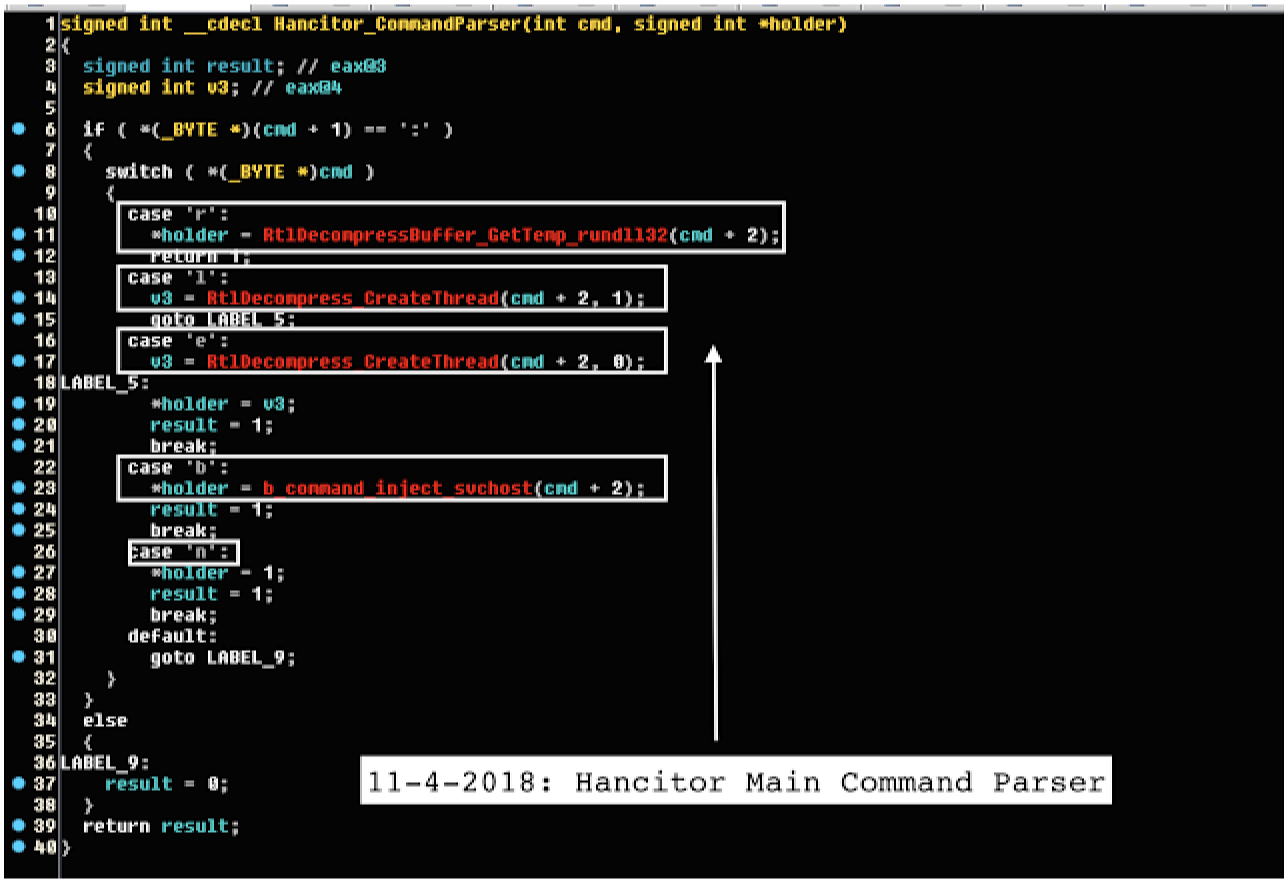
Let's Learn: In-Depth Reversing of Hancitor Dropper/Loader: 2016 vs 2018 Malware Progression

vkremez.com/2018/11/lets-learn-in-depth-reversing-of.html

Goal: Analyze the latest Hancitor variant (build "25xce10") to determine dropper and downloader malware progression in time from 2016 to the latest version in 2018.
Source:
Original Packed Hancitor Loader 32-Bit (x86) (MD5: 0cabdc2d4b83cd8b210fd2bd15d54bdc)
Unpacked Hancitor Dropper & Loader 32-Bit (x86) (MD5: fc7748f302a1566c27568e094873817a)



```
signed int __cdecl Hancitor_CommandParser(int cmd, signed int *holder)
{
  signed int result; // eax@3
  signed int v3; // eax@4

  if ( *(_BYTE *)(cmd + 1) == ':' )
  {
    switch ( *(_BYTE *)cmd )
    {
      case 'r':
        *holder = RtlDecompressBuffer_GetTemp_rundll32(cmd + 2);
        return 1;
      case 'l':
        v3 = RtlDecompress_CreateThread(cmd + 2, 1);
        goto LABEL_5;
      case 'e':
        v3 = RtlDecompress_CreateThread(cmd + 2, 0);
LABEL_5:
        *holder = v3;
        result = 1;
        break;
      case 'b':
        *holder = b_command_inject_svchost(cmd + 2);
        result = 1;
        break;
      case 'n':
        *holder = 1;
        result = 1;
        break;
      default:
        goto LABEL_9;
    }
  }
  else
  {
LABEL_9:
    result = 0;
  }
  return result;
}
```

11-4-2018: Hancitor Main Command Parser

Outline

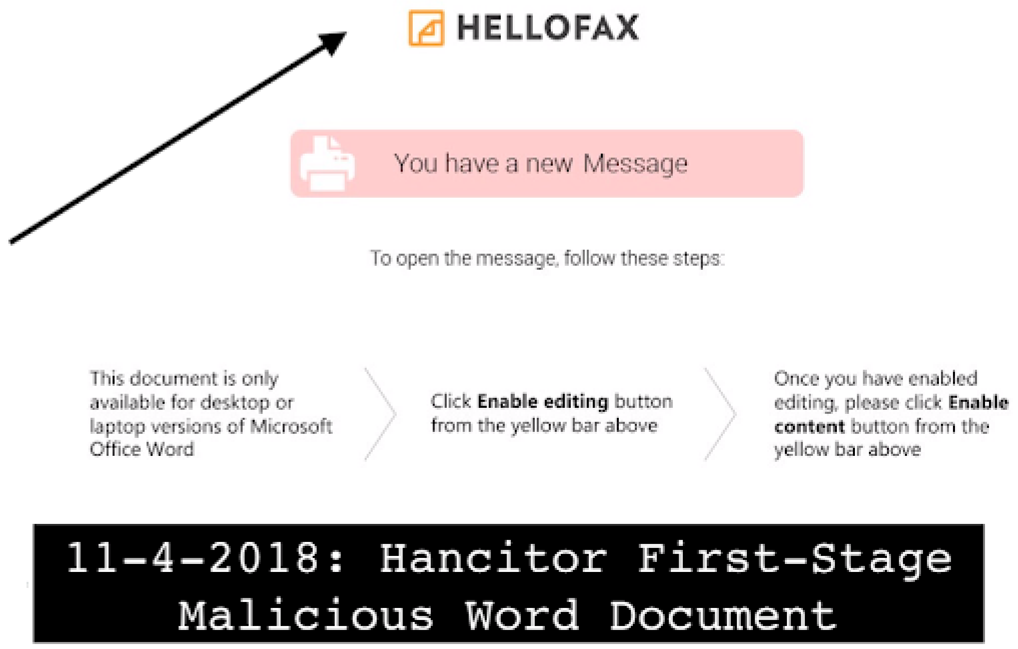
I. Background

The group behind Hancitor distribution campaigns remains to be one of the more resourceful and sophisticated cybercrime loader-as-a-service group delivering various payloads - ranging from simple credential stealer malware to point-of-sale and banking malware variants (from Pony Stealer, EvilPony Stealer, AZORult Stealer to Neverquest Banker, Panda Banker, Gozi ISFB Banker, and Danabot Banker).

One of the most interesting malware analysis revolves around source code-level analysis malware development progression in time. I highly recommend reading this article on Hancitor titled "A Closer Look At Hancitor" written by Nick Hoffman and Jeremy Humble. I will utilize the malware sample from this article to compare against one of the latest Hancitor variants (h/t to @malware_traffic for the latest sample). Additionally, I recommend reading @benkow_'s blog titled "Hancitor Panel Overview" to learn more about the Hancitor panel.

II. Original Packed Hancitor Loader 32-Bit (x86)

The Hancitor malspam email chain included the first-stage loader as a malicious Microsoft Word document requiring a victim to enable macros to view the fax message as if it is from a legitimate company HelloFax.


HELLOFAX

You have a new Message

To open the message, follow these steps:

This document is only available for desktop or laptop versions of Microsoft Office Word

Click **Enable editing** button from the yellow bar above

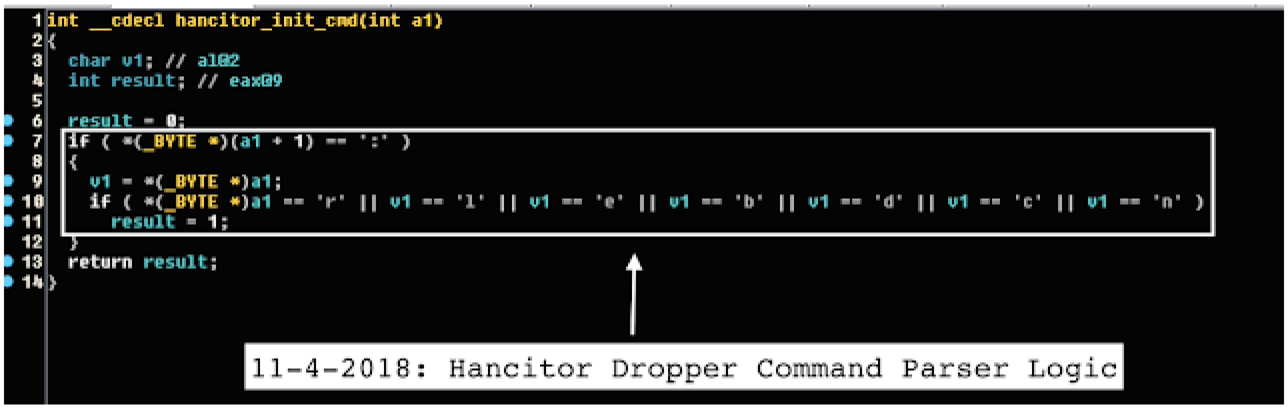
Once you have enabled editing, please click **Enable content** button from the from the yellow bar above

11-4-2018: Hancitor First-Stage Malicious Word Document

III. Unpacked Hancitor Dropper & Loader 32-Bit (x86)

By and large, while the group behind the malware is rather experienced and persistent, the Hancitor dropper remains to be a simple and unsophisticated dropper and loader type of malware that comes with little development from 2016. Some of the notable lack of development and adjustment includes its reliance on ANSI API calls as well as unsophisticated WriteProcessMemeory injection method, modified %TEMP% run method with joined function on CreateProcessA rather than as a separate function, absence of Winhost32.exe check logic and its derivative functions, additional error check on CreateProcessA and exception handling, improved parser function before

WriteProcessMemory injection, joined injection function, no deletion logic, different initial and entry function.
The reviewed 2016 Hancitor dropper version contains 66 functions with the size of 20.00 KB (20480 bytes), while the latest 2018 Hancitor version consists of 51 functions with the size of 20.50 KB (20992 bytes). During pseudo source-code level analysis, it is revealed that the code contains 8 partial function matches (including perfect match, same MD index and constants, strongly connected components, and similar small pseudo-code), 34 unreliable function matches (including perfect match, same MD index and constants, strongly connected components, similar small pseudo-code, strongly connected components small-primes-product, and loop count).
The notable differences include the absence of the connectivity check in the latest version. For example, the 2016 version contained tried to see if it can reach google.com as part of its logic. Moreover, the 2016 version did not contain the RC4 encryption with RtlDecompressBuffer API call to decode the next stage as opposed to the 2018 variant. The "e" command appears to be a new one relative to the 2016 version.
A. Hancitor MainThreadProcessor
In this case, I am looking into the Hancitor build "25xce10". The Hancitor malware receives multiple commands that it leverages for parsing and executing additional steps. The commands and their execution are separated by ":", and the URLs, for example, are separated with "|" symbol.


```
int __cdecl hancitor_init_cmd(int a1)
{
  char v1; // al@2
  int result; // eax@9

  result = 0;
  if ( *(_BYTE *)(a1 + 1) == ':' )
  {
    v1 = *(_BYTE *)a1;
    if ( *(_BYTE *)a1 == 'r' || v1 == 'l' || v1 == 'e' || v1 == 'b' || v1 == 'd' || v1 == 'c' || v1 == 'n' )
      result = 1;
  }
  return result;
}
```

Hancitor has logic to parse the following commands:

Command	Description
"r"	Download and execute .DLL or .EXE
"l"	Download and execute .EXE in separate thread (arg=1)
"e"	Download and execute .EXE in separate thread (arg=0)
"b"	Download and inject code into svchost.exe
"d"	N/A (not implemented; used to delete itself in older version)
"c"	N/A (not implemented)

"n"	N/A (not implemented)

The full pseudo-coded Hancitor main command processor function is as follows:

```
////////////////////////////////////////////////////////////////////////////////
//////////// Hancitor MainThreadProcessor //////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
signed int __cdecl HancitorMainCommandProcessor(int cmd, signed int *a2)
{

  if ( *(_BYTE *)(a1 + 1) == ':' )
  {
    switch ( *(_BYTE *)cmd )
    {
      case 'r':
// "r" command -> download and execute .DLL or .EXE
// if .DLL -> via rundll32.exe, $s,f1 in %TEMP% or if .EXE -> CreateProcessA

*a2 = RtlDecompressBuffer_GetTemp_rundll32(cmd + 2);
        return 1;
      case 'l':
// "l" command -> download and execute .EXE in separate thread (arg=1)
// check for "MZ" header, call VirtualAlloc and/or dynamic API resolution

        v3 = RtlDecompress_CreateThread(cmd + 2, 1);
        goto LABEL_5;
      case 'e':
// "e" command -> download and execute .EXE in separate thread (arg=0)
// check for "MZ" header, call VirtualAlloc and/or dynamic API resolution

        v3 = RtlDecompress_CreateThread(cmd + 2, 0);

LABEL_5:
        *a2 = v3;
        result = 1;
        break;
      case 'b':
// "b" command -> download and inject code into svchost.exe
// check for "MZ" header, call allocate memory and write code into the memory

*a2 = RtlDecompressBuffer_Call_svchost_injection(cmd + 2);
        result = 1;
        break;
      case 'n':
// "n" command -> no processor for this command

*a2 = 1;
        result = 1;
        break;
      default:
        goto LABEL_9;
    }
  }
  else
  {
LABEL_9:
    result = 0;
  }
  return result;
}
```

B. Hancitor SystemInfo Generation
By and large, the Hancitor malware collects generic information regarding the host that includes various information filled into the formatted collector string:

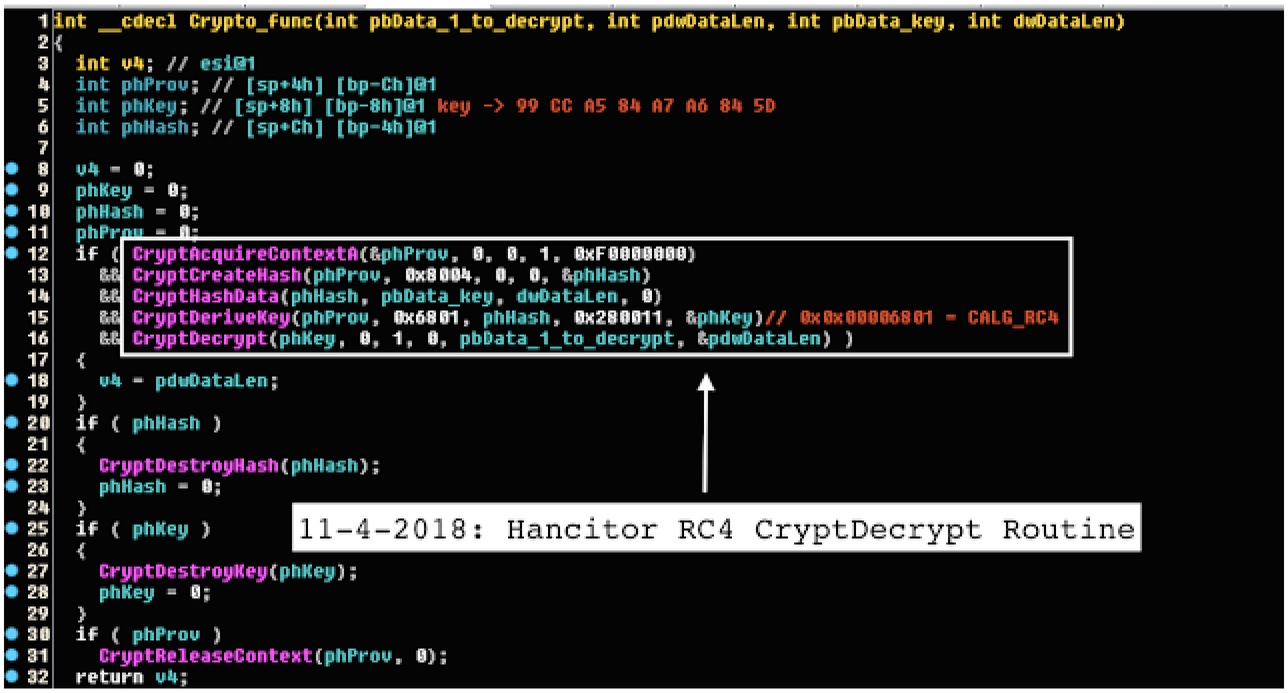
GUID=%I64u&BUILD=%s&INFO=%s&IP=%s&TYPE=1&WIN=%d.%d(x32|64)

The shortened pseudo-coded C++ Hancitor SystemInfo function is as follows:

```
////////////////////////////////////////////////////////////////////////////////
//////////// Hancitor MainThreadProcessor ///////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
 signed int __cdecl SystemInfo(int a1, int a2, int a3)
{
  version_ret = GetVersion();
  bot_id = qword_1E6178;
  win_version = version_ret;
  build_bot_1 = HIDWORD(qword_1E6178);
  if ( !qword_1E6178 )
  {
    LODWORD(v7) = Adapteers_GetWindowsDirectoryA();
    build_bot_1 = HIDWORD(v7);
    bot_id = v7;
    qword_1E6178 = v7;
  }
  GetComputerNameA_0((signed int)&comp_name);
  ExternalAPI_resolution(v8, bot_id, (int)&external_ip);
  win_version_1 = (unsigned __int8)win_version;
  check_if_x64 = GetNativeSystemInfo() == 1;
  comp_name_info = pbData_1;
  if ( check_if_x64 )
  {
    if ( !pbData_1 )
    {
      pbData_1 = GetProcessHeap_0(0x2000);
      func1(pbData_1, (char *)&unk_1E4018, 0x2000);
      Crypto_func(pbData_1, 0x2000, (int)&pbData_key, 8);
      comp_name_info = pbData_1;
    }
    wsprintfA(
      &formatted_systeminfo,
      "GUID=%I64u&BUILD=%s&INFO=%s&IP=%s&TYPE=1&WIN=%d.%d(x64)",
      bot_id,
      build_bot_1,
      comp_name_info,
      &comp_name,
      &external_ip,
      win_version_1,
      HIBYTE(win_version));
  }
```

C. Hancitor RC4 CryptDecrypt Routine

The Hancitor dropper heavily utilizes RtlDecompressBuffer and Crypto API to decrypt its payloads. Rather than relying on custom decryption, the malware simply implements the RC4 decryption logic.



```
int __cdecl Crypto_func(int pbData_1_to_decrypt, int pdwDataLen, int pbData_key, int dwDataLen)
{
  int v4; // esi@1
  int phProv; // [sp+4h] [bp-Ch]@1
  int phKey; // [sp+8h] [bp-8h]@1 key -> 99 CC A5 84 A7 A6 84 5D
  int phHash; // [sp+Ch] [bp-4h]@1

  v4 = 0;
  phKey = 0;
  phHash = 0;
  phProv = 0;
  if ( CryptAcquireContextA(&phProv, 0, 0, 1, 0xF0000000)
    && CryptCreateHash(phProv, 0x8004, 0, 0, &phHash)
    && CryptHashData(phHash, pbData_key, dwDataLen, 0)
    && CryptDeriveKey(phProv, 0x6801, phHash, 0x280011, &phKey)// 0x0x00006801 = CALG_RC4
    && CryptDecrypt(phKey, 0, 1, 0, pbData_1_to_decrypt, &pdwDataLen) )
  {
    v4 = pdwDataLen;
  }
  if ( phHash )
  {
    CryptDestroyHash(phHash);
    phHash = 0;
  }
  if ( phKey )
  {
    CryptDestroyKey(phKey);
    phKey = 0;
  }
  if ( phProv )
    CryptReleaseContext(phProv, 0);
  return v4;
```

IV. Yara Signature

```
import "pe"

rule crime_win32_hancitor_dropper {
   meta:
      author = "@VK_Intel"
      reference = "Detects Hancitor Dropper/Loader"
      date = "2018-11-04"
      hash1 = "c61f929981c573e43dd08f0c5a047ba3a3167436b493df819461d4e7953a91ed"
   strings:
      $s1 = "Rundll32.exe %s,f1" fullword ascii
      $s2 = "explorer.exe" fullword ascii
      $s3 = "Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like
Gecko" fullword ascii
      $s4 = "GUID=%I64u&BUILD=%s&INFO=%s&IP=%s&TYPE=1&WIN=%d.%d(x64)" fullword ascii
      $s5 = "GUID=%I64u&BUILD=%s&INFO=%s&IP=%s&TYPE=1&WIN=%d.%d(x32)" fullword ascii
      $s6 = "http://api.ipify.org" fullword ascii
      $s7 = "Content-Type: application/x-www-form-urlencoded" fullword ascii

      $crypt_sysfunction = { 8d ?? ?? ?? ?? ?? 50 e8 ?? ?? ?? ?? 8d ?? ?? 50 e8 ?? ??
?? ?? 0f b6 c3 83 c4 08 c1 eb 08 89 ?? ?? 0f b6 db e8 ?? ?? ?? ?? 83 f8 01 a1 ?? ??
?? ?? 75 ?? 85 c0 75 ?? 68 00 20 00 00 e8 ?? ?? ?? ?? 68 00 20 00 00 68 18 40 1e 00
50 a3 ?? ?? ?? ?? e8 ?? ?? ?? ?? 6a 08 68 10 40 1e 00 68 00 20 00 00 ff ?? ?? ?? ??
?? e8 ?? ?? ?? ?? a1 ?? ?? ?? ?? 83 c4 20}

      $external_ip_resolution = { 55 8b ec 51 80 ?? ?? ?? ?? ?? ?? 75 ?? 8d ?? ?? 50
6a 20 68 80 61 1e 00 68 c4 31 1e 00 e8 ?? ?? ?? ?? 83 c4 10 83 f8 01 75 ?? 8b ?? ??
c6 ?? ?? ?? ?? ?? ?? 68 80 61 1e 00 ff ?? ?? ff ?? ?? ?? ?? ?? b8 01 00 00 00 8b e5
5d c3 68 dc 31 1e 00 ff ?? ?? c6 ?? ?? ?? ?? ?? ?? ff ?? ?? ?? ?? ?? 33 c0 8b e5 5d
c3}

      $main_cmd_processor = { 55 8b ec 8b ?? ?? 80 ?? ?? ?? 75 ?? 0f ?? ?? 83 c0 9e
83 f8 10 77 ?? 0f ?? ?? ?? ?? ?? ?? ff ?? ?? ?? ?? ?? ?? 8d ?? ?? 50 e8 ?? ?? ?? ??
8b ?? ?? 83 c4 04 89 ?? b8 01 00 00 00 5d c3 6a 01 8d ?? ?? 50 e8 ?? ?? ?? ?? 8b ??
?? 83 c4 08 89 ?? b8 01 00 00 00 5d c3 6a 00 eb ?? 8d ?? ?? 50 e8 ?? ?? ?? ?? 8b ??
?? 83 c4 04 89 ?? b8 01 00 00 00 5d c3 8b ?? ?? c7 ?? ?? ?? ?? ?? b8 01 00 00 00 5d
c3 33 c0 5d c3}

   condition:
      ( uint16(0) == 0x5a4d and
         filesize < 60KB and
         pe.imphash() == "9ea0470ccffd0a7ac8dd70e0968d3e95" and
         ( all of them )
      or  5 of ($s*)  and ( $main_cmd_processor  ) or ( $crypt_sysfunction and
$external_ip_resolution ) )
      }
```