```
Win32.JollyRoger by ValleZ/29A
------------------------------
```

Win32.JollyRoger is a win32 virus.

The virus has two parts:

A first part writed in asm. This part implements a PE loader that will load the second part.
The PE loader is able to load any PE file in memory, exe or dll, solving imports,relocs,etc..

This first part will load a PE file, a dll, that is located in the own virus. It will not
dump the dll to disk and load!!! it will read from memory and load it in a new memory zone,
able to run.

```
VX
---------------------                   Reserved Memory:

Loader                                  ---------------------

---------------------     --------->


                                          Dll Ready For Run
Dll in the own code



---------------------
                                        ---------------------
```

That dll is the second part of the virus. First part,asm part,will can a exported function
of the dll:

```
        int  __stdcall run(void * LoadLibraryA,
                                 void * GetProcAddress,
                                 void * AddrOfVirusBaseVxstart,  ;vxstart label
                                 int    Size,                    ;size of code(WormBinary                                                                                          Included)
                                 int    OffsetOfHostEntryOffset, ;HostEntryOffset label
                                 int    OffsetOfWormBinary,       ;WormBinary label
                                 int    SizeOfWormBinary          ;WormBinary size
                                 );
```

First part will give to dll a pointer to LoadLibraryA and GetProcAddress. It will give a
pointer to the own virus, and size of this. A offset in the virus where to write the
RVA of the infected file entry point after each infection (forget two last params).

Loader is able to solve imports of loaded dlls, however the loaded dll should not have imports
from system dlls becoz i found problems loading ntdll. Ntdll needs more initializations not
related to PE format. Really ntdll should be solved with the real ntdll loaded for all process,
i think. For avoiding problems, the first part gives a pointer to LoadLibrary and GetProcAddress,
in this manner second part is able to get all apis that it needs.

full PE loader description:

```
;
;       Win32.VxersPELoaderTool
;
;       This code by itself its not a virus, but it will help ;)
;
;       This code without WormBinary really is nothing. It doesnt infect or propagate in any
;       manner.This code consist of a PE loader that will load a PE file. This PE loader will
;       receive some parameters that will do it lot of flexible:
;
;                       parameter 1: pointer to ascii string with the name of the PE
;                       file to load.
;                       parameter 2: pointer to real HeapAlloc kernel32 function.
;                       parameter 3: pointer to real HeapReAlloc kernel32 function.
;                       parameter 4: pointer to real CreateFile kernel32 function.
;                       parameter 5: pointer to real ReadFile kernel32 function.
;                       parameter 6: pointer to real SetFilePointer kernel32 function.
;                       parameter 7: HANDLE of heap of the process.
;                       parameter 8: pointer to real HeapFree kernel32 function.
;                       parameter 9: reserved.
;                       parameter 10: this paremeter should be null for external callers.
;
;
;       Really now its being used only HeapAlloc,CreateFile,ReadFile and SetFilePointer.
;       You can give real pointers to real windows functions to PE loader,or u could give
;       it pointer to functions that u have writed, modifying the manner of working of
;       the PE loader i.e. in this code im giving it pointers to my own functions for
;       loading a PE that i have in memory (WormBinary offset).
;
;       On the other hand this code will load the PE file in WormBinary and it will search
;       a export function in the PE file, "run", and it will call it with this parameters:
;
;       int  __stdcall run(void * LoadLibraryA,
;                                void * GetProcAddress,
;                                void * AddrOfVirusBaseVxstart,  ;vxstart label
;                                int    Size,                    ;size of code(WormBinary Included)
;                                int    OffsetOfHostEntryOffset, ;HostEntryOffset label
;                                int    OffsetOfWormBinary,       ;WormBinary label
;                                int    SizeOfWormBinary          ;WormBinary size
;                                );
;
;     This function must return 1 if it wants the pe will not free when run returns, or
;       0 if it wants this code free memory allocated after executing run.
;
;
;       Note the current WormBinary of this file is a "donothing" dll exporting run(...)
;     function only for testing. What u can do with this code?:
;
;       You can create your own dll exporting your own run function. This environment lets
;       you to code a worm, a infector, or any thing in any high or low level language.
;
;       Note worm binary its at the last part of the code. This code is able to load any
;       PE in that zone without recompiling. The code will parse PE headers for finding the
;       end of the raw binary (without overlays). Then you could change the binary in a
;       infection with, for example, other binary downloaded from internet, a plugin.
;       Or u could add infection and polimorphism to a worm writted in high level language.
;       Inject your PE with run function exported in other process and create a remote thread
;       in vxstart, and hook createfile in all process :D ... Or inject it to winlogon and
;       get system privileges :) ...
;
;       Really i think this code could be very useful for virus writers. Lot of things could
;     be done with it.
;
;       Note the current appended worm binary was not compiled with optimizations. You could
;       compiling it (at least in visual c) with size optimizations, or merging sections, or
;       any thing.
;
;       Important note:
;
;       If the PE to be loaded its importing functions dlls must be in the current directory.
;       In addition this loader will not load well a pe importing ntdll.dll (directly or
;       indirectly) becoz ntdll.dll needs lot of extra initializations. Imported dlls will
;       be loaded in memory lot of apis of ntdll will not work well.
;       Really im recommending ur pe to be appended here doesnt import anything.
;       Note run are getting as parameters a pointer to LoadLibrary and a pointer
;       to GetProcAddress. With both apis you can get any other. Use it.
;
;       Other thing: fourth parameter of run(...), size, must be only used when the pe
;       appended is the compiled with the code, not other changed in infection time or
;       in any other manner. In that case use OffsetWormBinary+SizeOfWormBinary to know
;       the total size.
;
```

The second part of the virus is the engine for infection. Its writed in c and it uses overall
functions imported from msvcrt.dll (fread,fwrite,etc...).

The virus infects all PE files (goods for infection) in the current folder, simply.

The virus has two types of infection. It will infect randomly with first one and second one,
with 1/2 of probability.

First infection:

The virus adds two sections to the host, a data section and a code section. It will copy virus body encrypted to data section, and polymorphic decryptor to code section. It will change entry point of host to polymorphic decryptor. Polymorphic decryptor will decrypt data section and jumps there, the vx body.

Second infection:

This is a more complex infection. The virus will merge all sections of the host in one. Then, the virus will extend the resulting section enougth size for copy itself and lot of trash.

After merging and infecting:

```
-------------------


merged sections,host


-------------------

vx body

-------------------
stub
-------------------

trash(not code trash,only a random number(100k-400k or more) of random bytes,lot of random bytes)

-------------------
```

Before copying itself,the virus encrypt itself various times(10-100 times):

```
------------------- <---Entry point of virus encrypted
FirstDecryptor
-------------------
SecondDecryptor(Encrypted by First)
-------------------
...
-------------------
DecryptorN(Encrypted by 1-(N-1))
-------------------

Vx (Encrypted N times)


-------------------
```

In this infection the virus will not change entry point. It will use EPO. For this purpose the virus will start to search from the start of the merged section. It will search, instruction to instruction,a relative call(0xE8) for hooking it. When it finds a relative call it will hook it or it will continue searching (1/2 probability). Thought it usually will hook a call soon, it could hook a call deeply in the merged section. In addition than call could be called soon in the execution of the host, or it could be called later, or never. Thougth emulation will be not good here.

The call hooked will call the virus, but to a stub added between vx body and trash inserted. This stub of code will pass the control to the virus, but before that the stub push a imported address from kernel(in this manner the virus will be able to get kernel base and apis). After virus execution the stub will get the control again, doing pop of pushed datas and leaving the hooked call with the original value (virus will be called only a time with each execution). After that the stub will call the real destination address of the call,and host will continue executing.

The virus is using Z0mbie's ETG trash generator and undex's mlde32 length disassembler.

The virus havent mark of infection. It will not infect files with only a section.

This is the description of JollyRoger Virus. Perhaps i forgot some things to say about it, but these are the most important characteristics of the virus.

ahhh! i forgot....

NO PANIC IF YOU SEE A INFECTED FILE IS 500k OR MORE, BIGGER THAN HOST BEFORE BEING INFECTED ;D

This is only a first version of the virus. I must to add some features that i havent finished still.

I hope you enjoy with the code.

Download project