

Knowledge Fragment: Bruteforcing Andromeda Configuration Buffers

byte-atlas.blogspot.ch/2015/04/kf-andromeda-bruteforcing.html

```

00000310 01 00 00 80 00 00 00 00 00 00 00 00 00 00 00 00 ...e.....
00000320 B6 C4 67 4E 8B FF 55 8B EC E9 70 25 B2 F1 90 90 90 ÅÄgN<yÜ<iép*ñ..
00000330 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000340 90 90 90 90 B8 7D 00 00 00 E9 C7 D2 97 FC 90 90 .....}...éÇÖ-ü..
00000350 90 90 90 90 14 07 03 00 D4 E2 04 63 53 03 86 E4 ...[]...ôâ.cS.tâ
00000360 82 5D 97 1C C6 F8 58 9C F0 8F 2C DA 79 0B 6D 1C ,]-.EsXos,Úy.m.
00000370 CE CB 9D BA 81 C5 C9 42 60 F1 63 48 87 45 00 C1 ÎÉ.°.ÃËB`ñcH+E.Á
00000380 FE 34 8B BF BB 84 93 0D B7 CA 47 DC 2F 8A 35 8A p4<¿»»^\..EGÜ/ŠSŠ
00000390 2D 48 87 31 33 B5 B1 3D 4F A8 2F 49 17 4D E4 58 -H+13µ±=O"/I.MáX
000003A0 93 11 A4 81 3B 4E 1E 8A 28 79 F7 8F 16 5A 85 2F `..π.;N.Š(y+.Z../
000003B0 0A 11 3E 4A DF 5B 70 06 57 9D 33 F0 80 AE AD 6A ..>Jš[p.W.3š@-j
000003C0 13 D2 ED 95 50 CE E7 24 DD 4C D8 DB 84 4D 56 13 .ôi+Pİç$.LØŨ„MV.
000003D0 40 83 06 2D 3C 13 F5 52 59 F3 34 1F 84 AC 5C 46 Øf.-<.šRYó4.„~\F
000003E0 13 EC E8 12 C8 50 8D 87 8B 59 A8 D6 17 0D 4C D8 .iè.ÈP.+<Y"Ö..LØ
000003F0 DB 84 4D 56 4E 52 C6 5C 3A 3B 54 F3 51 58 F1 39 Ů„MVNRÈ\;;TóQXš9
00000400 58 90 A1 02 1F DD 4C D8 DB 84 4D 56 13 40 83 06 X.j...LØŨ„MV.Øf.
00000410 2D 3C 19 FB 4B 55 BA 2F 13 94 E6 1B 4B 18 E4 BF -<.úKU°/.°æ.K.äç
00000420 55 D6 5C 98 1D DD 4C D8 DB 84 4D 56 0D 54 9E 15 UØ\`..LØŨ„MV.Tž.
00000430 24 21 19 FF 11 53 E6 26 59 89 A7 16 40 04 AF B7 $!.ÿ.Sæy%Š.Ø.~.
00000440 13 D6 00 F0 1B CB C7 A3 C5 68 48 CA B7 6A 91 BB .Ö.š.ÈÇĚĀhHÈ.`j»
00000450 83 E9 07 EE D2 78 8B 88 85 78 28 6B 3F 39 72 36 fé.iØx<`_..x(k?9r6
00000460 6F 88 FF DB 63 6D B4 F5 F3 89 99 C5 68 8D 68 6B ó`ÿÜcm`šóš™Āh.hk
00000470 7B 62 9D 05 00 00 00 00 00 00 00 00 00 00 00 00 {b.....
00000480 77 75 61 75 73 65 72 76 00 00 00 00 57 69 6E 44 wuauerv...WinD
00000490 65 66 65 6E 64 00 00 00 4D 70 73 53 76 63 00 00 efend...MpsSvc..
000004A0 53 68 61 72 65 64 41 63 63 65 73 73 00 00 00 SharedAccess...
000004B0 77 73 63 73 76 63 00 00 37 35 34 30 33 37 65 37 wscsvc..754037e7
000004C0 62 65 38 66 36 31 63 62 62 31 62 38 35 61 62 34 be8f61cbb1b85ab4
000004D0 36 63 37 64 61 37 37 64 00 00 00 77 73 32 5F 6c7da77d....ws2
000004E0 33 32 2E 64 6C 6C 00 00 47 65 74 41 64 64 72 49 32.dll1..GetAddrI
000004F0 6E 66 6F 57 00 00 00 6E 74 64 6C 6C 2E 64 6C nfoW....ntdll.dll

```

- BID
- RC4 KEY
- LIST POINTER / URL LENGTH
- LIST CONTENT / CRYPTED URL
- FAKE RC4 KEY

This blog post details how the more recent versions of Andromeda store their C&C URLs and RC4 key and how this information can be bruteforced from a memory dump.

Storage Format

The Andromeda configuration always starts with the value that is transferred as "bid" to the C&C server.

It is 4 bytes long and most likely resembles a builder / botnet ID. In some binaries I had a look at, this was likely a Y-M-D binary date as in the example shown below: 14-07-03.

After an arbitrary number of random bytes concatenated to the "bid", the binary RC4 key of length 16 bytes follows.

This key is both used to decrypt the configuration as well as to encrypt the C&C traffic.

Note that this key is stored in reversed order to decrypt the configuration buffer.

Next, more arbitrary random bytes are added, and then a linked list of encrypted C&C URLs follows.

The first byte of each list entry is the offset to the next list item; a zero byte pointer indicates the end of the list.

Each list entry is simply encrypted with the reversed RC4 key as described previously,

resulting in the crypted C&C entries having identical substrings at the start, the crypted equivalent of "http" => "\x0D\x4C\xD8\xDB".

```

00000300 8C 04 F9 7F 80 04 F9 7F 00 00 00 00 02 00 00 80  &.ù.é.ù.....é
00000310 01 00 00 80 00 00 00 00 00 00 00 00 00 00 00 00  ...é.....
00000320 B6 C4 67 4E 8B FF 55 8B EC E9 70 25 B2 F1 90 90  &ÏgN<ÿU<iép*ñ..
00000330 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  .....
00000340 90 90 90 90 B8 7D 00 00 00 E9 C7 D2 97 FC 90 90  ...>)...éÇÖ-ù..
00000350 90 90 90 90 14 07 03 00 D4 E2 04 63 53 03 86 E4  ....[]...Ôá.cS.+â
00000360 82 5D 97 1C C6 F8 58 9C F0 8F 2C DA 79 0B 6D 1C  ,]-.EeXœS.,Ûy.m.
00000370 CE CB 9D BA 81 C5 C9 42 60 F1 63 48 87 45 00 C1  ÎË.°.ÃËB`ñcH+E.Á
00000380 FE 34 8B BF BB 84 93 0D B7 CA 47 DC 2F 8A 35 8A  b4<ç>>`".ÊGÛ/SSŠ
00000390 2D 48 87 31 33 B5 B1 3D 4F A8 2F 49 17 4D E4 58  -H+13µ±=O"/I.MaX
000003A0 93 11 A4 81 3B 4E 1E 8A 28 79 F7 8F 16 5A 85 2F  `".κ.;N.Š{y÷..Z../
000003B0 0A 11 3E 4A DF 5B 70 06 57 9D 33 F0 80 AE AD 6A  ..>Jß[p.W.3&e@-j
000003C0 13 D2 ED 95 50 CE E7 24 0D 4C D8 DB 84 4D 56 13  .Ói•Pİç$.LØÛ„MV.
000003D0 40 83 06 2D 3C 13 F5 52 59 F3 34 1F 84 AC 5C 46  Øf.-<.ðRYó4„-\\F
000003E0 13 EC E8 12 C8 50 8D 87 8B 59 A8 D6 17 0D 4C D8  .iè.ÈP.+<Y`Ö..LØ
000003F0 DB 84 4D 56 4E 52 C6 5C 3A 3B 54 F3 51 58 F1 39  Û„MVNRÆ\;:TóQXŘ9
00000400 58 90 A1 02 1F 0D 4C D8 DB 84 4D 56 13 40 83 06  X.i...LØÛ„MV.Øf.
00000410 2D 3C 19 FB 4B 55 BA 2F 13 94 E6 1B 4B 18 E4 BF  -<.ûKU°/“æ.K.âç
00000420 55 D6 5C 98 1D 0D 4C D8 DB 84 4D 56 0D 54 9E 15  UÖ\`..LØÛ„MV.Tž.
00000430 24 21 19 FF 11 53 E6 26 59 89 A7 16 40 04 AF B7  $!.ÿ.SæYk$.Ø.~
00000440 13 D6 00 F0 1B CB C7 A3 C5 68 48 CA B7 6A 91 BB  .Ó.8.EÇŁĂhHE•j`»
00000450 83 E9 07 EE D2 78 8B 88 85 78 28 6B 3F 39 72 36  fé.iÓx<`~x{k?9r6
00000460 6F 88 FF DB 63 6D B4 F5 F3 89 99 C5 68 8D 68 6B  o`ÿÜcm'óó™`Ăh.hk
00000470 7B 62 9D 05 00 00 00 00 00 00 00 00 00 00 00 00  {b.....
00000480 77 75 61 75 73 65 72 76 00 00 00 00 57 69 6E 44  wuauerv...WinD
00000490 65 66 65 6E 64 00 00 00 4D 70 73 53 76 63 00 00  efend...MpsSvc..
000004A0 53 68 61 72 65 64 41 63 63 65 73 73 00 00 00 00  SharedAccess...
000004B0 77 73 63 73 76 63 00 00 37 35 34 30 33 37 65 37  wscsvc..754037e7
000004C0 62 65 38 66 36 31 63 62 62 31 62 38 35 61 62 34  be8f61cbb1b85ab4
000004D0 36 63 37 64 61 37 37 64 00 00 00 00 77 73 32 5F  6c7da77d....ws2_
000004E0 33 32 2E 64 6C 6C 00 00 47 65 74 41 64 64 72 49  32.dll..GetAddrI
000004F0 6E 66 6F 57 00 00 00 00 6E 74 64 6C 6C 2E 64 6C  nfoW....ntdll.dll
00000500 6C 00 00 00 4E 74 4F 70 65 6E 53 65 63 74 69 6F  l...NtOpenSectio

```

- BID
- RC4 KEY
- LIST POINTER / URL LENGTH
- LIST CONTENT / CRYPTED URL
- FAKE RC4 KEY

Andromeda config buffer and fake RC4 key

Concealment of the configuration on bot initialization

During its initialization, the Andromeda bot parses this configuration buffer and stores its parts on the heap. Each data blob is prefixed with an indicator (crc32 over part of host processes' header, or 0x706e6800, xor bot_id), allowing the malware to identify its fragments on the heap in a similar way to the technique known as egg hunting.

```

parseConfig      proc near                ; CODE XREF: sub_7FF93428+914j
                                        ; DATA XREF: destroyParseConfig+540 ...
var_24           = dword ptr -24h
var_20           = dword ptr -20h
arg_0            = dword ptr  8

    push    ebp
    mov     ebp, esp
    sub     esp, 24h
    push    21h
    call    allocateMemory
    mov     ds:rc4key_offset, eax
    test   eax, eax
    jz     locret_7FF90F5D
    mov     eax, [ebp+arg_0]
    and    [ebp+var_20], 0
    push    esi
    xor     eax, 443BB35Dh
    push    edi
    xor     edi, edi
    mov     [ebp+var_24], eax
    xor     esi, esi

loc_7FF90EBF:                                     ; CODE XREF: parseConfig+594j
    push    dword ptr ds:rc4key[esi]
    call    j_tohl
    push    eax
    lea    eax, [ebp+var_24]
    push    eax
    mov     eax, ds:rc4key_offset
    add    eax, edi
    push    eax
    call    ds:sprintfA
    add    esi, 4
    add    esp, 0Ch
    add    edi, 8
    cmp    esi, 10h
    jnb    short loc_7FF90EBF
    mov     eax, ds:rc4key_offset
    xor    ecx, ecx
    add    eax, 1Fh

loc_7FF90EF5:                                     ; CODE XREF: parseConfig+704j
    mov     dl, [eax]
    mov     byte ptr [ebp+ecx+var_24], dl
    inc    ecx
    dec    eax
    cmp    ecx, 20h
    jnb    short loc_7FF90EF5
    mov     al, ds:cnc_urls
    and    [ebp+arg_0], 0
    mov     esi, offset cnc_urls
    test   al, al
    jz     short loc_7FF90F5B
    push    ebx

loc_7FF90F15:                                     ; CODE XREF: parseConfig+C84j
    movzx   edi, al
    lea    eax, [edi+5]
    push    eax
    call    allocateMemory
    mov     ecx, [ebp+arg_0]
    or     ecx, 706E6800h
    xor    ecx, ds:dwBotId
    lea    ebx, [eax+4]
    push    ebx
    mov     [eax], ecx
    push    edi
    lea    eax, [esi+1]
    push    eax
    call    memcpy
    push    edi
    push    ebx
    push    20h
    lea    eax, [ebp+var_24]
    push    eax
    call    rc4crypt
    inc    [ebp+arg_0]
    lea    esi, [esi+edi+1]
    mov     al, [esi]
    test   al, al
    jnz    short loc_7FF90F15
    pop    ebx

loc_7FF90F5B:                                     ; CODE XREF: parseConfig+824j
    pop    edi
    pop    esi

locret_7FF90F5D:                                     ; CODE XREF: parseConfig+144j
    leave  4
parseConfig      endp

```

function used to handle the config and store rc4_key + C&C URLs on the heap

Afterwards, as a means of anti-analysis, the parsing routine is overwritten with a static 4 bytes (to kill the function prologue) and another function of the bot (in this case the function responsible for settings up hooking) in order to destroy the pointers to the RC4 key and C&C list.

```

destroyParseConfig proc near          ; CODE XREF: sub_7FF93428+961p
                                     ; DATA XREF: destroyParseConfig+241w
    push    ebp
    mov     ebp, esp
    push    esi
    push    edi
    lea    edi, parseConfig
    lea    esi, installHooks
    mov     ecx, offset loc_7FF90F79
    sub     ecx, edi

loc_7FF90F79:                          ; DATA XREF: destroyParseConfig+111w
    rep movsb dword ptr ds:parseConfig, 0CC0004C2h
    mov     dword ptr ds:destroyParseConfig, 0CC0004C2h
    pop     edi
    pop     esi
    pop     ebp
    retn
destroyParseConfig endp

```

```

installHooks proc near                ; CODE XREF: checkForRights+2111p
                                     ; checkForRights+2301p ...
arg_0      = dword ptr 8
arg_4      = dword ptr 0Ch
arg_8      = dword ptr 10h

    push    ebp
    mov     ebp, esp
    push    esi
    mov     esi, [ebp+arg_0]
    mov     al, [esi]
    push    edi
    xor     edi, edi
    cmp     al, 0E9h
    jnz     short loc_7FF91B79
    lea    eax, [eax+esi+5]

loc_7FF91B6B:                          ; CODE XREF: installHooks+3A1j
    push    [ebp+arg_8]
    push    [ebp+arg_4]
    push    eax
    call    installHooks
    jmp     short loc_7FF91BEF

loc_7FF91B79:                          ; CODE XREF: installHooks+E1j
    cmp     al, 0EBh
    jnz     short loc_7FF91B90
    movzx  eax, byte ptr [esi+1]
    test   al, al
    jns     short loc_7FF91B8A
    or     eax, 0FFFFFF00h

loc_7FF91B8A:                          ; CODE XREF: installHooks+2F1j
    lea    eax, [eax+esi+2]
    jmp     short loc_7FF91B6B

loc_7FF91B90:                          ; CODE XREF: installHooks+271j
    mov     eax, esi

loc_7FF91B92:                          ; CODE XREF: installHooks+4C1j

```

```

destroyedParseConfig dd 0CC0004C2h    ; CODE XREF: seg000:7FF90EA9E1p
                                     ; sub_7FF93428+911p
                                     ; DATA XREF: ...

    mov     esi, [ebp+8]
    mov     al, [esi]
    push    edi
    xor     edi, edi
    cmp     al, 0E9h
    jnz     short loc_7FF90EB5
    mov     eax, [esi+1]
    lea    eax, [eax+esi+5]

loc_7FF90EA7:                          ; CODE XREF: seg000:7FF90ECA1j
    push    dword ptr [ebp+10h]
    push    dword ptr [ebp+0Ch]
    push    eax
    call    near ptr destroyedParseConfig
    jmp     short loc_7FF90F2B

loc_7FF90EB5:                          ; CODE XREF: seg000:7FF90E9E1j
    cmp     al, 0EBh
    jnz     short loc_7FF90ECC
    movzx  eax, byte ptr [esi+1]
    test   al, al
    jns     short loc_7FF90EC6
    or     eax, 0FFFFFF00h

loc_7FF90EC6:                          ; CODE XREF: seg000:7FF90EBF1j
    lea    eax, [eax+esi+2]
    jmp     short loc_7FF90EA7

loc_7FF90ECC:                          ; CODE XREF: seg000:7FF90EB71j
    mov     eax, esi

loc_7FF90ECE:                          ; CODE XREF: seg000:7FF90EDC1j

```

top: function to destroy the parseConfig by overwriting with installHooks(), left: installHooks() right: resulting parseConfig

Extraction of RC4 key and C&C URLs

Although the exact offsets of RC4 key and C&C URL list are not available when examining a finally initialized Andromeda memory image in the injected process, it is possible to recover this information through guessing.

Finding the "bid"

Characteristic for all encountered versions of Andromeda is a format string similar to the following:

```
id:%lu|bid:%lu|os:%lu|la:%lu|rg:%lu
```

or more recently:

```
{"id":%lu,"bid":%lu,"os":%lu,"la":%lu,"rg":%lu}
```

As its fields are likely filled in with a `*sprintf*` function, we can identify the offset of the "bid" by statically examining parameters passed to said string format API call (this can e.g. be achieved with a carefully crafted regex).

```

E8 02 0B 00 00    call    sub_7FF91BF5
FF 35 80 3D F9 7F  push   ds:dword_7FF93D80
A3 88 3D F9 7F    mov     ds:dword_7FF93D88, eax
50               push   eax
FF 35 90 3D F9 7F  push   ds:os_id
FF 35 54 03 F9 7F  push   ds:botnet_id
FF 35 78 3D F9 7F  push   ds:bot_id
68 8C 05 F9 7F    push   offset aIdLuBidLu0sLuL ; "id:%lu|bid:%lu|os:%lu|la:%lu|rg:%lu"
57               push   edi
FF 15 E4 01 F9 7F  call   ds:jmp_sprintf
83 C4 1C          add     esp, 1Ch
8B D8             mov     ebx, eax
E8 EC FE FF FF    call   sub_7FF91013
8B F0             mov     esi, eax
85 F6             test    esi, esi
74 2D             jz     short loc_7FF9115A

```

7FF90354	botnet_id	dd	30714h
7FF90358		db	004h ; +
7FF90359		db	0E2h ; G
7FF9035A		db	4
7FF9035B		db	63h ; c
7FF9035C		db	53h ; s
7FF9035D		db	3

reference to the botnet/builder id "bid" with a characteristic sequence of instructions

Treating the "bid" as start for the potential configuration buffer, we can assume its end by searching for a zero dword value starting at the offset of the "bid".
 For the tested memory dumps, the resulting potential configuration buffer had a length of around 300 bytes.

Identifying crypted C&C URL candidates

As described above, the C&C URLs are stored as a linked list.
 Randomly assuming that a server address will be somewhere between 0x8 and 0x30 characters long, we can extract all byte sequences from the potential configuration buffer that match this property (start bytes highlighted):

```

0000 14 07 03 00 d4 e2 04 63 53 03 86 e4 82 5d 97 1c  .....cS....]..
0010 c6 f8 58 9c f0 8f 2c da 79 0b 6d 1c ce cb 9d ba  ..X...,y.m....
0020 81 c5 c9 42 60 f1 63 48 87 45 00 c1 fe 34 8b bf  ...B`.cH.E...4..
0030 bb 84 93 0d b7 ca 47 dc 2f 8a 35 8a 2d 48 87 31  .....G./5.-H.1
0040 33 b5 b1 3d 4f a8 2f 49 17 4d e4 58 93 11 a4 81  3..=O./I.M.X...
0050 3b 4e 1e 8a 28 79 f7 8f 16 5a 85 2f 0a 11 3e 4a  ;N..(y...Z./..>J
0060 df 5b 70 06 57 9d 33 f0 80 ae ad 6a 13 d2 ed 95  .[p.W.3....]....
0070 50 ce e7 24 0d 4c d8 db 84 4d 56 13 40 83 06 2d  P.$..L...MV.@...-
0080 3c 13 f5 52 59 f3 34 1f 84 ac 5c 46 13 ec e8 12  <..RY.4....F...
0090 c8 50 8d 87 8b 59 a8 d6 17 0d 4c d8 db 84 4d 56  .P...Y....L...MV
00a0 4e 52 c6 5c 3a 3b 54 f3 51 58 f1 39 58 90 a1 02  NR...;T.QX.9X...
00b0 1f 0d 4c d8 db 84 4d 56 13 40 83 06 2d 3c 19 fb  ..L...MV.@...-<..
00c0 4b 55 ba 2f 13 94 e6 1b 4b 18 e4 bf 55 d6 5c 98  KU./....K...U...
00d0 1d 0d 4c d8 db 84 4d 56 0d 54 9e 15 24 21 19 ff  ..L...MV.T.$!...
00e0 11 53 e6 26 59 89 a7 16 40 04 af b7 13 d6 00 f0  .S.&Y...@.....

```

00f0 1b cb c7 a3 c5 68 48 ca b7 6a 91 bb 83 e9 07 eehH..j.....
0100 d2 78 8b 88 85 78 28 6b 3f 39 72 36 6f 88 ff db .x...x(k?9r6o...
0110 63 6d b4 f5 f3 89 99 c5 68 8d 68 6b 7b 62 9d 05 cm.....h.hk{b..

resulting in the following candidate sequences (offset, length, start bytes):

offset: 0x000, 14->070300...
offset: 0x00f, 1c->c6f858...
offset: 0x016, 2c->da790b...
offset: 0x019, 0b->6d1cce...
offset: 0x01b, 1c->cecb9d...
offset: 0x033, 0d->b7ca47...
offset: 0x038, 2f->8a358a...
offset: 0x03c, 2d->488731...
offset: 0x046, 2f->49174d...
offset: 0x048, 17->4de458...
offset: 0x04d, 11->a4813b...
offset: 0x052, 1e->8a2879...
offset: 0x054, 28->79f78f...
offset: 0x058, 16->5a852f...
offset: 0x05b, 2f->0a113e...
offset: 0x05c, 0a->113e4a...
offset: 0x05d, 11->3e4adf...
offset: 0x06c, 13->d2ed95...
offset: 0x073, 24->0d4cd8...
offset: 0x074, 0d->4cd8db...
offset: 0x07b, 13->408306...
offset: 0x07f, 2d->3c13f5...
offset: 0x081, 13->f55259...
offset: 0x087, 1f->84ac5c...
offset: 0x08c, 13->ece812...
offset: 0x08f, 12->c8508d...
offset: 0x098, 17->0d4cd8...
offset: 0x099, 0d->4cd8db...
offset: 0x0b0, 1f->0d4cd8...
offset: 0x0b1, 0d->4cd8db...
offset: 0x0b8, 13->408306...
offset: 0x0bc, 2d->3c19fb...
offset: 0x0be, 19->fb4b55...
offset: 0x0c3, 2f->1394e6...
offset: 0x0c4, 13->94e61b...
offset: 0x0c7, 1b->4b18e4...
offset: 0x0c9, 18->e4bf55...

offset: 0x0d0, 1d->0d4cd8...
offset: 0x0d1, 0d->4cd8db...
offset: 0x0d8, 0d->549e15...
offset: 0x0db, 15->242119...
offset: 0x0dc, 24->2119ff...
offset: 0x0dd, 21->19ff11...
offset: 0x0de, 19->ff1153...
offset: 0x0e0, 11->53e626...
offset: 0x0e3, 26->5989a7...
offset: 0x0e7, 16->4004af...
offset: 0x0ec, 13->d600f0...
offset: 0x0f0, 1b->cbc7a3...
offset: 0x106, 28->6b3f39...

Identifying the RC4 key

Next, we can try to decrypt these URL candidates by using all possible RC4 keys from the potential configuration buffer.

For this, we take every consecutive 16 bytes, hex encode them, reverse their order, and perform RC4 against all C&C URL candidates.

Example: candidate sequence at offset 0xd1, length: 0x1d bytes:

```
00d0 1d 0d 4c d8 db 84 4d 56 0d 54 9e 15 24 21 19 ff ..L...MV.T..$!..  
00e0 11 53 e6 26 59 89 a7 16 40 04 af b7 13 d6 00 f0 .S.&Y...@.....
```

bruteforce decryption attempts:

```
rc4(candidate, "c179d5284e68303536402e4d00307041") -> 60a1619e84209c  
rc4(candidate, "6cc179d5284e68303536402e4d003070") -> d378675057f8f2  
rc4(candidate, "8f6cc179d5284e68303536402e4d0030") -> 84ff7a9c4e2168  
rc4(candidate, "858f6cc179d5284e68303536402e4d00") -> 3b5dd0750955f6  
[... 44 more attempts ...]  
rc4(candidate, "33137884d2a853a8f2cd74ac7bd03948") -> 7cea19689c5d40  
rc4(candidate, "5b33137884d2a853a8f2cd74ac7bd039") -> 38ca7a0068f32e  
rc4(candidate, "1b5b33137884d2a853a8f2cd74ac7bd0") -> 6429d8151a51c2  
rc4(candidate, "d31b5b33137884d2a853a8f2cd74ac7b") -> 687474703a2f2f
```

finally we hit a result of 687474703a2f2f which translates to "http://" and the whole URL decrypts to "hxxp://sunglobe.org/index.php" (defanged).

As soon as we decrypt the first sequence starting with "http" we have likely identified the correct RC4 key and can proceed to decrypt all other candidates to complete the list of C&C URLs.

RC4 key used for config: d31b5b33137884d2a853a8f2cd74ac7b

Actual traffic RC4 key: b7ca47dc2f8a358a2d48873133b5b13d

All resolving candidates:

0d4cd8db844d560d549e15242119ff1153e6265989a7164004afb713d6

-> hxxp://sunglobe.org/index.php

0d4cd8db844d56134083062d3c19fb4b55ba2f1394e61b4b18e4bf55d65c98

-> hxxp://masterbati.net/index.php

0d4cd8db844d564e52c65c3a3b54f35158f1395890a102

-> hxxp://0s6.ru/index.php

0d4cd8db844d56134083062d3c13f55259f3341f84ac5c4613ece812c8508d878b59a8d6

-> hxxp://masterhomeguide.com/index.php

Conclusion

It's obvious that the above described method can be optimized here and there. But since it executes in less than a second on a given memdump and gave me good results on a collection of Andromeda dumps, I didn't bother to improve it further.

sample used:

md5: a17247808c176c81c3ea66860374d705

sha256: ce59dbe27957e69d6ac579080d62966b69be72743143e15dbb587400efe6ce77

[Repository with defanged memdump + extraction code](#)