

VIRUS ANALYSIS 3

Bad IDEA

Péter Ször
Data Fellows

In many cases, backups are not sufficient to recover from virus infections, as they are often already infected. If the administrator makes a mistake creating a backup batch, flawed backups could be produced for months. When recovery from backup is needed, it is often too late to recognize that the actual information on it is wrong, and if it is good, it is still time-consuming to restore clean environments. Thus, disinfection remains, for many, the commonest method of recovery from virus infection.

Virus writers like neither detection nor disinfection, which is why hundreds of them have already used polymorphism. Naturally, a response to this was implemented in most good anti-virus products. A few years ago, we could have spent a couple of weeks writing a special algorithm to detect a new polymorphic virus. Nowadays, such viruses only require a few minutes' work with good scanning technology. Virus writers, however, were unlikely to abandon polymorphism without a rejoinder, testing the best anti-virus products against their ideas until they found some way around them. Unfortunately, some of them also release their creations by spreading them through Internet news groups. One such virus writer is Spanska (which is Swedish for 'Spanish').

His latest creation is IDEA.6155, a polymorphic virus with three layers of encryption. The first layer is decrypted by FSE (a mutation engine). The second decryptor does not contain its own key, but is decrypted using a 'brute-force attack'. Finally, the third decryptor uses IDEA (International Data Encryption Algorithm) – one of the most secure block algorithms available to the public. The virus contains the key for decrypting its IDEA layer, so this cannot be considered a security feature. Regardless, IDEA.6155 is a resident COM, EXE, and ZIP infector, and it is difficult to disinfect. While its size is variable, it still uses stealth, even under *Windows 95's* DIR command. It is the first known virus to correctly infect 'saved' COM programs (such as CHOICE.COM) in the *Windows 95* COMMAND directory. It also includes a new form of attack against *TbScan's* validation file. Unfortunately, despite its complexity, IDEA is quite workable. During testing, I was unable to find the usual fatal bugs which prevent the spread of such viruses.

Execution of an infected program

When the virus is executed, the FSE polymorphic engine decrypts the first layer byte by byte. This involves multiple methods such as XOR, ADD, ROR, DEC, etc, with 8-bit variable keys. FSE executes about 100,000 instructions to decrypt the first layer. Since the length of the decryptor

varies (in COM infections from 29 to 58 bytes, and from 31 to 60 bytes in EXE infections), the first byte of the encrypted layer is at a variable position. This makes detection of the virus a little more complicated.

When the first layer has been decrypted, the second decryptor takes control. This one starts testing all possible key combinations – a 'brute-force' cryptographic attack. There are four different methods to decrypt each word with four 8-bit keys – in order XOR1, ROR, SUB and finally XOR2 are tried. The maximum value of the key combinations is 16, 8, 32 and 64 respectively. The same 8-bit keys are used to decrypt the lowest and highest byte of each word. The function checks the decryption of three words at the beginning of the second encrypted layer. If the key is found, the virus uses it to decrypt the second layer, then passes control to the third decryptor. If none of the keys work, the virus terminates to DOS.

The possible key combinations for the second decryptor are restricted, otherwise decryption of an infected program could produce a noticeable execution delay. Testing the complete key space takes about five seconds on an *Intel 386* machine, as the 'attack' needs an average of three million instructions. The code of the third, IDEA-based decryptor was written by Fauzan Mirza. The assembly source of this IDEA implementation can be found at many locations on the Internet.

In 1990, the first incarnation of the IDEA cipher (then called PES – Proposed Encryption Standard) was released. Following cryptanalysis, the authors strengthened the cipher, calling the new algorithm IPES (Improved PES). The name was changed to IDEA in 1992. IDEA's current claim to fame is that it is part of *PGP*. It uses 64-bit blocks and a 128-bit key. The same algorithm is used for both encryption and decryption. There are no bit-level permutations, but IDEA mixes three algebraic groups: XOR, addition modulo 2^{16} , and multiplication modulo $2^{16}+1$ (this operation can be seen as IDEA's S-box).

Obviously the virus cannot use brute force against its IDEA encryption layer. It contains the variable key for decryption, but the location of that key can be at two different positions in the virus, which makes full decryption more complicated for anti-virus products. Following the IDEA decryption, the virus is completely decrypted.

Installation and Payload

When an infected program runs, the virus issues its 'Are you there?' call, Int 21h AX=6969h. It assumes it is already resident if AX returns unaltered. If the system time is 15:30, the virus calls its main activation routine, which displays the message 'Warning! strong crypto inside'. Before the text appears on the screen, the virus creates the

file C:\VIRUS.COM. This is a copy of the *EICAR* Standard Anti-virus Test file normally used to test the operation of those anti-virus products that support it. In other words, it is a harmless program detected by anti-virus products *as if* it were a virus.

Since IDEA.6155 is in an endless loop at this point, the PC has to be rebooted. Many users will then check their PCs with a scanner, and this 'low-level psychology' may thus work against them. When their anti-virus product detects (and deletes) the VIRUS.COM file, they will likely assume that the problem has been detected (and fixed), while in fact, the virus is still able to spread.

If IDEA is not already active, it allocates memory for itself, marking it as allocated by DOS in its MCB. After that, it hooks Int 21h, and uses anti-heuristic tricks when calling particular interrupts to avoid detection by heuristic analysers. Then it checks the time for the activation routine, and executes the host program.

Infection

Once active, the virus monitors various DOS Int 21h functions. Interestingly, IDEA is directory-stealthed under *Windows 95* too, paying attention to long filename directory functions. The stealth mechanism is disabled when some archivers are executed (such as PKZIP, ARJ, RAR). That routine also checks the extension of the files. If it is 'ZIP', the virus adds an infected README.COM into the archive. Three small demonstration programs are used as hosts. These were probably not written by IDEA's author and contain advertisements for WWW sites and BBS systems.

The virus infects COM and EXE files during execution, but avoids infecting COMMAND.COM and several well-known anti-virus programs. Its self-recognition mark is the value 69h ('i') at offset 03h in COM files or at offset 12h in EXE files. It checks whether the potential host is an EXE and infects appropriately. It does not infect COM programs shorter than 1000 bytes or longer than 57,000 bytes.

IDEA pays particular attention to 'saved' *Windows 95* programs that have 'ENUNS' protection. There are a few viruses (such as Memorial, see VB, September 1997, p.6 and December 1997, p.9) which avoid infecting these programs, but IDEA is the first known virus to infect them without any problem. It simply moves the 'ENUNS' string to the end of the infected file and patches the value of the word after it. It is much easier than it sounds. This protection is bypassed by adding the difference in size between the infected and original file to the original check value. So much for *Microsoft's* protection scheme.

The minimum virus size is 6155 bytes. This includes the shortest possible FSE decryptor, the actual virus, and a random value calculated from the original timestamp (up to 29 bytes). The polymorphic engine is called to create the first decryptor. The other encryption layers are then applied and finally the whole bundle is written to the host.

If infecting a COM file, the first four bytes, including the self-recognition mark, are modified in order to point to the new entry point. With EXE files, the infection marker is at offset 11h. IDEA modifies the CS:IP fields of the EXE header to point to its own code. IDEA handles file attributes correctly, infecting read only files properly. It also restores the program's original date-stamp.

Modification of ANTI-VIR.DAT

Several viruses avoid detection by integrity checkers by deleting their checksum files. However, IDEA avoids quick recognition by patching this file instead. This attack is targeted against *ThunderByte's* ANTI-VIR.DAT checksum files. As the names of checksummed files are not encrypted in this file, the virus easily patches the first character of its host's entry after infecting it. The integrity checker will not alarm on the file change, treating the (now infected) file as not having been checksummed. The next time the checksumming procedure runs it will create a new checksum entry, but for the infected program. This shows that it is generally better to encrypt such checksum files, although this may reduce the speed of the product somewhat.

Conclusion

On the DOS front, we can expect to see the appearance of more, and more complex, polymorphic viruses which are hard to detect and harder to disinfect. The bad news is that the day of the complex, workable polymorph seems to have arrived. This represents new challenges for anti-virus vendors in 1998, particularly those that implement thorough disinfection.

IDEA.6155

Alias:	Spanska.6155.
Type:	Resident, stealth, polymorphic.
Infection:	COM and EXE files. Adds an infected README.COM to ZIP files.
Self-recognition in Files:	69h ('i') at offset 03h in COM files and offset 12h in EXE files.
Hex Pattern in Files:	Not possible.
Hex Pattern in Memory:	5055 8BEC C746 0200 405D 58CD 21C3 B43C 33C9 CD21 C3B4
Payload:	Displays an animation at 15:30 each day an infected file is run. Causes <i>TbScan</i> to revalidate infected files.
Removal:	Under clean system conditions, restore infected files from backups or replace with originals.