

Anti-UPX Unpacking Technique - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2022-03-14 · Archived: 2026-04-05 17:56:08 UTC

- [IoT](#)

Malware targeting Windows OS (PE format) has a variety of obfuscation and packing techniques in place so that they complicate the code analysis processes. On the other hand, there are only a few types of packing techniques for Linux-targeting malware (ELF format), and it is mainly UPX-based. This blog article explains the details of Anti-UPX Unpacking technique, which is often applied to Linux-targeting malware.

Malware with Anti-UPX Unpacking Technique

The most well-known malware using Anti-UPX Unpacking technique is Mirai and its variants, which target IoT devices. Figure 1 shows the headers of UPX-packed binary and Mirai. The normal UPX packing uses “UPX!” as a magic number, while Mirai assigns a different value to each sample.

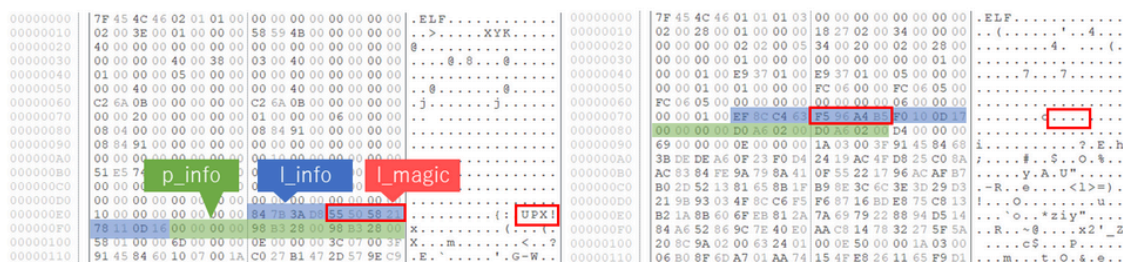


Figure 1: UPX-packed binary (left) and Mirai header (right)

UPX-packed binary contains the following information in the header. Normally, only “l_magic” is altered, but “p_filesize” and “p_blocksize” are also zero-padded in some samples.

```

struct l_info // 12-byte trailer in header for loader (offset 116)
{
    uint32_t l_checksum;
    uint32_t l_magic; // magic number = "UPX!"
    uint16_t l_lsize;
    uint8_t l_version;
    uint8_t l_format;
};

struct p_info // 12-byte packed program header follows stub loader
{
    uint32_t p_progid;
    uint32_t p_filesize;
};
    
```

```
uint32_t p_blocksize;
};
```

Besides Mirai, there are many other types of malware using this technique, including BoSSaBot (seen around 2014), as well as some coin miners and SBIDIOT malware, more recently. This is also applied to some types of malware which were used by Lazarus group. Figure 2 shows a part of ELF-VSingle’s code, which is associated with the group. The magic number is replaced with “MEMS”.

```
00000000 | 7F 45 4C 46 01 01 01 03 | 00 00 00 00 00 00 00 00 | .ELF.....
00000010 | 02 00 03 00 01 00 00 00 | 28 26 08 08 34 00 00 00 | ..... (&.4...
00000020 | 00 00 00 00 00 00 00 00 | 34 00 20 00 03 00 28 00 | .....4. .... (
00000030 | 00 00 00 00 01 00 00 00 | 00 00 00 00 00 80 04 08 | .....
00000040 | 00 80 04 08 40 AE 03 00 | 40 AE 03 00 05 00 00 00 | .....@...@.....
00000050 | 00 10 00 00 01 00 00 00 | 00 00 00 00 00 30 08 08 | .....0...
00000060 | 00 30 08 08 00 00 00 00 | 44 C8 07 00 06 00 00 00 | .0.....D.....
00000070 | 00 10 00 00 51 E5 74 64 | 00 00 00 00 00 00 00 00 | .....Q.td.....
00000080 | 00 00 00 00 00 00 00 00 | 00 00 00 00 06 00 00 00 | .....
00000090 | 10 00 00 00 B7 C7 69 BA | 4D 45 4D 53 24 08 0D 0C | .....i MEMS$...
000000A0 | 00 00 00 00 34 26 0B 00 | 34 26 0B 00 D4 00 00 00 | .....4&.4&.....
000000B0 | 79 00 00 00 08 00 00 00 | 77 1F A4 F9 7F 45 4C 46 | y.....w....ELF
000000C0 | 01 00 02 00 03 00 1B 32 | 88 04 F6 BF BD DF 08 34 | .....2.....4
000000D0 | 0E 64 23 0B 2F 16 20 00 | 05 00 28 00 12 00 11 00 | .d#./... (. ....
000000E0 | 5B BB CA 3B 3B 80 46 07 | 70 E8 0A 00 05 27 10 00 | [.;;.F.p...'.
000000F0 | 6D DB FD 7D 3F A4 1E A4 | 78 0F 08 07 18 3A 1E A0 | m..}?...x.....
00000100 | 7F 06 06 EB DC 42 6E 3F | 07 2D 09 04 07 EE EC 6A | .....Bn?-. ....j
00000110 | 7F 51 E5 74 64 00 01 06 | 7D 00 52 3E E0 B2 ED C0 | .Q.td...}.R>....
00000120 | 7F 5C 37 26 5C 37 77 F7 | 92 24 49 92 00 00 00 A0 | .\7&\7w..$I.....
00000130 | FF 9C E7 0A 00 52 8D 03 | 00 08 46 0D 00 97 F6 6F | .....R....F....o
00000140 | FF 83 EC 0C E8 0D 00 08 | B5 08 07 76 19 83 C4 0C | .....v.....
00000150 | C3 00 01 37 7F 7F DF F7 | 06 88 F9 08 8A 57 40 73 | ...7.....W@s
00000160 | 02 22 FF 74 24 1C 16 07 | 3E 66 90 DE DE FD EF 57 | ." .t$.>f.....W
```

Figure 2: ELF_VSingle header

Unpacking Anti-UPX Unpacking binary

Binary based on Anti-UPX Unpacking technique cannot be unpacked using the normal upx command. However, it is actually easy to unpack it. In most cases, the only change made to such binary is its magic number “UPX!”. You can unpack it with upx command by changing this value back to “UPX!”. Figure 3 shows the process of changing the magic number in order to unpack it using upx command.

```

/cygdrive/c/data/upx$ xxd sample |head
00000000: 7f45 4c46 0101 0103 0000 0000 0000 0000  .ELF.....
00000010: 0200 2800 0100 0000 1827 0200 3400 0000  ..(.....'..4...
00000020: 0000 0000 0202 0005 3400 2000 0200 2800  .....4. ...(.
00000030: 0000 0000 0100 0000 0000 0000 0000 0100  .....
00000040: 0000 0100 e937 0100 e937 0100 0500 0000  ....7...7.....
00000050: 0000 0100 0100 0000 fc06 0000 fc06 0500  .....
00000060: fc06 0500 0000 0000 0000 0000 0600 0000  .....
00000070: 0000 0100 ef8c c463 f596 a4b5 f010 0d17  .....c.....
00000080: 0000 0000 d0a6 0200 d0a6 0200 d400 0000  .....
00000090: 6900 0000 0e00 0000 1a03 003f 9145 8468  i.....?.E.h

/cygdrive/c/data/upx$ xxd sample |tail -n 4
00013960: abee a5ef 0000 0000 00f5 96a4 b500 0000  .....
00013970: 0000 0000 f596 a4b5 0d17 0e0a 0a81 5265  .....Re
00013980: 2d24 2d4a 9c03 0000 5101 0000 d0a6 0200  -$-J....Q.....
00013990: 5000 0018 8000 0000  P.....

/cygdrive/c/data/upx$ upx -d sample
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2020
UPX 3.96w Markus Oberhumer, Laszlo Molnar & John Reiser Jan 23rd 2020

File size      Ratio      Format      Name
-----
upx: sample: NotPackedException: not packed by UPX

Unpacked 0 files.

/cygdrive/c/data/upx$ vi -b sample
/cygdrive/c/data/upx$ xxd sample |head
00000000: 7f45 4c46 0101 0103 0000 0000 0000 0000  .ELF.....
00000010: 0200 2800 0100 0000 1827 0200 3400 0000  ..(.....'..4...
00000020: 0000 0000 0202 0005 3400 2000 0200 2800  .....4. ...(.
00000030: 0000 0000 0100 0000 0000 0000 0000 0100  .....
00000040: 0000 0100 e937 0100 e937 0100 0500 0000  ....7...7.....
00000050: 0000 0100 0100 0000 fc06 0000 fc06 0500  .....
00000060: fc06 0500 0000 0000 0000 0000 0600 0000  .....
00000070: 0000 0100 ef8c c463 5550 5821 f010 0d17  .....UPX!...
00000080: 0000 0000 d0a6 0200 d0a6 0200 d400 0000  .....
00000090: 6900 0000 0e00 0000 1a03 003f 9145 8468  i.....?.E.h

/cygdrive/c/data/upx$ xxd sample |tail -n 4
00013960: abee a5ef 0000 0000 0055 5058 2100 0000  .....UPX!...
00013970: 0000 0000 5550 5821 0d17 0e0a 0a81 5265  ...UPX!.....Re
00013980: 2d24 2d4a 9c03 0000 5101 0000 d0a6 0200  -$-J....Q.....
00013990: 5000 0018 8000 0000  P.....

/cygdrive/c/data/upx$ upx -d sample
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2020
UPX 3.96w Markus Oberhumer, Laszlo Molnar & John Reiser Jan 23rd 2020

File size      Ratio      Format      Name
-----
173776 <-    80280    46.20%    linux/arm    sample

Unpacked 1 file.

```

Figure 3: Example of unpacking binary

We have created a tool that enables unpacking binary with Anti-UPX Unpacking techniques. This tool is intended for this purpose only, and it may not work otherwise.

JPCERTCC/upx-mod - GitHub <https://github.com/JPCERTCC/upx-mod/releases/tag/v4.00-beta>

```

/cygdrive/c/data/upx$ xxd sample |head
00000000: 7f45 4c48 0101 0103 0000 0000 0000 0000  .ELF.....
00000010: 0200 2800 0100 0000 1827 0200 3400 0000  ..(.....'.4...
00000020: 0000 0000 0202 0005 3400 2000 0200 2800  .....4. ...(.
00000030: 0000 0000 0100 0000 0000 0000 0000 0100  .....
00000040: 0000 0100 e937 0100 e937 0100 0500 0000  ....7...7.....
00000050: 0000 0100 0100 0000 fc08 0000 fc08 0500  .....
00000060: fc08 0500 0000 0000 0000 0000 0600 0000  .....
00000070: 0000 0100 ef8c c463 f596 a4b5 f010 0d17  .....c.....
00000080: 0000 0000 d0a6 0200 d0a6 0200 d400 0000  .....
00000090: 6900 0000 0e00 0000 1a03 003f 9145 8468  i.....?.E.h

/cygdrive/c/data/upx$ upx -d sample
          Ultimate Packer for eXecutables
          Copyright (C) 1996 - 2020
UPX 3.96w      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

-----
File size      Ratio      Format      Name
-----
upx: sample: NotPackedException: not packed by UPX

Unpacked 0 files.

/cygdrive/c/data/upx$ upx_mod -d sample
          Ultimate Packer for eXecutables
          Copyright (C) 1996 - 2021
UPX 4.0.0      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 1st 2021

-----
File size      Ratio      Format      Name
-----
174960 <-    80280    45.88%    linux/arm    sample

Unpacked 1 file.

```

Figure 4: Sample use of upx_mod command

Detect Anti-UPX Unpacking Technique

Binary packed with this technique can be identified manually, just by looking at the code. In order to avoid oversight, we recommend Yara-based automatic detection like below. This rule does not detect binary packed with normal UPX.

```

rule upx_antiunpack_elf32 {
  meta:
    description = "Anti-UPX Unpacking technique to magic renamed for ELF32"
    author = "JPCERT/CC Incident Response Group"

  condition:
    uint32(0) == 0x464C457F and
    uint8(4) == 1 and
    (
      (
        for any magic in (uint32(filesize - 0x24)) : (magic == uint32(uint16(0x2C) * uint16(0x2A))
        not for any magic in (0x21585055, 0) : (magic == uint32(uint16(0x2C) * uint16(0x2A)) + uin
      )
    )
    or
    (

```

```
        for any magic in (uint32(filesize - 0x24)) : (magic == uint32(uint16be(0x2C) * uint16be(0x2A) +
        not for any magic in (0x21585055, 0) : (magic == uint32(uint16be(0x2C) * uint16be(0x2A) +
    )
    )
}

rule upx_antiunpack_elf64 {
    meta:
        description = "Anti-UPX Unpacking technique to magic renamed for ELF64"
        author = "JPCERT/CC Incident Response Group"

    condition:
        uint32(0) == 0x464C457F and
        uint8(4) == 2 and
        (
            (
                for any magic in (uint32(filesize - 0x24)) : (magic == uint32(uint16(0x36) * uint16(0x38)
                not for any magic in (0x21585055, 0) : (magic == uint32(uint16(0x36) * uint16(0x38) + uint16(0x38)
            )
            or
            (
                for any magic in (uint32(filesize - 0x24)) : (magic == uint32(uint16be(0x36) * uint16be(0x38)
                not for any magic in (0x21585055, 0) : (magic == uint32(uint16be(0x36) * uint16be(0x38) + uint16be(0x38)
            )
        )
}
```

In closing

Many attack groups use malware based on Anti-UPX Unpacking technique. It is easy to unpack such malware, but you may waste your time in unpacking process unless you notice this feature beforehand. When you analyse packed ELF binary, we recommend checking first whether it uses Anti-UPX Unpacking technique.

Shusei Tomonaga

(Translated by Yukako Uchida)



[朝長 秀誠 \(Shusei Tomonaga\)](#)

Since December 2012, he has been engaged in malware analysis and forensics investigation, and is especially involved in analyzing incidents of targeted attacks. Prior to joining JPCERT/CC, he was engaged in security

monitoring and analysis operations at a foreign-affiliated IT vendor. He presented at CODE BLUE, BsidesLV, BlackHat USA Arsenal, Botconf, PacSec and FIRST Conference. JSAC organizer.

Related articles

```

*key = 0x427c7480;
*key[4] = 0x015913c2;
*key[8] = 0x0d472834;
*key[12] = 0x00007909;
*key[16] = 0x1379421;
*key[20] = 0x40003468;
*key[24] = 0x00798129;
*key[28] = 0x00000007;
v4 = m_ret_argOffset0x350(a1 + 3);
if ( !((v3 = <CryptAcquireContext>)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x1, 0xf000000) )
return 0;
v5 = m_ret_argOffset0x350(a1 + 3);
handlehash0 = a1 + 3;
if ( !((v3 = <CryptCreateHash>)(*a1, 0x0004, 0, 0, a1 + 3) )
{
LABEL_0:
if ( *a1 )
return 0;
v6 = m_ret_argOffset0x350(a1 + 3);
(v4 = <CryptReleaseContext>)(*a1, 0);
return 0;
}
if ( !(<CryptHashData>)(*handlehash0, key, 16, 0) )
{
v6 = m_ret_argOffset0x350(a1 + 3);
v8 = a1 + 3;
!(v8 = <CryptDeriveKey>)(*a1, 0x0004, *handlehash0, 0x000000, a1 + 2)) // CAB_AES_128
{
if ( *handlehash0 )
{
v6 = m_ret_argOffset0x350(a1 + 3);
(v5 = <CryptDestroyHash>)(*handlehash0);
}
goto LABEL_0;
}
v9 = m_ret_argOffset0x350(a1 + 3);
(v10 = <CryptSetKeyParam>)(*v9, 3, 0x0001, 0); // SP_PAD010 = PKCS017
v11 = m_ret_argOffset0x350(a1 + 3);
(v11 = <CryptSetKeyParam>)(*v11, 1, 0x, 0); // IV = parameter
v12 = m_ret_argOffset0x350(a1 + 3);
(v12 = <CryptSetKeyParam>)(*v12, 0, 0x0002, 0); // SP_PAD0 = CBC
return *v9;
}

```

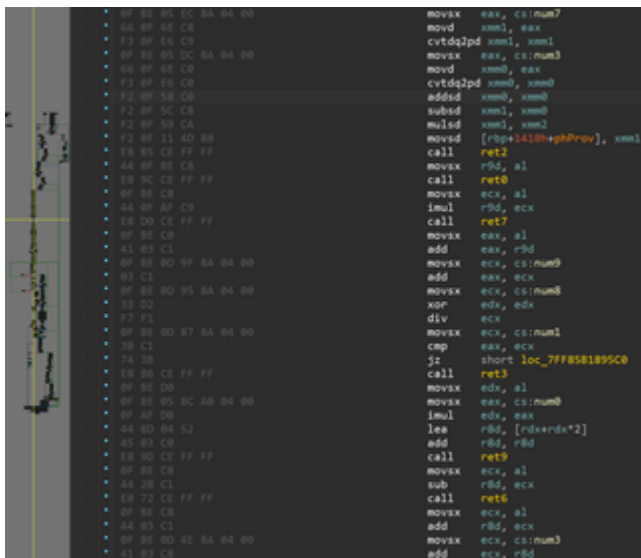
[Update on Attacks by Threat Group APT-C-60](#)

```

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 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 MA0GCSqGSIB3DQEB
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 gQCNS381HP2V3JD4
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcaLhAkpM0QAG
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 +IzYpXnMU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 q+cRxMoLmHNoq2U
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 sIB/Swnc3wXub0a
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZwmHU3wID
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-----
MIGFMA0GCSqGSIB3DQEBAQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcaLhAkpM0QAGRn6Nw6
RHnVST/1HJ+zHLH82q7XKmo+RU+IzYpXnMU7pMs1Sdq+cRxMoLmHNoq2UTWk9o9RodczTZxsk
bM7Tzk7UZjyapTIJfcq6BwMdsMx6gH40s1B/Swnc3wXub0aqEokKorZwmHU3wIDAQAB
-----END PUBLIC KEY-----

```

[CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks](#)

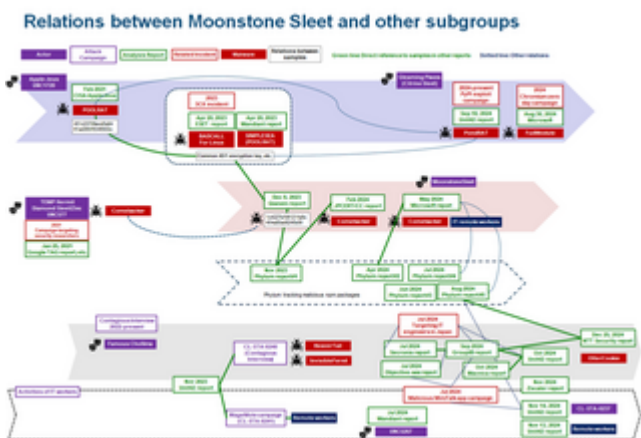


[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities](#)

```
__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;
}
```

[DslodgRAT Malware Installed in Ivanti Connect Secure](#)



[Tempted to Classifying APT Actors: Practical Challenges of Attribution in the Case of Lazarus's Subgroup](#)

Source: https://blogs.jpccert.or.jp/en/2022/03/anti_upx_unpack.html