Security Solution for False Antivirus Detection

CATALIN POP, MARIUS POPESCU, ANTOANELA NAAJI
Faculty of Natural Sciences, Engineering and Computer Science
"Vasile Goldis" Western University
Arad 310025, Bd. Revolutiei nr. 94-96
ROMANIA
pop_catalin88@yahoo.com, popescu.marius.c @ gmail.com, anaaji@uvvg.ro, http://cs.uvvg.ro

Abstract: - False antiviruses, also known as rogue, block access to PC operating systems, in which case a classic antivirus proves inadequate. The purpose of the software described in this paper is to improve the products already existing on the market by adding it to a security product, as a module, or to create other specific tools. The advantage of the "heuristic" security solution suggested is that it requires very little memory (max. 1 Mb) and has no false alarms. The solution can be added to a "cloud" module, which finds an existing file in the list and sends it to an analysis lab, together with all files that its interacts with.

Key-Words: - False antivirus, security solution, rogue

1 Introduction

In computing, the term "virus" is applied to various software programs, which can be found under the generic name of malware; users employ the term "virus" instead of "malware" [3], [5], [6], [7]. False antiviruses are part of the Trojan category, having a lifespan ranging from a few hours to a week. They are classified into two categories: FakeAV (copies of legitimate security programs or some other programs that seem to be security programs) and Fake Alert (alerts warning users that the PC is infected).

False antiviruses disguise themselves as: antiviruses, anti spyware, anti adware, and, in some cases, even as computer maintenance programs.

The best known methods for creating alerts are:

- falsely creating a Security Center and warning the user that the antivirus year is not activated or not installed;
- prompting messages in the tray area, warning of security issues.

Recently, a new method appeared, displaying a window that warns the user that there is a security issue and that a PC driver scan is recommended.

2 The "False Antivirus" Application

2.1 False Antivirus Analyzing Lab

The false antivirus analysis lab used in this paper is made up of two computers: a virtual one (implemented using the Oracle VM VirtualBox application) and a dedicated one (used only for tests).

The research used the Windows XP operating system. As in the case of the virtual computer, the operating system is installed only once, after which a backup application is used, the resulting file being then copied to an external drive (USB). The tools employed are open - source and depend on the type of malware analyzed.

If the computer runs the analysis module, then it is much more efficient to use the Ultimate Boot CD application (collection of applications run from a CD, in the memory), because the behavior can be viewed in real time. Tools employed in this case are: *RegShot* (open - source program that can create snapshots and compare registries), *Autoruns* (utility for viewing and changing programs that are launched on startup), *ProcMon* (utility offering real-time information on active processes), *Rootkit Revealer* (a program offering advanced detection of rootkits) and *GMER* (a program that can search rootkits in key parts of the operating system installed on the PC).

Regardless of which computer the analysis is run on, one must first create a backup of registries and then save it on an external drive [9]. After creating several administrator accounts, the file to be analyzed is run, and then the computer is reset. If the computer does not start, then the *Ultimate Boot CD* application will be used to make a copy of registries (which will be compared to the first), using *RegShot*; then log is analyzed to establish why the PC does not start (in most cases, the *winlogon.exe*, *userinit.exe* files are copied and the computer starts).

In the report provided by the *RegShot* tool, a few files are added from the list of infected files, which are to be deleted; *OpenRogueRemoval* may also delete some rootkits, which appear if specialized software is used *(GMER* and *Rootkit Revealer* – programs which detect rootkits).

2.2 Presentation of the "False Antivirus" Application

False antiviruses create the same files, the only difference is their name [1]; this can be seen observed similarly when the *ZBot* file is runing (a very dangerous botnet).

The application has four components: *scan, view startup, view active processes* and *check system files*. The program is written in C++ [4], and designed for use on Windows operating systems (XP, Vista and 7).

Upon scanning, the user has the option of scanning a default file or creating a new file. The user may enter the destinations of files that he wishes to delete into a file; it is recommended for deletion to occur in safe mode, because, if the file is active, then normal deletion is not possible (an example is the executable file *"explorer.exe").*

View startup - is a window used for viewing programs that are launched at startup; this does not use the direct method for interrogating registries, but a command of the Windows operating system:

```
wmic startup get caption,
command>StartupLog.txt
```

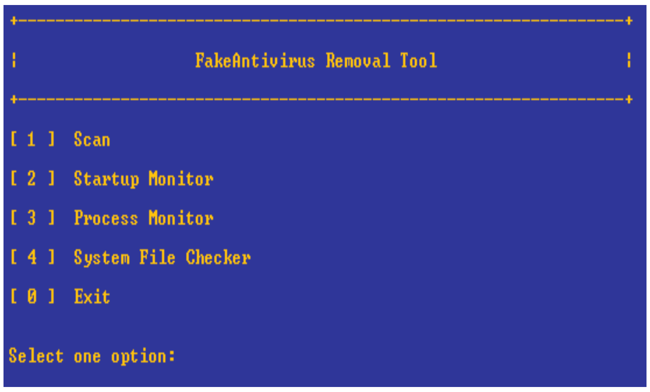
View active processes - is an alternative to the *task manager*, intended for *command prompt*, which lists all running processes, their name and PID (*Process* ID). Its purpose is to access processes, even if the computer is infected.

Check system files - is an option that requires the operating system installation CD. It is only used if the computer is greatly infected and only after disinfection or in cases where the operating system has problems starting. The option uses a Windows command and has a disadvantage for PCs where *windows updates* is on, as updates change viral files (those in the system). This option will replace those files, and on the next *update* they will be downloaded again.

The menu of the application contains the classic components, the last option being exit. Each option is built so that, after each operation, the program is closed. In order to use a new option or even the same one, the program must be run again. After each option, the memory is not too much loaded, thus providing the possibility to run this program very easily. Menu structure is based on the following sequence of C + + program:

```
void   option1();   void   option2();   void
option3(); void option4(); int main(){Menu-
GUI}
```

Option1(), corresponding to the scanning module, contains the strategy by which files are deleted. *Option2()* lists and ends active processes. *Option3()* lists the programs in *startup* and writes them into a log file. *Option4()* checks system files, requiring the operating system installation CD. In the *main* section of the program, the interface is drawn (Fig. 1). After the interface has been created, the selection option was added.

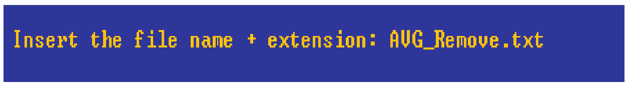


```
+-------------------------------------------------------------+
|                  FakeAntivirus Removal Tool                  |
+-------------------------------------------------------------+
[ 1 ]  Scan
[ 2 ]  Startup Monitor
[ 3 ]  Process Monitor
[ 4 ]  System File Checker
[ 0 ]  Exit

Select one option:
```

Fig. 1. Menu interface

Scan (Fig. 2). The working principle of the scanning engine consists of drawing information from a file and deleting the file. For the application to delete malware, the engine reads the entire line, regardless of the type of characters contained, thus deleting the entire malware file. After having defined all variables, the file can be entered manually, the input form being "name.extension". The file is browsed, transforming char characters into string types, so that the file might be deleted. If an error occurred upon deleting the file, then the message "Not Deleted" is displayed, then the message "File Deleted" is shown, then "Pause" is entered, for the user to be able to view, in detail, which files were deleted and which were not, the file being eventually closed.



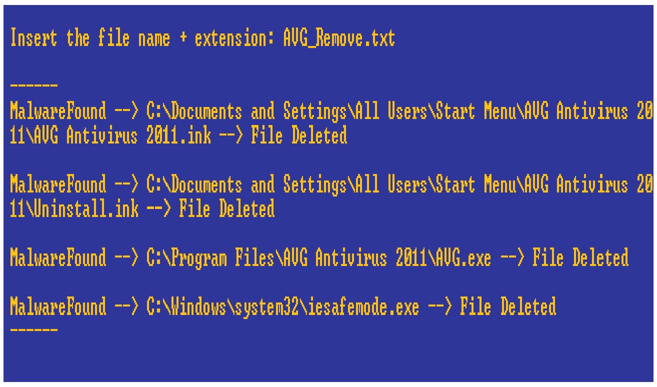
```
Insert the file name + extension: AVG_Remove.txt
```

Fig .2 Explanatory interface on scanning and neutralizing a false antivirus

For example, a file is shown in which the locations of the false antivirus "AVG 2011" were entered (*FakeAV_AVG2011.txt*):

```
C:\Documents and Settings\All Users\Start
Menu\AVG Antivirus 2011\AVG Antivirus
2011.lnk
C:\Documents and Settings\All Users\Start
Menu\AVG Antivirus 2011\Uninstall.lnk
C:\Program Files\AVG Antivirus 2011\avg.exe
C:\WINDOWS\system32\iesafemode.exe
```

If the user's PC is infected with false antivirus *AVG Anti-Virus 2011*, both on Windows XP and on Windows Vista or Windows 7, the destinations can be added to a single file; the only difference is the destination user name (Fig. 3). In this case, regardless of the operating system, in order to uninstall this false antivirus (using the application presented in the paper), one must enter *safe mode* with *command prompt*, since this malware is injected into the executable file *explorer.exe.*



```
Insert the file name + extension: AVG_Remove.txt

------
MalwareFound --> C:\Documents and Settings\All Users\Start Menu\AVG Antivirus 20
11\AVG Antivirus 2011.lnk --> File Deleted

MalwareFound --> C:\Documents and Settings\All Users\Start Menu\AVG Antivirus 20
11\Uninstall.lnk --> File Deleted

MalwareFound --> C:\Program Files\AVG Antivirus 2011\AVG.exe --> File Deleted

MalwareFound --> C:\Windows\system32\iesafemode.exe --> File Deleted
------
```

Fig. 3. Explanatory interface on the false antivirus AVG 2011 neutralization

Startup. This informative module creates log file, listing all entries to the *startup* file. Its purpose is to provide an alternative to products already existing on the market. In case the computer is infected with a false antivirus, blocking access to viewing *startup* entries, the module offers the possibility to view entries using safe mode with *command prompt* by *option2()* from the main menu.

The user has to choose between two variants (if the file already exists, overwrite it, and if not, the file will be created):

```
        if(startupselect == 1)
         {system("wmic startup get
caption,command > StartupLog.txt");
          cout<<"StartupLog.txt was created
\n"; system("pause"); }
        exit(1);}
```

The *wmic* command can create log files, not only in the current directory. In this case, the log file is created in “C: \”. The reason why it was chosen for the log file to be generated in the same place as the application is so that it may allow to create the file, regardless of the device that is running the program (to avoid incompatibilities). An example of a file generated on an uninfected computer is:

```
Caption
Command
Sidebar  %ProgramFiles%\Windows
Sidebar\Sidebar.exe/autoRun
Sidebar  %ProgramFiles%\Windows
Sidebar\Sidebar.exe/autoRun
DAEMON Tools Lite "C:\Program Files\DAEMON
Tools Lite\DTLite.exe" -autorun
ctfmon.exe
C:\WINDOWS\system32\ctfmon.exe
Malwarebytes'Anti-
Malware(reboot)"C:\ProgramFiles\Malwarebyte
s'Anti-Malware\mbam.exe" /runcleanupscript
COMODO Internet Security"C:\Program
Files\COMODO\COMODO Internet
Security\cfp.exe" -h
SunJavaUpdateSched  C:\Program Files\Common
Files\Java\Java Update\jusched.exe"
Caption - name of application
Command - destination and parameters (for
example,-h rund hidden)
```

Active Processes. The module offers the possibility to view and end a process, and it does not work if run from an antivirus CD or from similar CDs. The option may be used on an infected computer, provided that 2-3 accounts are created beforehand, with the administrator's permission. *Option3()* has the following program sequence:

```
void option3()
       {system("cls");
       int processelect;int pidselect;

SetConsoleTextAttribute(GetStdHandle(STD_OU
TPUT_HANDLE),15);
cout<<endl<<"List    of    Active    Processes
"<<endl;
```

The *handler: CreateToolhelp32Snapshoot* is used to make a *read-only* copy of objects in the system memory, thus:

```
       HANDLE
hSnapShot=CreateToolhelp32Snapshot(TH32CS_S
NAPPROCESS,0);
       BOOL    WINAPI    Process32Next(HANDLE
hSnapshot,LPPROCESSENTRY32 lppe);
```

where: *Process32Next* takes information from the following process, and *hSnapShot* is a *snapshot handler* [8]. *TH32CS_SNAPPROCESS* includes the list of *snapshot* processes [7], and *lppe* is a pointer to a structure. *PROCESSENTRY32* returns the *true* function if the data on the first process were copied into the *buffer:*

```
       PROCESSENTRY32*       processInfo=new
PROCESSENTRY32;
 processInfo>dwSize=sizeof(PROCESSENTRY32);

while(Process32Next(hSnapShot,processInfo)!
=FALSE)
       {
SetConsoleTextAttribute(GetStdHandle(STD_OU
TPUT_HANDLE),8);

       cout<<endl<<"===================";

       SetConsoleTextAttribute(GetStdHandle
(STD_OUTPUT_HANDLE),15);
        cout<<endl<<"+--
>Name:"<<processInfo->szExeFile;
        cout<<endl<<"+--
>PID:"<<processInfo->th32ProcessID;}
```

ProcessInfo is a member of the structure *dwSize = sizeof (PROCESSENTRY32),* which provides information on processes and browses up to the last (after which the name and *PID* are displayed). After listing the processes by *name* and PID (compound listing of all processes at that time, being a "copy" of the processes and not a real-time listing), the possibility to end a process by PID was implemented, and listing is identical to that in *Task Manager:*

```
       SetConsoleTextAttribute(GetStdHandle
(STD_OUTPUT_HANDLE),8);
       cout<<endl<<"=====================";

       SetConsoleTextAttribute(GetStdHandle
(STD_OUTPUT_HANDLE),15);
       cout<<endl;  cout<<"[1]  Complete  the
process"<<endl;
       cout<<"[0]         Exit"<<endl<<"+--
>";cin>>processelect;
       if  (processelect==1)
       {cout<<"Complete   the   process   with
PID\n";cout<<"===>";cin>>pidselect;
```

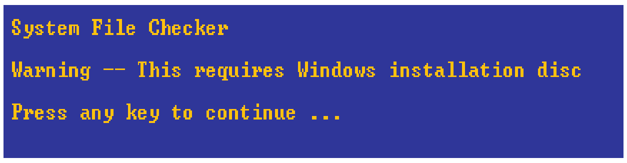
In order to end the process, the *process handler* is used*:*

```
HANDLE
proces=OpenProcess(PROCESS_TERMINATE,FALSE,
pidselect);
        TerminateProcess(proces,0);
        CloseHandle(proces);
        cout<<"[!]
Process"<<pidselect<<"complete"<<endl;}
```

For the user to view the processes without exiting the application, the exit option was introduced.

Check system files (Fig. 4). This option, in order to be functional on Windows XP, Vista and Windows 7 operating systems, uses the *sfc* command provided by Microsoft. The option has three parameters: */scannow* (scans all system files immediately), */scanonce* (scans all system files once), and */scanboot* (scans all the system files each time the computer restarts). Since all parameters require administrator rights, the */scannow* parameter is used, entering a break beforehand:

```
void option4()
{system("cls");   cout<<"Verifying   system
files \n\n"<<endl;
cout<<"Attention, this option requires the
operating      system      CD      \n\n"<<endl;
system("pause");system("sfc//scannow");exit
(1);}
```



```
System File Checker

Warning -- This requires Windows installation disc

Press any key to continue ...
```

Fig. 4 Check system files option

3 Future direction of development

Further amendments need to be made to the program, so that the user:

- would not open the file with strict rules and would be able to enter the username or administrator name (automatically detected by the program);
- would add real-time running options to the *startup* file and minimize it to the *tray* area, or start with attribute "-h" (if it finds a file that is in the list, then notify the user with the message *"Potential virus found");*
- could add a new option checking if *hostfile* was changed, and then sending a message to the user. *Hostfile* restricts access to certain sites. Viruses change this file and add security product addresses to it, because this way the "victim" cannot download a program to disinfect the computer;
- could add the *open source ClamAV* antivirus to the program, with the specification that the entire interface and options would be modified so as S to run in *safe mode* with *command prompt* (the *"Scan with ClamAV"* option will appear in the menu*).*

The idea is to try to create a program based on the fact that he same false antivirus involves the same files; the file is encrypted and sent *in wild* as a *0-Day Malware* file (0-Day Malware is a virus that manages to infect the computer out of lack of signatures or other virus detection methods featured in the antivirus [2]). A classic signature-detection antivirus cannot "catch" this malware on time.

Due to the fact that the program is *open-source,* any programmer can contribute both files and rules in order to improve it.

4 Conclusions

We implemented a program used for cleaning the computer of false antiviruses or false alerts, a program which aids current security solutions. In addition to this, the software also provides options that help the user, through a combination of *command prompt* and *console application,* in order for it to be run directly from the console.

The application is meant for users with advanced knowledge in computing, providing the possibility to create rules for different false antiviruses. Lists intended for one type of false antiviruses may contain different versions of the same antivirus (in most cases, viral components have a targeted user and operating system), in which the user must only change the username, or create a user with administrator rights. As regards the *startup*, the application provides more information than the *msconfig* file and has the advantage that it can be used without and interface (from the *command prompt).* Upon the listing processes, the application offers a great advantage, as viruses disable *msconfig, task manager* and *regedit,* and if the computer has another user with administrator rights, this second user can run this program and end that process. Checking protected files (system files or viral components), in case a virus is changed or injected, is a pretty good solution.

Another advantage of this application consists in the possibility of running both in *safe-mode* and from an antivirus CD or similar CDs.

References:

[1] Sean-Paul Correll, S.P., *Business of Rogueware,* PandaLabs research, 2009

[2] Halvorsen, F.M, *0-Day Malware,* Project from Norwegian University of Science and Tehnology, 2008

[3] Newman, J., *MacDefender* http://www.pcworld.com/article/226846/fake_macdefender_brings_malware_to_macs.html

[4] Popescu, M.C. *Aplicatii in informatica,* Universitaria Craiova, 2009

[5] Sanabria, A., *Malware Analysis Environment Design and Artitecture*, SANS Institute, 2007

[6] Szor, P. *The Art of Computer Virus Research and Defense*, Addison Wesley Professional, 2005

[7] *** *CreateToolhelp32Snapshot* http://msdn.microsoft.com/en-us/library/ms682489%28v=vs.85%29.aspx

[8] *** *Process32Next* http://msdn.microsoft.com/en-us/library/ms684836%28VS.85%29.aspx,

[9] *** *Detcraig, TrendMicro* http://community.trendmicro.com/t5/Web-Threat-Spotlight/Cross-Border-Korean-Shelling-Searches-Lead-to-FAKEAV/ba-p/20928