

# VIRUS ANALYSIS 1

## Happy Gets Lucky?

Péter Ször  
Data Fellows

Virus writers want their creations to spread as far and as fast as possible. Several of them have attempted to use the Internet to achieve this goal. A few years ago they simply tried to spam infected files to newsgroups. While very few were really successful in this, some of them became infamous because of it, especially a French virus writer called Spanska. His early DOS creations even became in-the-wild viruses.

Spanska's strategy was much better than average when it came to spreading his viruses. He posted infected hack programs to certain newsgroups which could be used to register commercial products. Thousands of people look for such hacks on the Internet every day, infecting their computers with the attached virus. Unfortunately, this all happened in the past – the present is automation.

More and more viruses are written with built-in network features. Win32/Parvo (see *VB*, January 1999, p.7) was the first virus which could spam hoax messages with infected attachments by using socket communication. Parvo was not very successful due to certain limitations.

We knew Parvo was the first, but would not be the last, of its kind. During January 1999, Spanska's new creation,

Win32/Ska, became well-known by thousands of Internet users the world over.

It is not particularly easy to classify Ska. While most virus researchers agree that it is a worm, others, including myself, have some doubts about this. The fact is that Ska cannot be classified as a real virus and it is not a traditional worm either. The general consensus is that its working mechanism shows more similarities with worms.

Ska gets inside a computer via an email or newsgroup attachment, affecting those

machines that run the attachment. If an unauthorized attachment called HAPPY99.EXE is run, Ska puts up an attractive fireworks display, which could easily be mistaken for a good-looking accessory to the message. However, when the fireworks burst on-screen, Ska has already modified the WSOCK32.DLL file (if it was available for access) in order to monitor all postings that are made from the machine. All Internet access goes through APIs placed in the WSOCK32.DLL.

Afterwards, Ska spams the newsgroup or email recipient with copies of itself any time the user tries to send a message across cyberspace. Two messages are posted from the machine each time – the original mail and a copy of it with a UU-encoded attachment called HAPPY99.EXE which goes to the same location. This attachment is exactly 10,000 bytes long when decoded.

Ska shares many similarities with chain letters. Chain letters are classified as worms. When someone receives a chain letter with attachments (most often a script), the executed attachment can look for other email addresses and post itself to those places without modifying anything on the local machine. Ska works that way, with one important difference. It modifies a local DLL in order to get control later. It does this by modifying the WSOCK32.DLL file (it does not patch the DLL in memory). Therefore, rebooting the attacked PC will not remove Ska from the machine.

### Executing HAPPY99.EXE

In the first stages, a user receives two messages similar to the ones we have seen on [samples@datafellows.com](mailto:samples@datafellows.com):

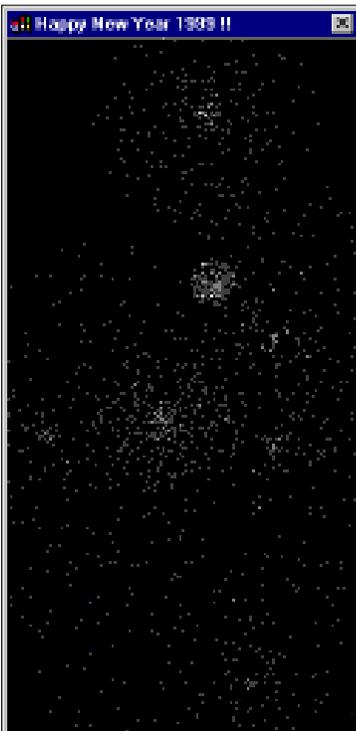
```
Date: Fri, 26 Feb 1999 09:11:40 +0100 (CET)
From: "XYZ" <xyz@xyz.cz>
Subject: VIRUS
X-Spanska: Yes
```

Note the 'X-Spanska: Yes' in the header. This is added by Win32/Ska. Everything beginning with X- is simply skipped by the mail servers. The body of the mail is a UU-encoded attachment.

The original message reads:

```
From: "XYZ" <xyz@xyz.cz>
To: <samples@datafellows.com>
Subject: VIRUS
Date: Fri, 26 Feb 1999 09:13:51 +0100
X-MSMail-Priority: Normal
X-MimeOLE: Produced By Microsoft MimeOLE
V4.72.3110.3
HAPPY NEW YEAR 1999.
I delete the attachment but it seems to still
block my computer.
What can I do. Thanks for your reply
```

When the user executes HAPPY99.EXE, Ska is activated.



Initially, it allocates memory for its own use. If that should fail, Ska will terminate immediately. After that, it checks if the system is Win32-compatible and terminates if it is not. Then it decrypts a short encrypted area of itself which contains the following text:

```
Is it a virus, a worm, a trojan? MOUT-MOUT
Hybrid (c) Spanska 1999.
```

After this, it gets the exact location of HAPPY99.EXE and copies it into the *Windows* system directory as SKA.EXE, unpacking an area into the previously allocated buffer. The unpacked code is used to create SKA.DLL (8192 bytes) in the *Windows* system directory. This DLL has two exported APIs – ‘mail’ and ‘news’, respectively.

Next, it makes a copy of WSOCK32.DLL, renaming it WSOCK32.SKA in the *Windows* system directory and tries to open WSOCK32.DLL for write. Ideally this is possible if the DLL file is not set to read-only and it is not in memory. Ska does not handle the read-only flag – setting the file to read-only can save WSOCK32.DLL from the patching. If the DLL is in memory, the file cannot be opened, and Ska tries to modify the registry field ..CurrentVersion\RunOnce to execute SKA.EXE from there during next boot (when WSOCK32.DLL will not be loaded). If WSOCK32.DLL is not in memory yet, then Ska tries to patch it.

It checks if WSOCK32.DLL is already patched by comparing the checksum field of the DOS EXE header to 7Ah. (If the checksum is not 7Ah it sets it to this value which is its self-recognition mark). Then it searches over the section table and makes the necessary modification. It sets the .text section to writeable and makes the virtual size of it a little bigger. This is in order to fit in there, in the section slack, with its short hook routines at the end of the .text section. Then it searches for two API names in the WSOCK32.DLL export sections. These are called ‘connect’ and ‘send’. It patches the export address entries of these APIs to point to new entries at the end of the .text section.

Next, it gets the address of a few APIs and saves pointers to the entry point of those functions which will be used later. Finally, it patches a very short routine at the end of the .text section in WSOCK32.DLL. Then the patch is ready. After this, Ska calls the message box animation routine, but only if HAPPY99.EXE was executed. If the SKA.EXE copy is executed from boot, the animation is not displayed.

### Posting a Message

When the patched WSOCK32.DLL is loaded in memory Ska intercepts the ‘connect’ and ‘send’ APIs. When the user makes a connection to anywhere, Ska checks the ports utilized. If it is mail or news port access Ska loads the SKA.DLL in memory. When a DLL is loaded, its initialization entry point is called. In this initialization routine another encrypted area is decrypted in the SKA.DLL which is related to the uu-encoding and a large enough buffer is allocated for future use. At that point, the original ‘connect’ entry point is called from WSOCK32.DLL.

When the intercepted ‘send’ is called, Ska re-checks if the sending is news- or mail-related by getting the addresses of ‘news’ or ‘mail’ APIs from SKA.DLL and calling them first. It copies part of the original email header to a new buffer, paying attention to ‘NEWSGROUPS:’, ‘MAIL FROM:’, ‘TO:’, ‘CC’ and ‘BCC’ and copying them for its own use. Finally, it adds the ‘X-Spanska: Yes’ string to the existing mail header.

It opens SKA.EXE and converts it into uu-encoded form in between the ‘begin 644 Happy99.exe’ and ‘end’ lines which marks the uu-encoded attachment. It posts this mail to the specified location. After this the original ‘send’ request is called which posts the original message without the HAPPY99.EXE attachment.

Ska logs the addresses where it could post itself. A text file called LISTE.SKA is created for that use in the *Windows* system directory. This text file will list a limited number of locations where Ska could post itself successfully.

### Conclusion

We are bound to see more and more viruses and worms capable of posting themselves all over the place. That means a new challenge for anti-virus researchers.

Very often we have to spend hours, days or a full week with some creations until we can provide a solution. Virus writers create something new again in a couple of months. The typical user wants a 24-hour solution. Always difficult, this is becoming even more challenging this year, especially when users still insist on executing uncertified attachments received from their ‘friends’.

Win32/Ska	
<b>Aliases:</b>	Happy99, SKA.A, I-Worm.Happy, MOUT-MOUT.
<b>Type:</b>	Win32 worm.
<b>Self-recognition in WSOCK32.DLL:</b>	Sets the checksum field of the DOS EXE header to 7Ah.
<b>Hex Pattern in WSOCK32.DLL and HAPPY99.EXE:</b>	3319 7508 8B44 2428 AA47 EB0A 3C77 751B 478B 4424 28AA E808 0000 0053 6B61 2E64
<b>Intercepts:</b>	Hooks Send and Connect APIs of WSOCK32.DLL.
<b>Payload:</b>	When executed as HAPPY99.EXE, it displays fireworks animation.
<b>Removal:</b>	Delete SKA.EXE and SKA.DLL and replace WSOCK32.DLL with WSOCK32.SKA in the <i>Windows</i> systems directory.