SWIMMING INTO TROJAN AND ROOTKIT GAMETHIEF.WIN32.MAGANIA HOSTILE CODE

Author: Giuseppe 'Evilcry' Bonfa

Editor: Shub-Nigurrath

ARTeam

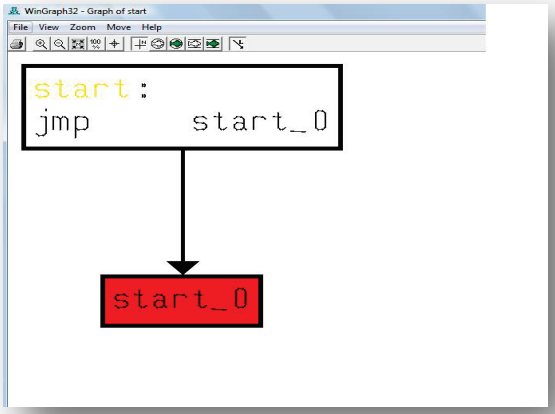
FOREWORDS

Trojan-GameThief.Win32.Magania, according to Kaspersky naming convention, monitors the user activities trying to obtain valuable information from the affected user, especially about gaming login accounts. From a functional point of view you can check this post also:

http://evilcodecave.blogspot.com/2009/08/how-fastly-bypass-hostile-code-for.html

where you can find also files and registry entries to Remove Megania.

In this paper we will analyse more deeply the structure of this malware, especially the polymorphic part that represents a typical sample of hostile code. Starting from the first load into IDA we can see that Megania's **PE structure** and **Import Table destroyed**, this is how looks from WinGraph:



As you can see IDA shows that's we're facing a crippled structure! This is a start for a nice reversing journey directly to the centre of the malware! These journeys and the nested-dolls structure are really typical of modern malware analysis, and are also a great portion of the fun!

[GENERAL MALWARE INFORMATION:

FILE NAME: G9.EXE
FILE SIZE: 105555 BYTES
MD5 : E4B3079A885277BC41CA39C30A0658EA
SHA1 : FDBF164757B1B686BAD7B66E25ACE4CA3F8C820B
SHA256:
E1D71D92B8E07F6FF9177071AF00C3E784B19817BF3231A1AB3FD49EAB

]

1 TABLE OF CONTENTS

DISCLAIMER/LICENSE

All code included with this tutorial is free to use and modify; we only ask that you mention where you found it. This tutorial is also free to distribute in its current unaltered form, with all the included supplements.

We have potentially illegal stuff inside. All the commercial programs used within our tutorials have been used only for the purpose of demonstrating the theories and methods described. These documents are released under the license of not using the information inside them to attack systems of programs for piracy. If you do it will be against our rules. No distribution of patched applications has been done under any media or host. The applications used were most of the times already been patched by other fellows, and cracked versions were available since a lot of time. ARTeam or the authors of the papers shouldn't be considered responsible for damages to the companies holding rights on those programs. The scope of this document as well as any other ARTeam tutorial is of sharing knowledge and teaching how to patch applications, how to bypass protections and generally speaking how to improve the RCE art. We are not releasing any cracked application. We are not at all encouraging people to release cracked applications; damages if there will be any have to be claimed to persons badly using information, not under our license.

This disclaimer applies to all ARTeam releases and tutorials!

VERIFICATION

ARTeam.esfv can be opened in the ARTeamESFVChecker to verify all files have been released by ARTeam and are unaltered. The ARTeamESFVChecker can be obtained in the release section of the ARTeam site: http://releases.accessroot.com

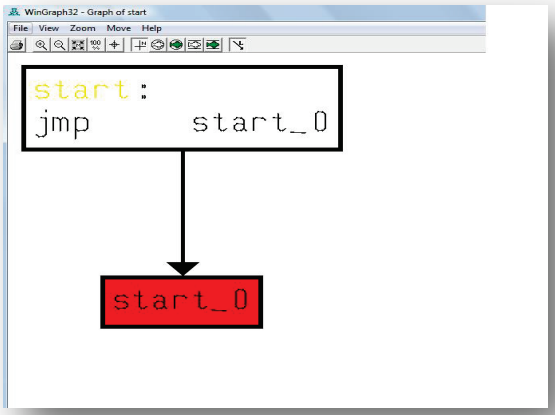
2 THE ESSAY

Let start the argument again from the beginning. **Trojan-GameThief.Win32.Magania**, according to Kaspersky naming convention, monitors the user activities trying to obtain valuable information from the affected user, especially about gaming login accounts. From a functional point of view you can check this post also:

http://evilcodecave.blogspot.com/2009/08/how-fastly-bypass-hostile-code-for.html

where you can find also files and registry entries to Remove Megania.

In this paper we will analyse more deeply the structure of this malware, especially the polymorphic part that represents a typical sample of hostile code. Starting from the first load into IDA we can see that Megania's **PE structure** and **Import Table destroyed**, this is how looks from WinGraph:



As you can see IDA shows that's we're facing a crippled structure!

The first thing to do is then to analyze how's the code around the entry point (CTRL+E)

```
.sdata:0043215A                 jmp     start_0

FUNCTION CHUNK AT .sdata:00432166 SIZE 00000011 BYTES

.sdata:00432099

.sdata:00432099                 push    esi

.sdata:0043209A                 push    ecx

.sdata:0043209B                 call    $+5
```

Above snapshot from IDA shows in **Red** the detached function chunks; in some cases IDA has problems resolving the functions, this is one of those situations. This happens because Function Chunks are provided for improving code locality but IDA considers it as continuous functions, the result is an isolated function that is missing all the links, this is why for IDA execution ends after **jmp start_0**. Function Chunks like these could be rectified using an IDA Python Script, but in this case we will resolve only a portion of the problem, and we'll not benefit much trying to master the Magania code. We will instead adopt a dynamic analysis. To reverse Megania you can use Syser or OllyDbg both works great, I used Olly 2 beta to dig the polymorphic code.