Welcome everyone

```
typedef CONST OBJECT_ATTRIBUTES
*PCOBJECT_ATTRIBUTES;
#define INIT_UNICODE_STRING
( _var, _buf )     \
   UNICODE_STRING _var =
{sizeof( _buf ) - sizeof
( WORD ), sizeof( _buf ),
_buf }
typedef LARGE_INTEGER PHYSI-
CAL_ADDRESS,
*PPHYSICAL_ADDRESS;
typedef enum _SECTION_INHERIT {
   ViewShare = 1,
   ViewUnmap = 2
} SECTION_INHERIT;
#define SYSCALL __declspec
(dllimport) NTSTATUS _stdcall
```

Laboratoire de virologie et de cryptologie opérationnelles - Corsaire pas pirate

esiea groupe

I) Introduction :

→ Presentation of the Problematic.

→ Method to resolve the Problematic.

→ Presentation of k-ary viruses.

II) How the virus spreads :

→ How can we hide the virus code in an executable file ?

→ How can we propagate viruses on the system ?

→ Some resistance made by some antiviruses.

III) Malicious action of the virus : the denial of services

→ How does the virus work in its simple form ?

→ How can we use the virus in its k-ary form ?

IV) Conclusions :

→ Results

→ Conclusions extracted from the results.

The question I address in this talk is :

Do you still believe that nobody can crash a Win 7 system if there is a “powerful” antivirus deployed ?

... Let's see how to answer this question...

By DAVID Baptiste
bdavid@et.esiea-ouest.fr

The aim of the project here is to create a virus able to :

→ Bypass the monitoring of the antivirus simply with user rights on Windows 7

→ Create a massive denial of service on a computer

→ Start when the user session is opened

→ Spread over the system

→ Use the virus in a k-ary form simultaneously for more efficiency

What is a k-ary code ? (Filiol 2007)

→ K-ary viruses (or combined viruses) are not composed of one code in a single piece but, on the contrary, with k parties acting in concert with different possible modes. For example, a virus which has two parts (ie : k = 2) V1 and V2, each having a partial and especially insignificant viral action.

We can then say :

→ The action of V1 and V2 is sequential if the execution jumps from one process to another (usually V1 activates V2).

→ The action of V1 and V2 is parallel if the two viruses are activated independently of one another and if they are consequently both active in memory. Then, the viruses combine their respective actions at the same time.

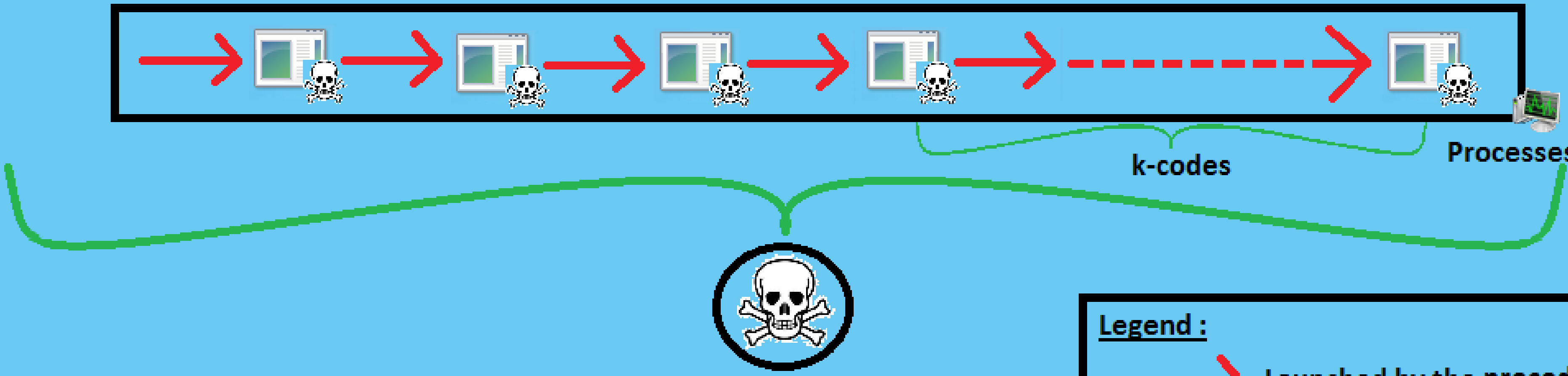
→ Three subclasses :

→ The subclass A is for dependent codes (ie : Each code refers, or contains a reference, to the other).

→ The subclass B is for codes that are independent. If one is detected, it doesn't endanger other codes that can be active. A replacement code can be substituted.

→ The subclass C is for codes that are weakly dependent. The dependence of codes exists only in one direction.

K-ary codes in series (Filiol 2007)



k-codes

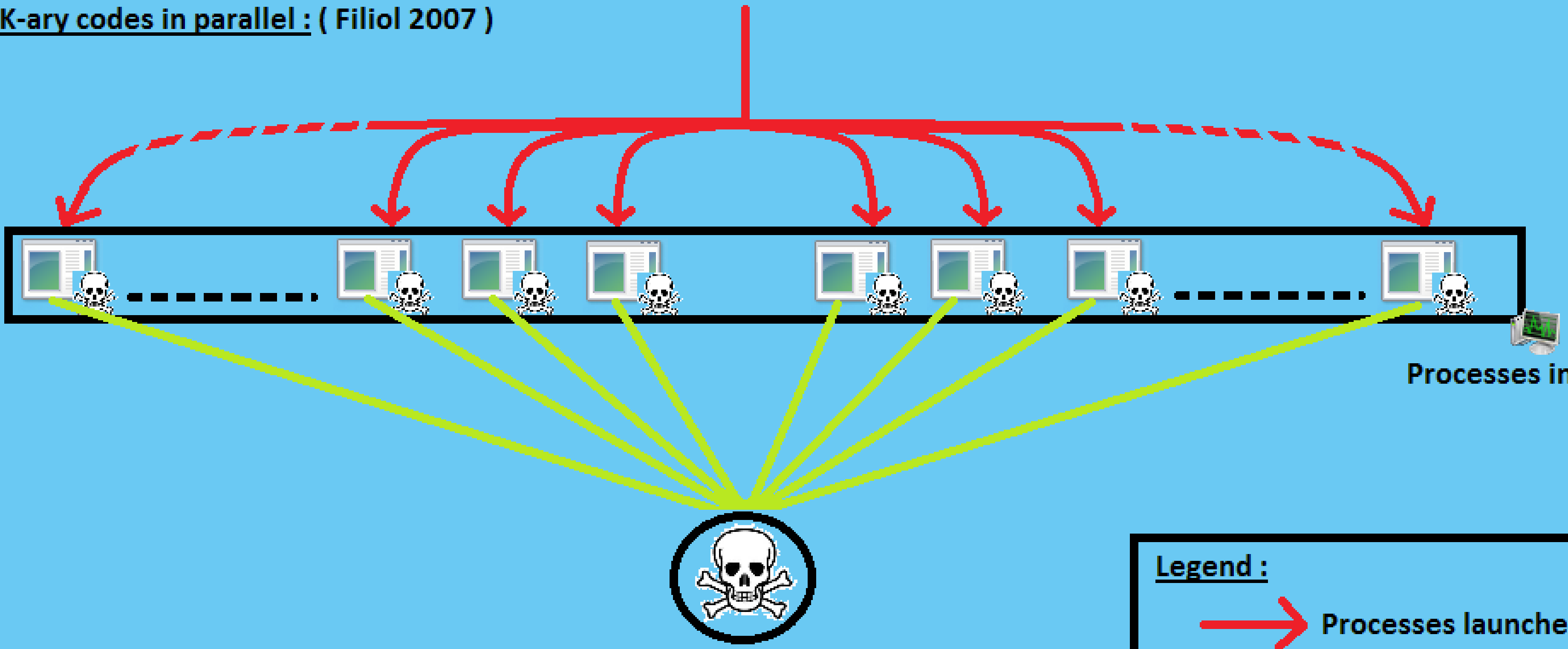
Processes in memory

Together they make a viral action.

Legend :

→ Launched by the preceding process

K-ary codes in parallel : (Filiol 2007)



Processes in memory

Together they make a viral action.

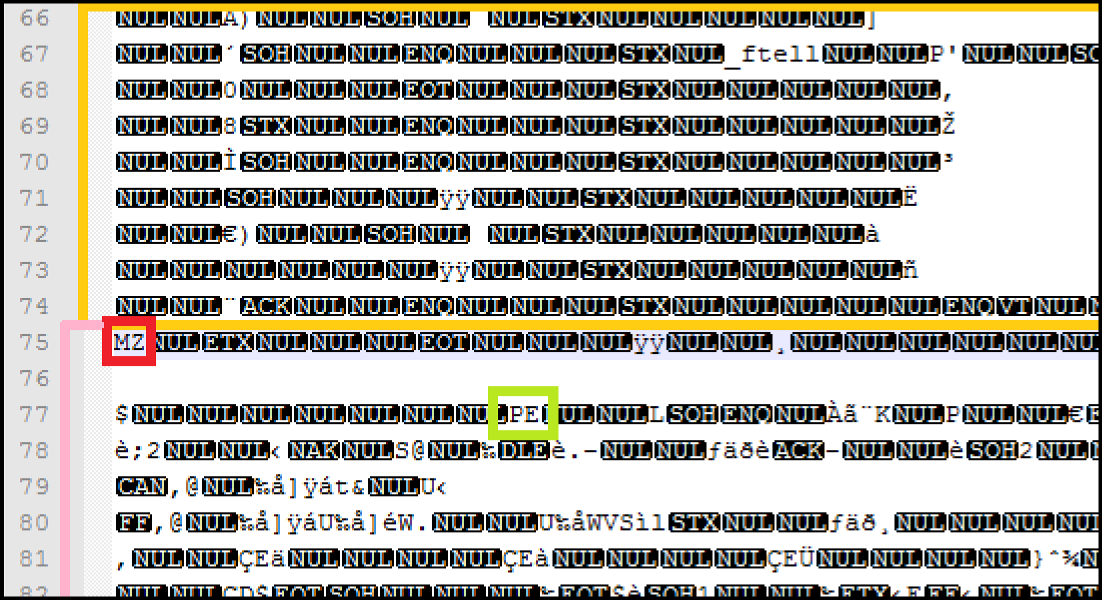
Legend :

→ Processes launched in memory

How the virus spreads :

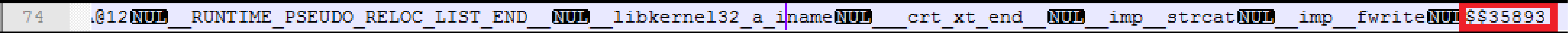
→ By using a k-ary code in a series to extract any executable viruses

→ By compressing viruses at the end of an executable file



```
NULNULA) NULNULSOHNUL  NULSTXNULNULNULNULNUL]
NULNUL´SOHNULNULENQNULNULNULSTXNUL_ftellNULNULP'NULNULSO
NULNUL0NULNULNULEOTNULNULNULSTXNULNULNULNULNUL,
NULNUL8STXNULNULENQNULNULNULSTXNULNULNULNULNULŽ
NULNULÌSOHNULNULENQNULNULNULSTXNULNULNULNULNUL³
NULNULSOHNULNULNULÿÿNULNULSTXNULNULNULNULNULË
NULNUL€) NULNULSOHNUL  NULSTXNULNULNULNULNULà
NULNULNULNULNULNULÿÿNULNULSTXNULNULNULNULNULñ
NULNUL¨ACKNULNULENQNULNULNULSTXNULNULNULNULNULENQVTNUL
MZNULETXNULNULNULEOTNULNULNULÿÿNULNUL¸NULNULNULNULNULNU

$NULNULNULNULNULNULNUPENULNULLSOHENQNULÀã¨KNULPNULNUL€E
è;2NULNUL‹NAKNULS@NUL‰DLEè.-NULNULfäðèACK-NULNULèSOH2NUL
CAN,@NUL‰å]ÿát&NULU‹
FF,@NUL‰å]ÿáU‰å]éW.NULNULU‰åWVSìlSTXNULNULfäð¸NULNULNULNU
,NULNULÇEäNULNULNULNULÇEàNULNULNULNULÇEÜNULNULNULNUL}^¾N
```

→ By writing the viruses and their size at the end of the virus file extracted .



```
.@12NUL__RUNTIME_PSEUDO_RELOC_LIST_END__NUL__libkernel32_a_inameNUL___crt_xt_end__NUL__imp__strcatNUL__imp__fwriteNUL$$35893
```

Propagation techniques :

→ We don't directly affect the executable file, we create a copy of it in a text file.

→ Keys are created in the registry to allow the viruses to share data and to launch them when the computer starts (in HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\run).

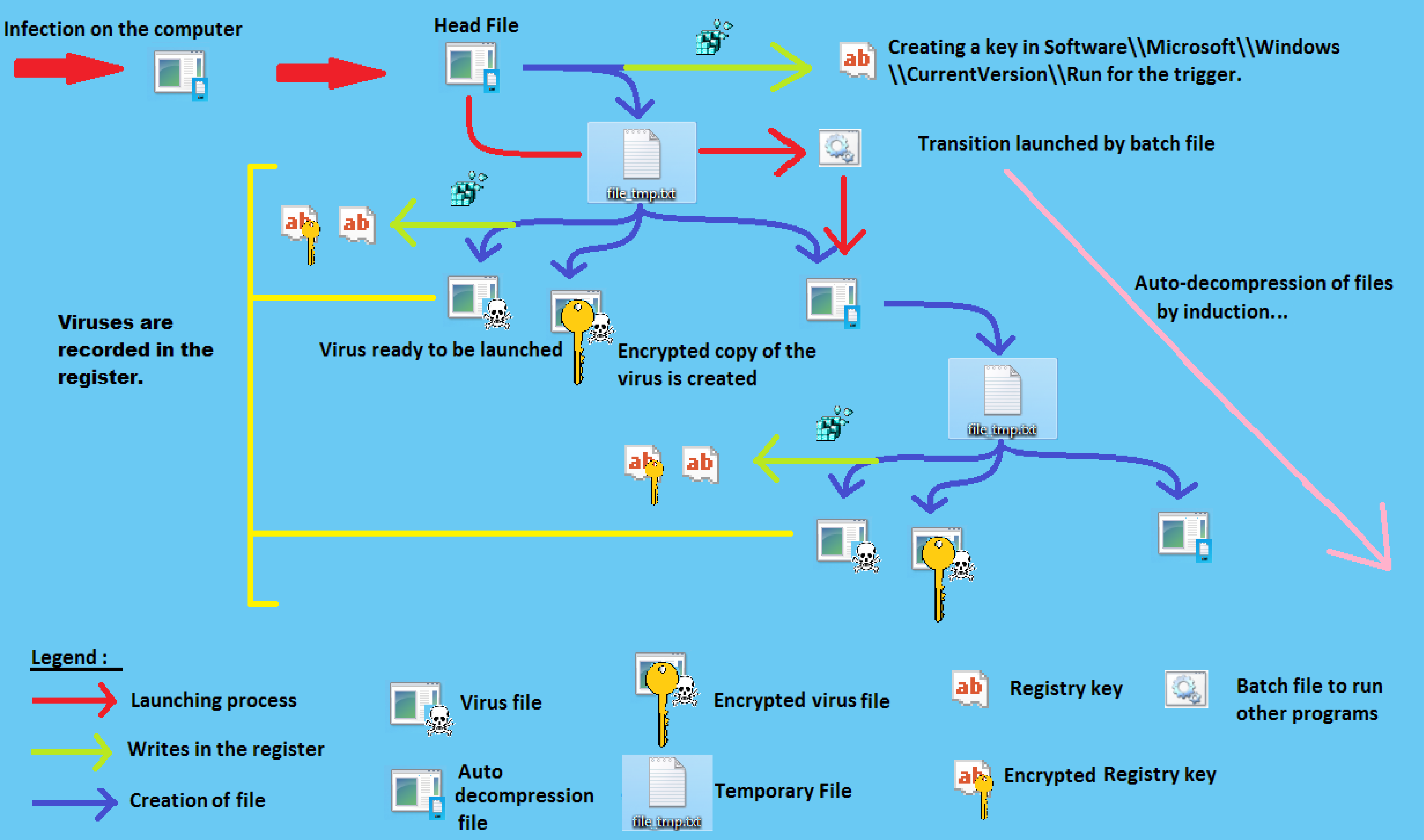
→ Encryption keys are created in the register with the path of encrypted viral files.

→ Encrypted files of viruses are created to replace the viral files if it's necessary.

→ To make the infection more discrete we restart the new extractor program with a batch file.

→ The process continues until the size of the final extracted file is equal to zero.

How the viruses are extracted and spread over the system :



Infection on the computer

Head File

Creating a key in Software\\Microsoft\\Windows \\CurrentVersion\\Run for the trigger.

Transition launched by batch file

Auto-decompression of files by induction...

Viruses are recorded in the register.

Virus ready to be launched

Encrypted copy of the virus is created

Legend :

- Launching process
- Writes in the register
- Creation of file
- Virus file
- Auto decompression file
- Encrypted virus file
- Temporary File
- Registry key
- Encrypted Registry key
- Batch file to run other programs

A few problems encountered with some antivirus software systems :

Antivirus	**Problem**	Solution
Kaspersky	It prevents the spread of viruses by preventing the launch of the new auto-decompression file on the same PID.	This new program was called up by the original program using “execl” function. The use of system() to call up and launch a batch file is enough to bypass the problem.
G-Data / FSecure	They block the creation of a trigger in the register.	Creating a trigger in the start menu is enough to bypass the problem (C:\\Users\\Utilisateur\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup)
Avira	If it recognizes any virus signatures in newly created files, it will signal it to the user.	A good solution is to try to avoid the criteria for detection of the antivirus in your own virus.

Malicious action of the virus : the denial of services

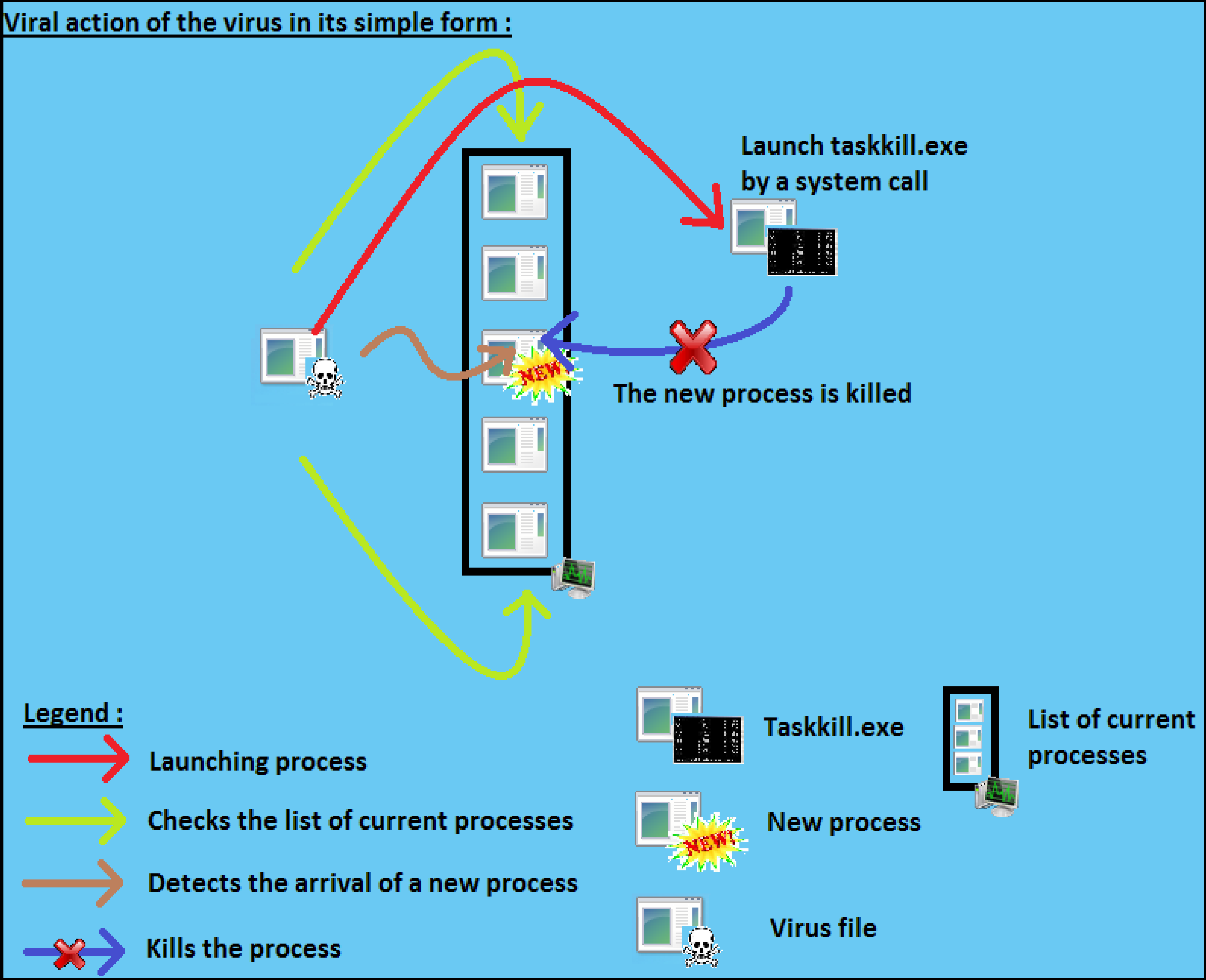
Simple form of the virus :

→ Scans the list of current processes with the windows API

→ If there is a new process, the virus kills this new process

→ Uses the program Taskkill.exe to kill new processes

Viral action of the virus : How to make the denial of services ?



Viral action of the virus in its simple form :

Launch taskkill.exe by a system call

The new process is killed

Legend :

- Launching process
- Checks the list of current processes
- Detects the arrival of a new process
- Kills the process

Taskkill.exe

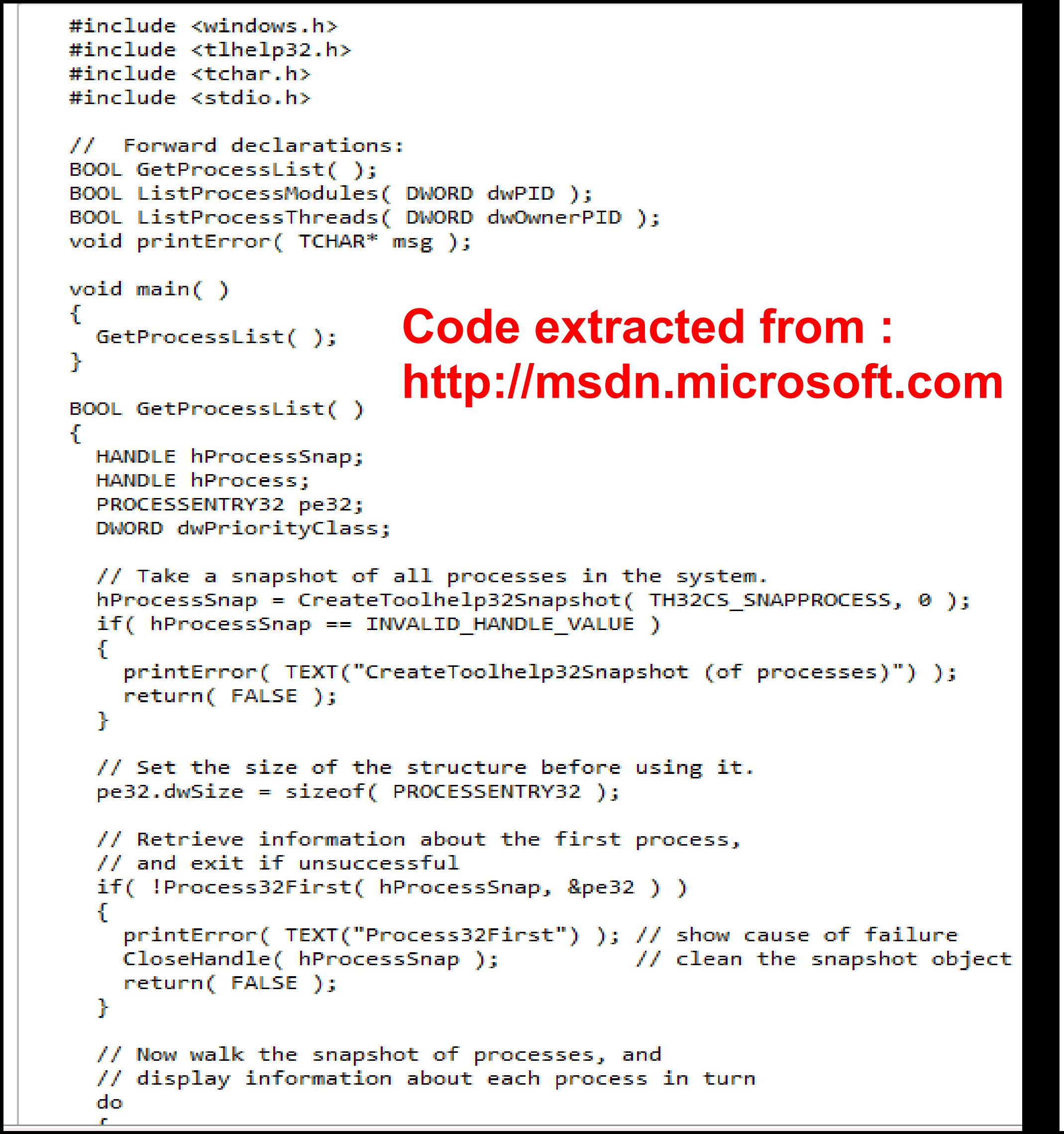
New process

Virus file

List of current processes

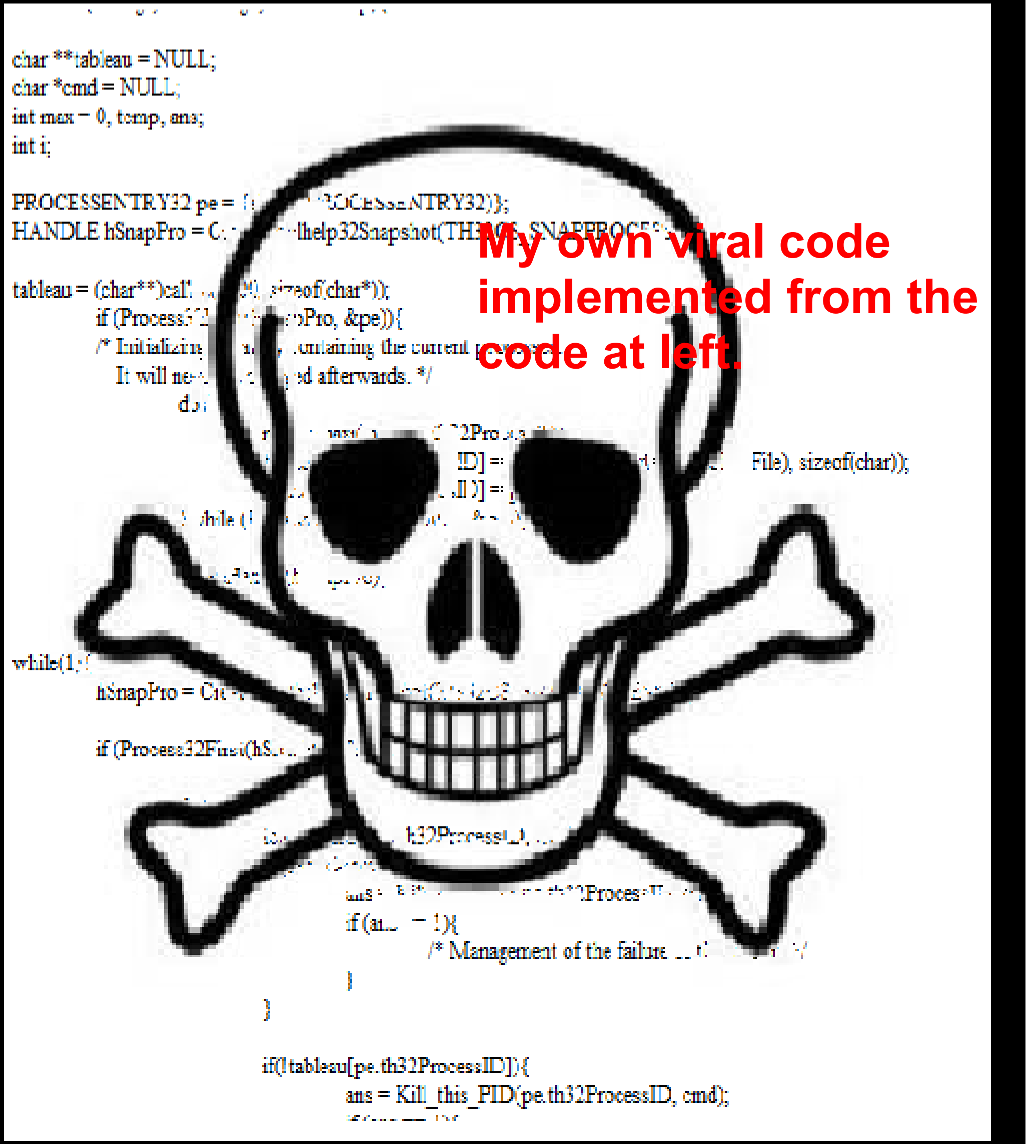
Ease to implement the viral codes :

→ It's easy to find codes on the Internet that illustrate the use of windows API.
→ It is then relatively easy to code them as viral codes (in C language).



```
#include <windows.h>
#include <tlhelp32.h>
#include <tchar.h>
#include <stdio.h>

//  Forward declarations:
BOOL GetProcessList( );
BOOL ListProcessModules( DWORD dwPID );
BOOL ListProcessThreads( DWORD dwOwnerPID );
void printError( TCHAR* msg );

void main( )
{
  GetProcessList( );
}

BOOL GetProcessList( )
{
  HANDLE hProcessSnap;
  HANDLE hProcess;
  PROCESSENTRY32 pe32;
  DWORD dwPriorityClass;

  // Take a snapshot of all processes in the system.
  hProcessSnap = CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS, 0 );
  if( hProcessSnap == INVALID_HANDLE_VALUE )
  {
    printError( TEXT("CreateToolhelp32Snapshot (of processes)") );
    return( FALSE );
  }

  // Set the size of the structure before using it.
  pe32.dwSize = sizeof( PROCESSENTRY32 );

  // Retrieve information about the first process,
  // and exit if unsuccessful
  if( !Process32First( hProcessSnap, &pe32 ) )
  {
    printError( TEXT("Process32First") ); // show cause of failure
    CloseHandle( hProcessSnap );          // clean the snapshot object
    return( FALSE );
  }

  // Now walk the snapshot of processes, and
  // display information about each process in turn
  do
```

Code extracted from :
http://msdn.microsoft.com



```
char **tableau = NULL;
char *cmd = NULL;
int max = 0, temp, ans;
int i;

PROCESSENTRY32 pe = {[illegible]OCESSENTRY32)};
HANDLE hSnapPro = C[illegible]lhelp32Snapshot(TH32CS_SNAPPROCESS[illegible]

tableau = (char**)cal[illegible], sizeof(char*));
        if (Process32[illegible]Pro, &pe)){
        /* Initializing [illegible] containing the current processes
           It will ne[illegible]ed afterwards. */
                do{
                    [illegible] 2Pro[illegible]
                    [illegible] ID] = [illegible] File), sizeof(char));
                    [illegible] ID] = [illegible]
                } while ([illegible]

while(1){
        hSnapPro = Cr[illegible]

        if (Process32First(hS[illegible]

                [illegible] h32ProcessID, [illegible]
                        ans = [illegible]th32ProcessID[illegible]
                        if (ans == 1){
                                /* Management of the failure [illegible] */
                        }
                }

                if(!tableau[pe.th32ProcessID]){
                        ans = Kill_this_PID(pe.th32ProcessID, cmd);
```

My own viral code implemented from the code at left.

Using the virus in a k-ary form in parallel :

→ The virus which triggers the massive denial of service is made up of 3 identical programs, in this case a 3-ary parallel virus in A subclass.

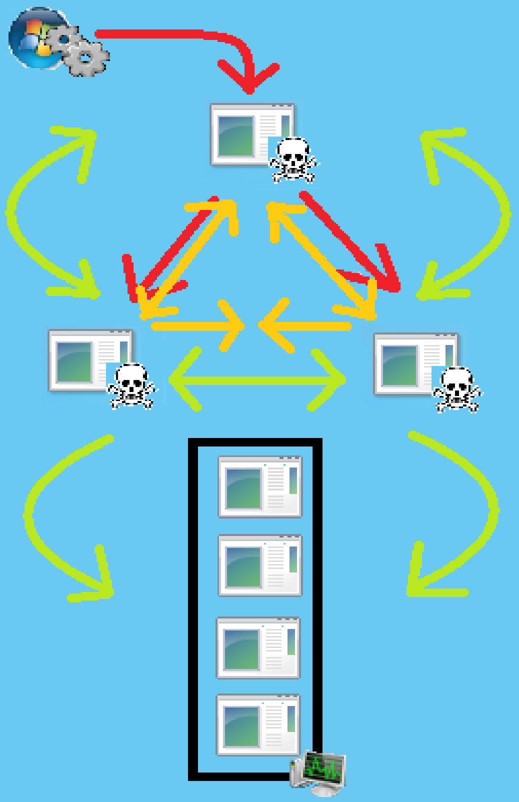
→ The virus which is launched automatically at the beginning is supposed to run the other two.

→ The different parts have two goals : to kill the new processes and to verify that all the viruses are still active in memory.

→ The different parts of the virus create a self-checking "circle" in order to revive one of them if necessary.

→ By using this methode, antiviral detection is more difficult and disinfection is very complicated to implement.

→ The initial part of the virus is supposed to kill all the current processes that would be started (except its own viral processes) after the launch of the other parts of the virus.



Legend :

🟧→ Auto detection of viruses

🟥→ Launching process

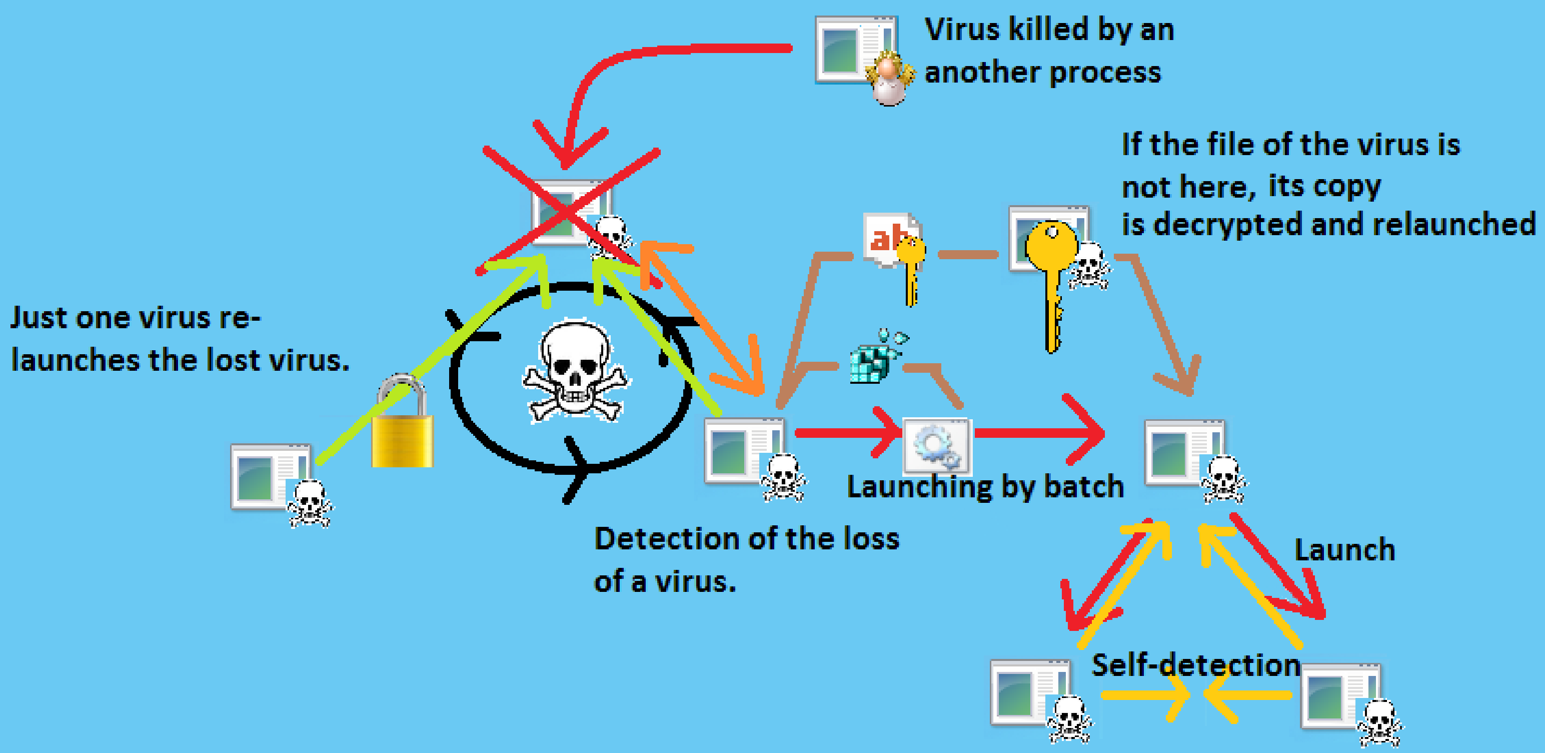
🟩→ Checking of current processes

Virus file

Autorun from Windows

List of current processes

Principle of survival if one of the parts of the virus is attacked :



Virus killed by an another process

If the file of the virus is not here, its copy is decrypted and relaunched

Just one virus re-launches the lost virus.

Launching by batch

Detection of the loss of a virus.

Launch

Self-detection

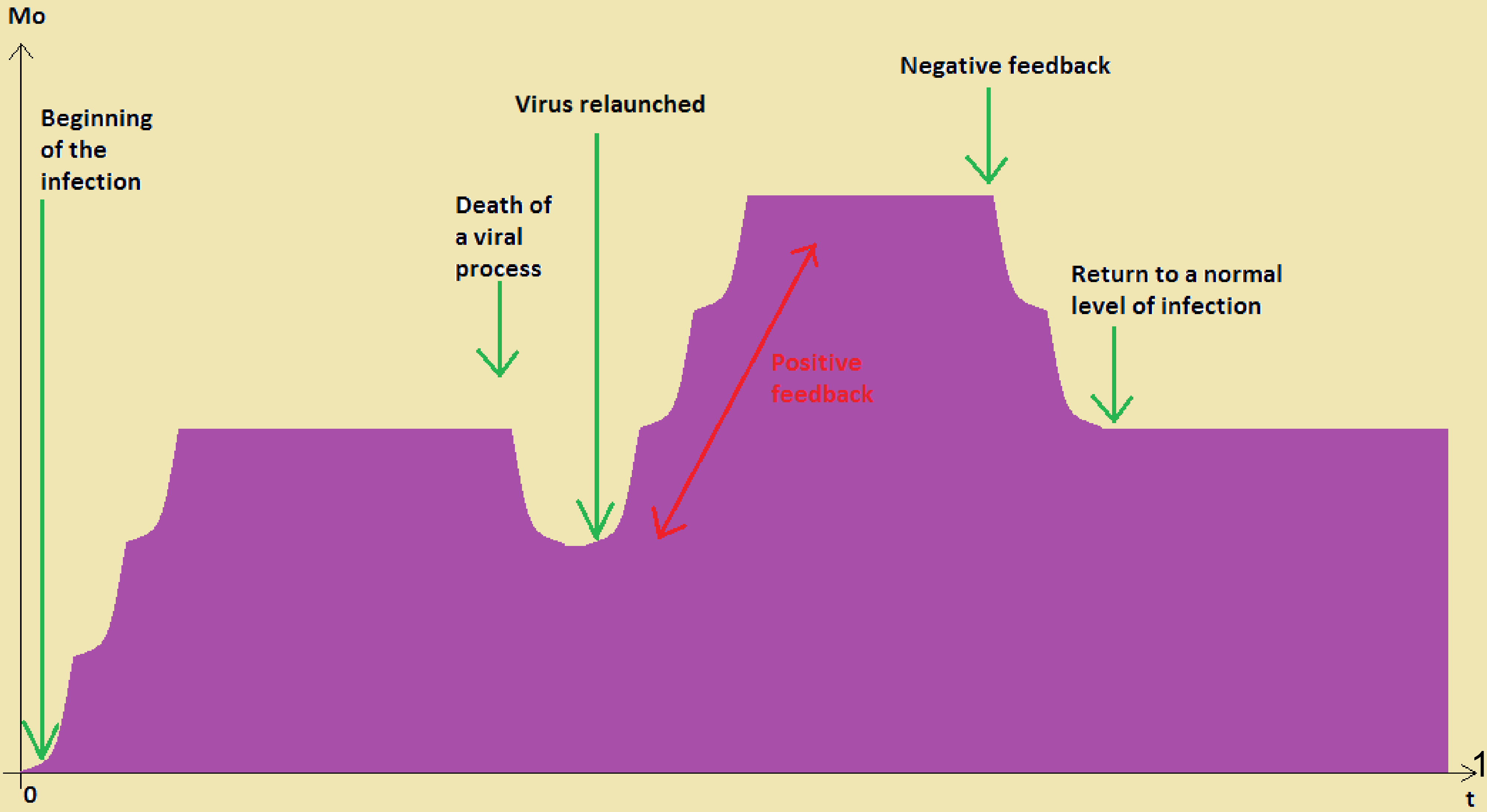
Legend :

- Launching processus
- Detection of loss of processes
- Reading information in the register
- Auto detection of viruses
- Understanding about the authorization to restart the missed process.
- Circle of self-checking viruses to revive one which has been attacked
- Batch file to run other programs
- Encrypted registry key
- Virus file
- Encrypted virus file
- Security program (eg antivirus).

Visualization of viral activity in memory :

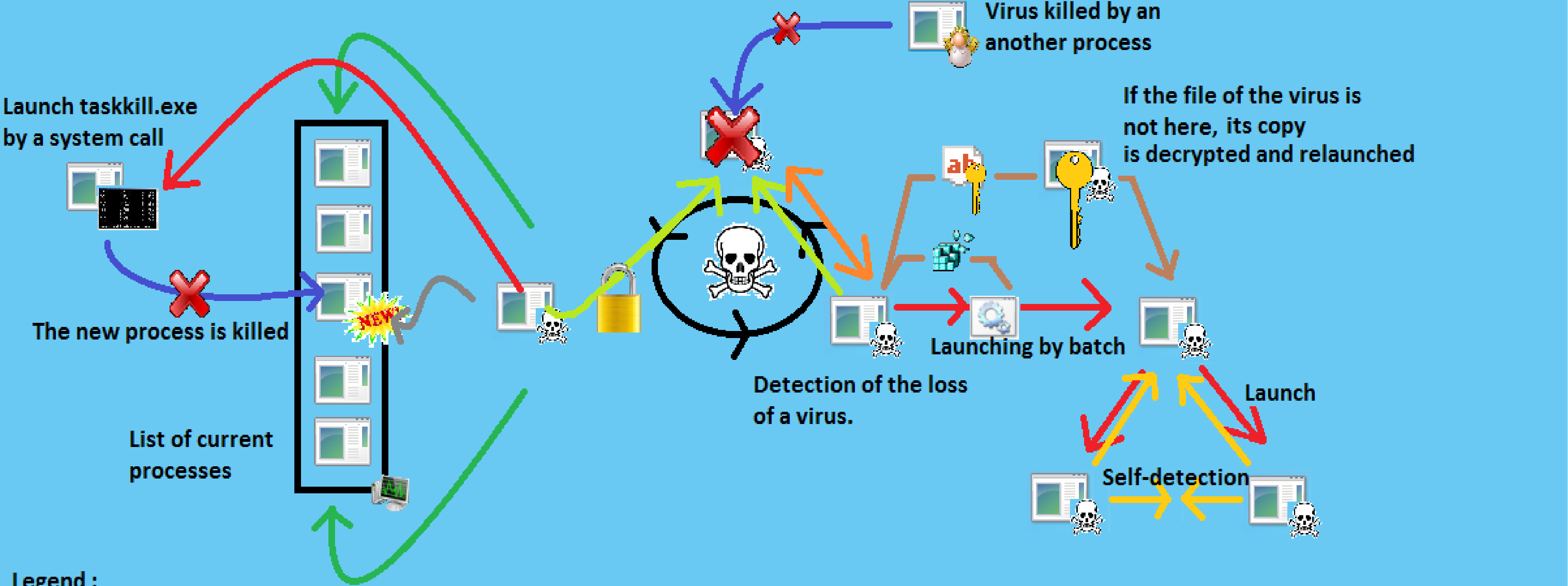
→ Simulation of the attack of a part of the virus

→ When the virus restarts, it launches two viral processes and kills all the processes that it can, with the exception of those that it has launched. The targets of new viruses include the parts of the previous virus which are deactivated through self-detection.



How works the virus in its final k-ary form :

How works the virus in its final k-ary form :



Virus killed by an another process

Launch taskkill.exe by a system call

The new process is killed

List of current processes

If the file of the virus is not here, its copy is decrypted and relaunched

Launching by batch

Detection of the loss of a virus.

Launch

Self-detection

Legend :

→ Launching processus

→ Detection of loss of processes

→ Reading information in the register

→ Auto detection of viruses

→ Checks the list of current processes

→ Kills the process

↔ Understanding about the authorization to restart the missed process

Circle of self-checking viruses to revive one which has been attacked

→ Detection of the arrival of a new process

Batch file to run other programs

Encrypted registry key

Virus file

Encrypted virus file

Security program (eg antivirus).

New process

Taskkill.exe

Important results :

Antivirus / Tests	Resistance against simple virus	Resistance to the establishment of a trigger	Resistance against the viral spread	Resistance against the virus in its final form
Avast	NO	NO	NO	NO
AVG	NO	NO	NO	NO
Avira	NO	NO	NO	YES
BitDefender	NO	NO	NO	NO
DrWeb	NO	NO	NO	NO
FSecure	NO	NO	NO	NO
GData	NO	NO	NO	NO
Kasperky	NO	NO	NO	NO
McAfee	NO	NO	NO	NO
Microsoft AV	NO	NO	NO	NO
NOD 32	NO	NO	NO	NO
Norton Symantec	NO	NO	NO	NO
Sophos	NO	NO	NO	NO
Safe N Sec	NO	NO	NO	NO
Trend Micro	NO	NO	NO	NO

Conclusions :

→ The use of the simple virus and a trigger is enough to crash a computer.

→ It's appalling that some antiviruses don't detect the creation of triggers and the spread of executable code from a binary file on the system.

→ We note that very few antiviruses verify effectively what may happen to the current processes over time. (An excessive and automatic killing of all processes by another process should rouse suspicions, shouldn't it ?)

→ If the virus is implemented for Windows, a similar implementation could probably be done with Linux or Macintosh.

Conclusions (2) :

→ Congratulations to the Microsoft antivirus : MSE. This antivirus can be easily killed using taskkill ! Knowing that the virus presented here kills all the processes that it can when it starts, MSE is one of the processes killed. In addition to not detecting the virus, the antivirus is neutralized by the virus... It's regrettable because Microsoft developers are the ones who are supposed to know all the ways of promoting safety in their processes (with system privileges, for example)... And they do not !

→ The problem is that Windows allows the user to watch and kill the current processes in memory. (Should Windows deprive the user of such tools when we know that they are vital when an application bugs ? Is it not the role of the antivirus to control and limit their use ?)

→ Too many antiviruses have been bypassed here by a very simple and very efficient virus. Is this really a good thing to trust only in the antiviruses for our security ?

Answer to the original question :

The original question was : **Do you still believe that nobody can crash a Win 7 system if there is a “powerful” antivirus deployed ?**

Yes it is possible !

→ Very easy to get codes online about windows API to make them viral.

→ Learning the C language on the internet is very easy nowadays.

→ The project has been completed in four months by a first year student !

→ A first-year student is even able to do it, so imagine someone more powerful...

What does this answer mean ?

→ Most antiviruses don't detect the attack, so they make no resistance to it.

→ Many antiviruses are not capable of withstanding of attacks made by someone who is not an expert. Soon anybody will be able to create and launch a virus.

→ Many antiviruses are not able to cope with an attack in real conditions.

→ Viral Weapons will be increasingly common in the future.

→ Testing our defense systems is crucial ! ⚠

Thank you very much for your attention.

If you have any questions, I would be happy to answer them...