03eEF1S3pnaDBrSENYNGExZzNZV1lmZTVxRGEwZllkNDJvVUJMK1
V6QjVEejIzTHdhUUlILzVQY3EwdXNBeGczSjFNVnE3Y25kN2I3WGM0VlRyZ0F4VG5QTW9JQ0ZRRG5GQTJrS3o0bFU3TlRLN3RC
TlR3QjJrM0xDSWFsT21tNVlEZVVocHNYVkhDcHlZTk1MdUlnMmpjQTBSbzl5eUVwQUxGc1BwTG1UTEVzTEJKZ2VjZENqbk9IS1
lhWU05d1NlV2k5Qll6U2hFKzVNKy9EcWFiRDlaYzBod2RlWkNQVTZYNkRuRlJHVnlvMHVJOXo3eURPeHlUdktoc0Z5VkpCMkFS
Z0dFWEpxQTNEQmwwTmNzN3o4Sjc4L2dQYldZSnpBN0NvcEdldGtnVGs1M3hvNUx0akt5aHd0S3V1QXlZMDRaWksyblpISktOZn
o5MXA1Q1l1bTVvN3Z0VmRpOXBaOHVZajQrdlIxM2p5NWxUMXkxUDEzYzNIZlB1OWU5NUdZd2F0Nk9QblI1U01lWG85dng1Y041
ODNMMWNYbHlZaEV1cEVMUDJkbzUvcGZnZjQyYkhLM0ZnRHRqVWZiNW16ZSs5WUw5bTBsMU95MjllMzlmbTI4eFcvTXZXMC8yWm
1QOS9YY3VPcUJLcnlqSDZ0RFVTa24ycE9vVjFqU1V6RVo0M3ZPbjZSMG9BUnhQR2p5THlrYTJPWmVSTmRNWFhBMnRQVGZjS1Fy
MkExNDJqNSs5OHZHUUxSYWlnK1kxemRQRklqT2Nvc0xTZDh1RnJkWXRsbGZvSmlQeFBZaWxXUVdrc1cwMkd1aVdqZTNiaHUrOG
5wYU9YTzl5TjdmcE1MNUluaU01M0lsbk8yRlhjeG1uSWptVTBaODI0TW1tdjZmV2twZDU5cDY2RE5EemZQbU8rODV4K29zZmhM
ZGFtdjJQWHlSd1QvN0p0S2NOVmFiMlJjN3g4eVdiUjY5S2ZkTHZmaVJWU2g1SkRjdHI2K1l4ZnNPb1pXcUhrK1NmWk4vSUEyVj
U0RGR5RFJIZ2tWcTdsSE5VS2FESDJ0UlpFcnZZZWZ3TzRodmlIK0lKQUFBPSIpKTtJRVggKE5ldy1PYmplY3QgSU8uU3RyZWFt
UmVhZGVyKE5ldy1PYmplY3QgSU8uQ29tcHJlc3Npb24uR3ppcFN0cmVhbSgkcyxbSU8uQ29tcHJlc3Npb24uQ29tcHJlc3Npb2
5Nb2RlXTo6RGVjb21wcmVzcykpKS5SZWFkVG9FbmQoKTs=
```

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 prcoess 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.

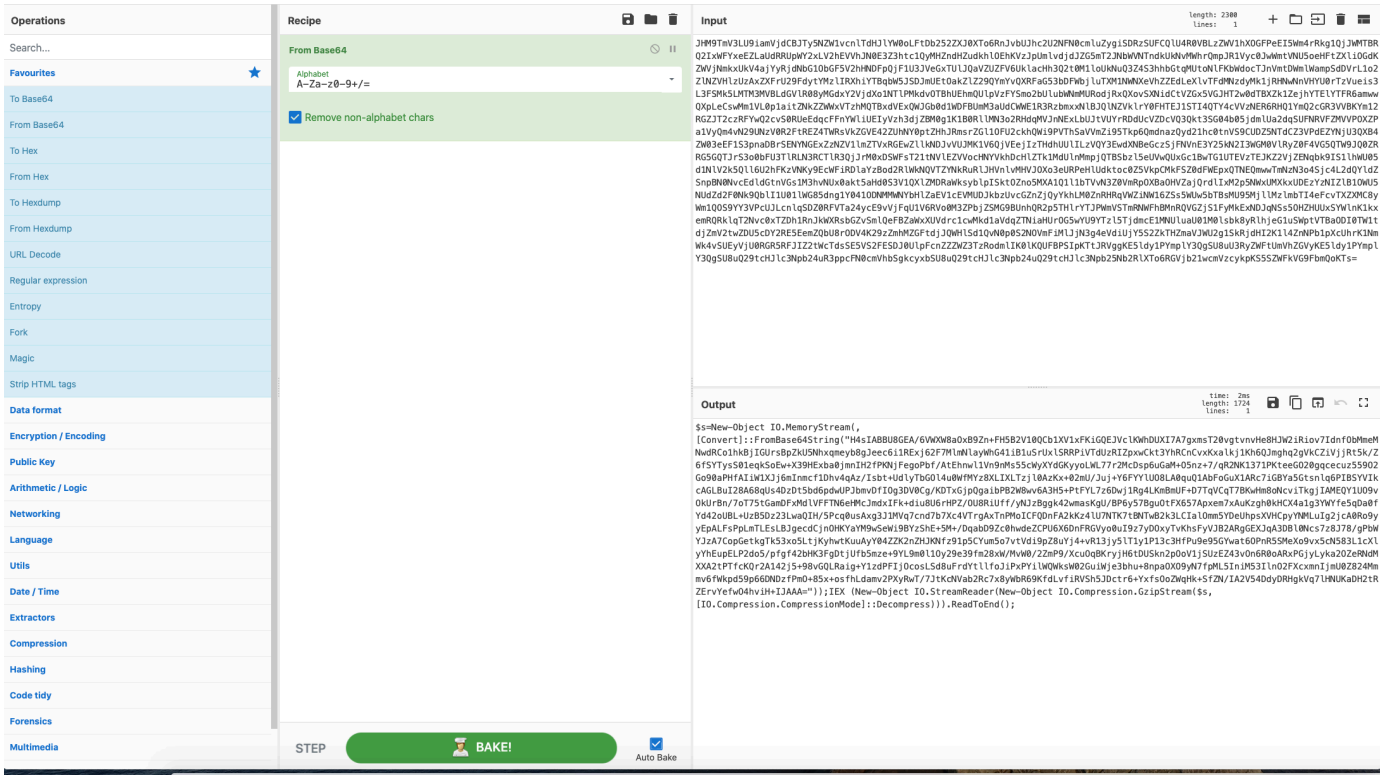

Figure 1: Base64 decode the original text

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

```
$s=New-Object
IO.MemoryStream(,[Convert]::FromBase64String("H4sIABBU8GEA/6VWXW8aOxB9Zn+FH5B2V10QCb1XV1xFKiGQEJVc
lKWhDUXI7A7gxmsT20vgtvnvHe8HJW2iRiov7IdnfObMmeMNwdRCo1hkBjIGUrsBpZkU5Nhxqmeyb8gJeec6i1RExj62F7MlmN
layWhG41iB1uSrUxlSRRPiVTdUzRIZpxwCkt3YhRCnCvxKxalkj1Kh6QJmghq2gVkCZiVjjRt5k/Z6fSYTysS01eqkSoEw+X39
HExba0jmnIH2fPKNjFegoPbf/AtEhnwl1Vn9nMs55cWyXYdGKyyoLWL77r2McDsp6uGaM+O5nz+7/qR2NK1371PKteeGO20gqc
ecuz559O2Go90aPHfAIiW1XJj6mInmcf1Dhv4qAz/Isbt+UdlyTbGOl4u0WfMYz8XLIXLTzjl0AzKx+02mU/Juj+Y6FYYlUO8L
A0quQ1AbFoGuX1ARc7iGBYa5Gtsnlq6PIBSYVIkcAGLBuI28A68qUs4DzDt5bd6pdwUPJbmvDfIOg3DV0Cg/KDTxGjpQgaibPB
2W8wv6A3H5+PtFYL7z6Dwj1Rg4LKmBmUF+D7TqVCqT7BKwHm8oNcviTkgjIAMEQY1UO9vOkUrBn/7oT75tGamDFxMdlVFFTN6e
HMcJmdxIFk+diu8U6rHPZ/OU8RiUff/yNJzBggk42wmasKgU/BP6y57BguOtFX657Apxem7xAuKzgh0kHCX4a1g3YWYfe5qDa0
fYd42oUBL+UzB5Dz23LwaQIH/5Pcq0usAxg3J1MVq7cnd7b7Xc4VTrgAxTnPMoICFQDnFA2kKz4lU7NTK7tBNTwB2k3LCIalOm
m5YDeUhpsXVHCpyYNMLuIg2jcA0Ro9yyEpALFsPpLmTLEsLBJgecdCjnOHKYaYM9wSeWi9BYzShE+5M+/DqabD9Zc0hwdeZCPU
6X6DnFRGVyo0uI9z7yDOxyTvKhsFyVJB2ARgGEXJqA3DBl0Ncs7z8J78/gPbWYJzA7CopGetkgTk53xo5LtjKyhwtKuuAyY04Z
ZK2nZHJKNfz91p5CYum5o7vtVdi9pZ8uYj4+vR13jy5lT1y1P13c3HfPu9e95GYwat6OPnR5SMeXo9vx5cN583L1cXl
yYhEupELP2do5/pfgf42bHK3FgDtjUfb5mze+9YL9m0l1Oy29e39fm28xW/MvW0/2ZmP9/XcuOqBKryjH6tDUSkn2pOoV1jSUzEZ43vOn
6R0oARxPGjyLyka2OZeRNdMXXA2tPTfcKQr2A142j5+98vGQLRaig+Y1zdPFIjOcosLSd8uFrdYtllfoJiPxPYilWQWksW02Gu
iWje3bhu+8npaOXO9yN7fpML5IniM53IlnO2FXcxmnIjmU0Z824Mmmv6fWkpd59p66DNDzfPmO+85x+osfhLdamv2PXyRwT/7J
tKcNVab2Rc7x8yWbR69KfdLvfiRVSh5JDctr6+YxfsOoZWqHk+SfZN/IA2V54DdyDRHgkVq7lHNUKaDH2tRZErvYefwO4hviH+
IJAAA="));IEX (New-Object IO.StreamReader(New-Object
IO.Compression.GzipStream($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.

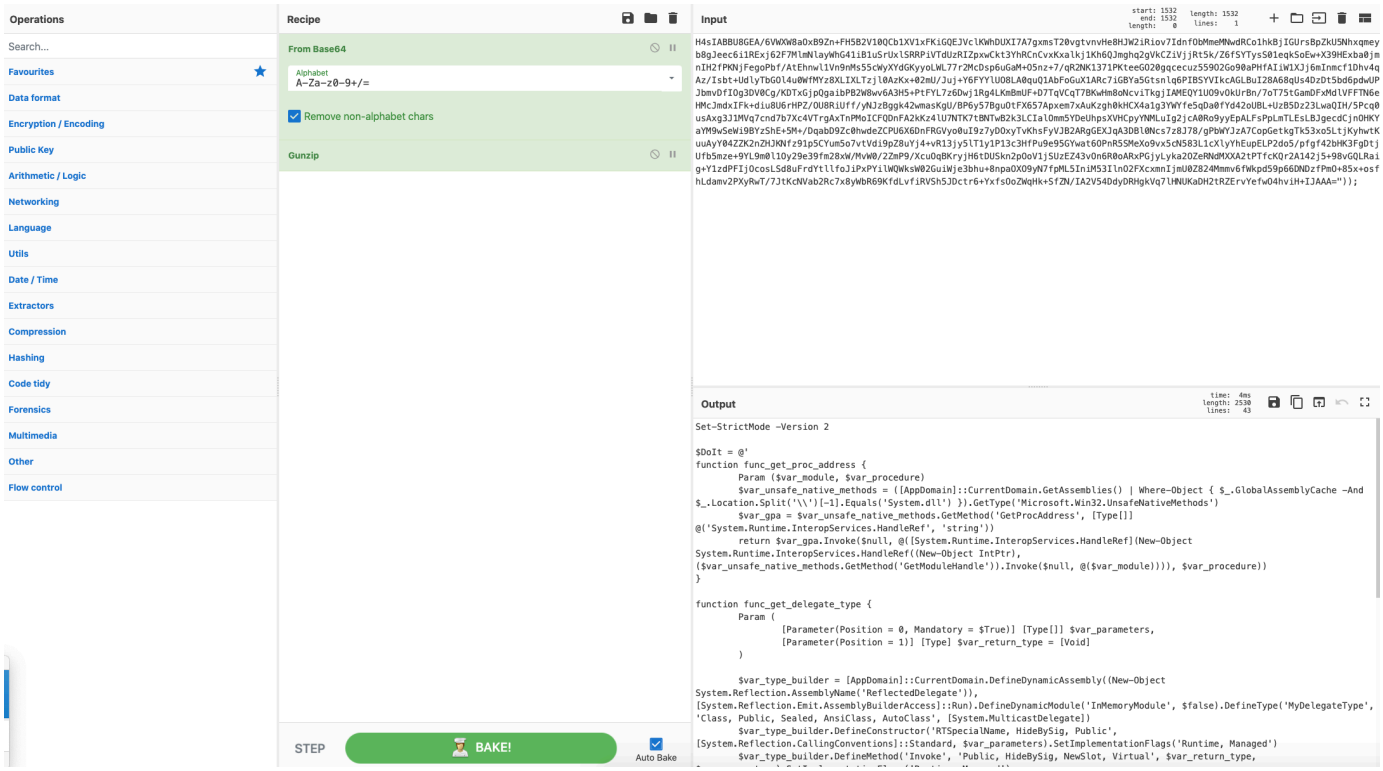

Figure 3: Second stage decode in cyberchef

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('TkxNSEZaYHdlWBZWE1JoFnNAYHVqEGERFmVMT3ZTUElSaWJTZWJwG3JhXg==')

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 servers a function, the important information is to iddentify the last two decoding steps required for this challenge.

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

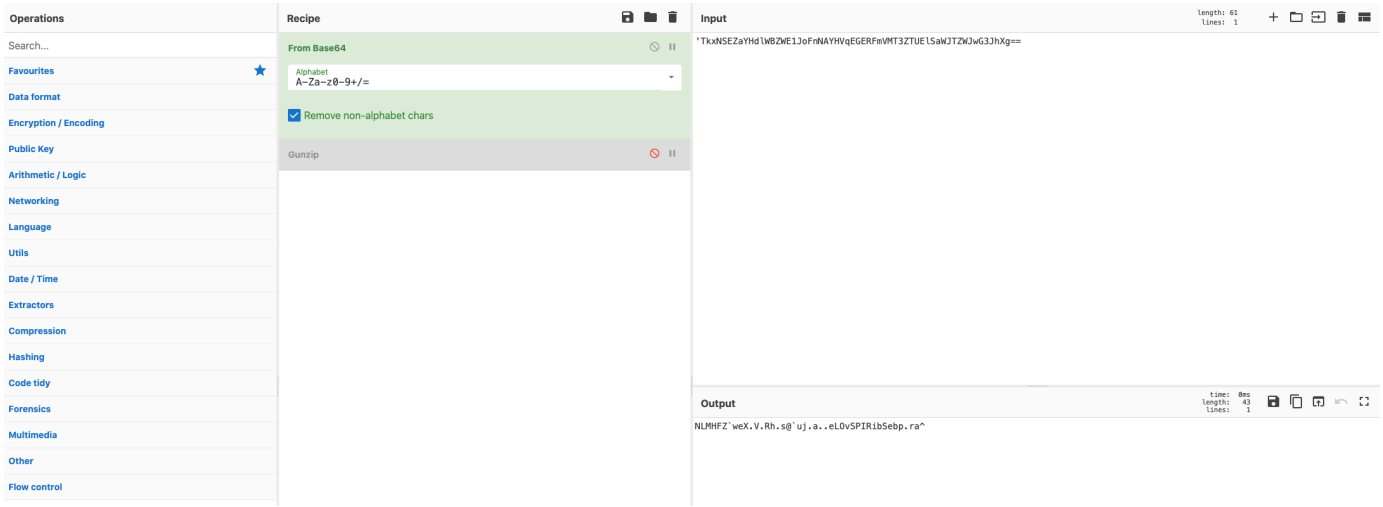

Figure 5: Third stage decode with bad results

The text decoded does not give a flag. The missing step is highligted in red in Figure 4. Contestents 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.

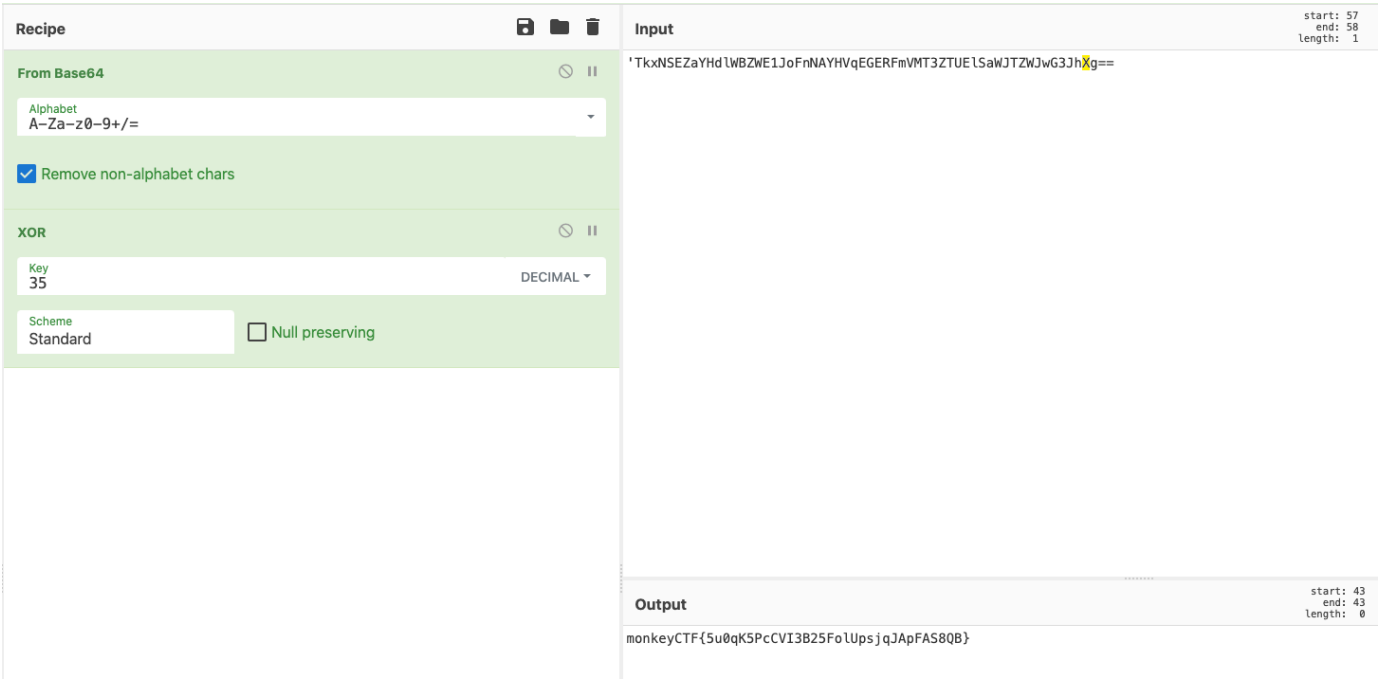

Figure 6: Decoding the last stage for the flag

The flag identified is **monkeyCTF{5u0qK5PcCVI3B25FolUpsjqJApFAS8QB}**.