ISSN 0956-9979 13666614 JUNE 2002

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Sentence construction:** May 2002 saw the sentencing of David L. Smith, three years after he confessed to having written and distributed the Melissa worm. With self-confessed bias and just a little leaning toward militancy, James Wolfe reveals just what this means to an unassuming virus researcher who happens to work in Corporate America. See p.2.

• **Blast from the past:** Peter Ferrie proves that old virus writers can learn new tricks. The appearance of Win32 virus W32/Chiton comes seven years after *VB* ran an analysis of MS-DOS virus Rainbow by the same virus writer. Details of how Chiton uses Thread Local Storage in *Windows NT, 2000* and *XP* can be found on p.4.

• **Over-eXPosure:** Another new platform for *VB*'s comparatives. This month's AV assault course is on *Windows XP* terrain. Matt Ham puts the products through their paces on p.16.

CONTENTS

COMMENT

Melissa Creator Vacationing at Club Fed

“ I would like some restitution for the week of my life that was wasted in response to David Smith’s exploration of his technological puberty. ”

Generally I don’t like to read the news in the morning since I haven’t quite had sufficient time to get the first sips of caffeine coursing through my veins. A few days ago, however, I couldn’t resist because I had heard that our old friend David ‘I’m not VicodinES’ Smith had finally been sentenced by the US Courts. I’m sure you all remember Davey-boy. He was the individual who gave us W97M/Melissa, which more or less ushered in the era of the mass-mailing virus. I’ve been asked by *Virus Bulletin* to provide a few comments on this little piece of news. Rather than going into the particulars of the verdict (which are well documented), I shall endeavour to give you my own biased (and probably somewhat militant) thoughts on just what this means to me, an unassuming virus researcher who happens to work in Corporate America.

First, let me go on record and say that I, like many, feel that Smith’s is a very light sentence. Considering the trouble that arose as a result of Melissa, I believe that he should have been given a much harsher sentence. Initially, I felt that he should have been given a *longer* sentence, but when you consider that people convicted of much worse crimes don’t get 20 months, I guess it isn’t too bad (although I would like some restitution for the week of my life that was wasted in response to David Smith’s exploration of his technological puberty). I have to admit that my first thoughts entailed Smith appearing before a firing squad, but I’ve since calmed down a little. I wonder if it would be too much to ask that they let him serve his sentence in Mississippi, where I believe they are still using chain gangs.

Monetarily, I think US$5000 is far too low. Both sides in the case agreed that damages caused by Melissa exceeded US$80 million, which means that Smith is being held responsible for only 0.006 per cent of the damage! At least make him pay the trial costs for goodness sake. I suppose I would have felt that justice had been done had he, in addition to the jail time and fine, been given a ridiculous amount of community service. Something in the neighbourhood of 10,000 hours would have been appropriate.

I like to lurk occasionally in some of the news groups to see what certain people are up to (especially when ex-*VB* editor Nick FitzGerald is throttling the script kiddies). On the day of Smith’s sentencing, I decided to review a few of the postings so I visited alt.comp.virus and two of the hacker groups. For the most part the reactions were as I expected. However, I was surprised as I read through some of the responses on one of the hacker news groups. Those whose postings I read seemed to feel that Smith probably did get off too lightly and that he was unlikely to serve any real time since he had already cooperated with the government for three years. Others thought that this was an exercise in futility by the US government and simply a waste of money. As I think about this, I have to say that I think these people probably have the best grip on the whole situation.

The last thing that I feel is worth mentioning is the difference between the way we (the United States) handle computer criminals and the way other countries do. It is nice to see that our courts have set a precedent for convicting virus writers. Too often people have got away with it on technicalities and other such nonsense.

It will be interesting if we ever figure out who wrote W32/Klez. Nick (FitzGerald) tells me that someone was shot (or hanged) last year for a minor hacking crime in China, which is where we suspect the author of W32/Klez lives. I suspect that they will give him/her the same treatment if he/she is found within their borders.

Ultimately, the effectiveness of a judicial system is measured by the deterrence of repeat crimes. How did this verdict do? Only time will tell, but I suspect that Melissa is destined to be remembered more as an anniversary of the day the computing community changed (and lost a lot of sleep) rather than by what happens to you if you break the law.

James Wolfe, independent researcher, USA

NEWS

Outraged of Slovakia

Recently *VB* received an email from an outraged user declaring that he had 'never read such nonsense' as that contained in a product review on the *CNET* website (http://www.cnet.com/). Intrigued, *VB* followed the link to a scathing review of *ESET*'s *NOD32*. This was something of a surprise since, in *VB*'s comparatives, *NOD32* has a pretty solid history of 100% detection of viruses in the wild. On closer inspection all became clear: *CNET*'s tests did not, in fact, include any virus samples. Instead, tester Ken Feinstein used virus simulation program Rosenthal Utilities.

This comes nearly two years after Joe Wells' open letter 'from the AV industry', which challenged *CNET*'s credibility for the same reason: the use of simulated viruses in product tests (as well as the creation of new live variants).

A little more rooting around the site revealed some heavily undisguised bias: *CNET*'s summary of every AV product listed (a total of ten, from eight different vendors) ends with a strong recommendation of *Norton AntiVirus* over the product in question. The more cynical among us might wonder exactly how much *Symantec* forked out for such staunch loyalty.

Clearly incensed, *ESET* has published on its own website a beautifully articulated tirade against the site and its testing procedures (http://www.nod32.com/news/cnet_zdnet.htm). *ESET*'s response makes for entertaining reading if only for the abundance of exclamation marks, underlining and bold italic text, which could not fail to convey their strength of feeling. *VB* concludes that, 20 months on, *CNET* is still 'pondering' the issues raised in Joe Wells' letter (see *VB*, November 2000, p.3) ▮

On-demand tests	Polymorphic	
	Number missed	%
CA Vet Rescue	106	97.00%
Command AntiVirus	164	93.01%
DialogueScience DrWeb	34	99.14%
Eset NOD32	89	97.75%
FRISK F-Prot	164	93.01%
GeCAD RAV	87	96.79%
Kaspersky KAV	35	99.08%
NAI VirusScan	48	98.78%
Norman Virus Control	151	93.76%
Sophos SWEEP	154	93.31%
VirusBuster VirusBuster	171	91.01%

Erratum

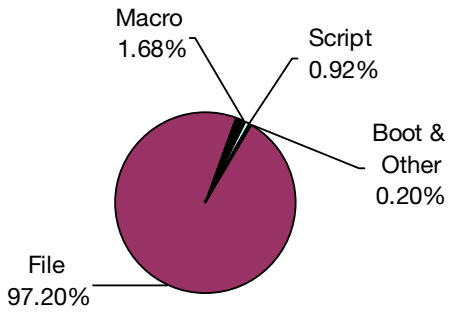
VB regrets that a number of errors appeared in *Virus Bulletin*'s recent comparative review on *Linux* (see *VB*, April 2002, p.17). The errors occurred in the table displaying the results of on-demand scanning tests and relate to the number of samples missed in the polymorphic test set. The correct figures are printed here. *VB* apologises for any confusion ▮

Prevalence Table – April 2002

Virus	Type	Incidents	Reports
Win32/Klez	File	5250	67.70%
Win32/Magistr	File	828	10.68%
Win32/BadTrans	File	473	6.10%
Win32/SirCam	File	257	3.31%
Win32/Hybris	File	136	1.75%
Win32/Fbound	File	131	1.69%
Win32/Mylife	File	127	1.64%
Win32/Nimda	File	94	1.21%
Win32/Gibe	File	80	1.03%
Haptime	Script	45	0.58%
Laroux	Macro	37	0.48%
Win32/MTX	File	33	0.43%
Win32/Aliz	File	23	0.30%
Win95/CIH	File	20	0.26%
Win32/Goner	File	19	0.25%
Marker	Macro	15	0.19%
VCX	Macro	12	0.15%
Kak	Script	11	0.14%
Divi	Macro	10	0.13%
Win32/Gokar	File	10	0.13%
LoveLetter	Script	9	0.12%
Win32/GOP	File	7	0.09%
Win32/Ska	File	7	0.09%
Others [1]		121	1.56%
Total		271	100%

[1] The Prevalence Table includes a total of 121 reports across 59 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

Distribution of virus types in reports



VIRUS ANALYSIS

Unexpected Resutls [*sic*]

Peter Ferrie
Symantec Security Response, Australia

In early 2000, while studying the latest release of the Portable Executable format documentation from *Microsoft*, I noticed the word 'callback' in a section describing data initialization. The section was called 'Thread Local Storage (TLS)'; in previous revisions of the documentation I had disregarded it, considering it uninteresting, but this time it had my full attention.

Where there are callbacks, there is executable code and where there is executable code, there may be viruses. However, it was a further two years before the appearance, in 2002, of the first virus that is aware of Thread Local Storage: W32/Chiton.

Blast from the Past

The virus writer's handle will be familiar to some. Calling himself 'roy g biv', perhaps from the colours of the rainbow, in 1993 he was the author of a virus that used the circular partition trick to make it difficult to boot from a floppy disk (by exploiting a bug that exists in MS-DOS v4.00-7.00 – see *VB*, September 1995, p.12).

It seems that, once again, roy g biv has created a piece of malware that may cause a few headaches for the anti-virus developers.

Modern History

All threads of a process share the address space and global variables of that process.

For applications that use a fixed number of unique threads, each thread can allocate memory and store the pointer in a separate global variable that the programmer has reserved for the purpose.

A problem exists for applications whose maximum thread count cannot be determined, or is unreasonably large, or those which execute multiple instances of a single thread. In these cases, there is no easy way to reserve a unique memory location for each of the threads, without the use of Thread Local Storage.

Thread Local Storage is a special storage class that is supported by *Windows NT*, *Windows 2000* and *Windows XP*. There are two types: dynamic and static.

Dynamic Thread Local Storage

Dynamic Thread Local Storage is used by applications containing threads whose size of local data is not constant, or which could not be determined when the application was compiled.

An application uses dynamic Thread Local Storage in the following way.

When a process is created, it allocates a Thread Local Storage index by calling the TLSAlloc() API. This index is stored in a global variable of the process.

Each thread that is created will allocate memory for its local data, then pass a pointer to this memory, and the process' Thread Local Storage index, to the TLSSetValue() API. At any time, a thread can retrieve the pointer to its local data by passing the process' Thread Local Storage index to the TLSGetValue() API.

The TLSGetValue() and TLSSetValue() functions access a table in the operating system's memory which is updated dynamically whenever a thread-switch occurs. This is how a single index can be used by all threads to access individual values.

Static Thread Local Storage

Static Thread Local Storage is used by applications in which all threads use a data block of the same size and contents (initially, at least). These data are described by a Thread Local Storage template.

In addition to the template, an array of callbacks can exist to provide customized initialization for each thread. Each of these callbacks is called before the process begins executing (the DLL_PROCESS_ATTACH event) and after the process stops executing (the DLL_PROCESS_DETACH event).

The callbacks are also called before a thread begins executing (the DLL_THREAD_ATTACH event) and after a thread stops executing (the DLL_THREAD_DETACH event), unless the DisableThreadLibraryCalls() API has been called first. In that case, only the process events will cause the callbacks to be called.

Since the callbacks are accessed via an array that is pointed to by a table, the address of which is stored in the tenth data directory in the Portable Executable header, this could be considered by anti-virus software (and researchers) to be an entry-point obscuring method (if they are not aware of Thread Local Storage data).

It is easy to understand why the virus author calls this technique 'the hidden entry point'.

Round and Round

The first time W32/Chiton is executed, it checks the event that caused its execution. The virus replicates only during

the DLL_PROCESS_DETACH event, which occurs when an application is exiting. The reason for this is that the virus uses the Structured Exception Handler list to gain access to KERNEL32.DLL. This file is not present in the list at an earlier time.

After gaining access to KERNEL32.DLL, the virus will retrieve the addresses of API functions that it requires, using the increasingly common CRC method to match the names.

Unlike the authors of some of the other viruses that use the CRC method, the author of W32/Chiton was aware that APIs must be stored in alphabetical order, so there is no need to search the CRC table repeatedly.

Additionally, the virus has support for both ANSI and Unicode functions merged into a single routine, and selects the set of APIs that is appropriate to the current platform (ANSI for *Windows 9x* and *ME*; Unicode for *Windows NT*, *2000* and *XP*).

The virus searches for files in the current directory and all subdirectories, using a linked-list instead of a recursive function. This is important from the point of view of the virus author, because the virus infects DLLs, whose stack size can be very small.

Filters

Files are examined for their potential to be infected, regardless of their suffix, and will be infected if they pass a very strict set of filters.

The first of these filters is the support for the System File Checker that exists in *Windows 98/ME/2000/XP*. The virus author was aware of the fact that the IsFileProtected() API requires a Unicode path, while directory searching on *Windows 9x* and *ME* require an ANSI path, so the virus transforms the path dynamically.

The remaining filters include the condition that the file being examined must be a character mode or GUI application for the Intel 386+ CPU, that the file must have no digital certificates, and that it must have no bytes outside of the image.

Touch and Go

When a file is found that meets the infection criteria, it will be infected. If relocation data exist at the end of the file, the virus will move the data to a larger offset in the file, and place its code in the gap that has been created. If there are no relocation data at the end of the file, the virus code will be placed here.

The infection will then proceed in one of two ways, depending on the file type.

For DLLs, the Thread Local Storage method is not used because a DLL will not call the TLS callbacks if the DLL is loaded using the LoadLibrary() API (and perhaps the virus author was concerned about the virus being labelled a WinNT virus, rather than a Win32 virus). Instead, the entry-point is altered to point directly to the virus code.

However, for EXE files, the Thread Local Storage method is used. The virus carries its own Thread Local Storage directory, which will be used should the target file contain no directory at all. The virus carries its own callback array for those hosts whose Thread Local Storage directory contains no callbacks.

When it encounters a host that already has a Thread Local Storage directory containing callbacks, the virus will save the address of the first callback and replace it with the address of the virus code.

Once the infection is complete, the virus will calculate a new file checksum, if one existed previously, before continuing to search for more files.

Once the file searching has finished, the virus will allow the application to exit by forcing an exception to occur. This technique appears a number of times in the virus code, and is an elegant way to reduce the code size, in addition to functioning as an effective anti-debugging method.

Since the virus has protected itself against errors by installing a Structured Exception Handler, the simulation of an error condition results in the execution of a common block of code to exit a routine. This avoids the need for separate handlers for successful and unsuccessful code completion.

Conclusion

It seems that some old dogs can learn new tricks. The author of W32/Chiton has moved successfully from the DOS platform to the Win32 platform, found a feature in the *Windows* Portable Executable file format that had (until now) been overlooked by anti-virus developers, and found a way to exploit it.

Additionally, the virus author distributed a document along with the virus source, which describes some further infection methods using Thread Local Storage. Interesting times lie ahead.

W32/Chiton	
Aliases:	W32/Shrug.
Type:	Direct-action parasitic appender/ inserter.
Infects:	*Windows* Portable Executable files.
Payload:	None.
Removal:	Delete infected files and restore them from backup.

TECHNICAL FEATURE

VB: Wearing the Inside Out

Richard Marko
ESET Software, Slovakia

With a user base of around five million, Visual Basic (VB) is probably the most widely used programming language in the world. Though not very suitable for 'hard-core' professionals, VB makes programming very simple and this is why it is so popular with beginners. Not surprisingly, we see a lot of worms, backdoors and other kinds of malware written in Visual Basic.

VB versions 5.0 and 6.0 are able to create two types of executable: p-code and native code (both require VB runtime DLL to execute). The majority of projects written in VB (including malware) are compiled to native code simply because it is the default option. However, it is very simple to change it to p-code, so there is good reason to study the internals of such executables. Understanding p-code will also help us to gain a better understanding of native code.

An article by Andy Nikishin and Mike Pavlyushchik, in the January 2002 issue of *Virus Bulletin* (see *VB*, January 2002, p.6) presents a good introduction to the internal structures of VB executables, especially those compiled to native code. In this article, executables compiled to p-code will be examined more closely.

p-code Basics

p-code, or *pseudo code*, is a set of stack-oriented CPU-independent instructions, an intermediate step between the high-level statements in Basic program and the low-level native code instructions executed by a computer's processor. It can be interpreted either by VB virtual machine (in case the project is compiled to p-code) or converted to a native code and optimized (in case the project is compiled to native code). It has instructions for loading, storing, initializing, object method calling, many instructions for arithmetic and logical operations, control flow and so on.

To avoid any misunderstandings in terminology, it should be noted that the p-code generated by Visual Basic for Application (VBA) differs from that generated by VB. While the p-code of VBA is basically the pre-processed source code stored in a special ready-to-interpret form, p-code of VB is a result of true compilation. It is, therefore, comparable to MSIL (Microsoft Intermediate Language) used by *Microsoft*'s *.NET* Framework.

p-code Internals

The internal structures of VB executables are not documented. Fortunately, however, debug information for VB runtime DLL is available. This way we can learn the names of p-code instructions and, generally, it makes the analysis easier.

To see how things really work we will first examine a simple 'Hello World!' project with one module and the following source code:

```
Sub Main()
MsgBox "Hello World!"
End Sub
```

We set 'Compile to P-Code' in project properties and compile it. After the PE executable is built we find a well-known stub at the entry point:

```
00401038 push   offset EXEPROJECTINFO
0040103D call   MSVBVM60:ThunRTMain
```

The ThunRTMain function exported by VB runtime DLL reads the EXEPROJECTINFO structure and starts VB project initialization. The structure contains a lot of important information, such as the address of the Main() function. The VB runtime will eventually call this address (if it is defined). Examining the code there we see:

```
004011CC mov    edx,00040175C
004011D1 mov    ecx,000401032
004011D6 jmp    ecx
…
00401032 jmp    MSVBVM60:ProcCallEngine
```

The code loads the address of a structure to edx and calls the p-code interpretation engine (ProcCallEngine) in the VB runtime library. The structure in edx (let us call it ProcDescription) contains all the important information about the Main() function, including the address of the corresponding module description structure, the size of local variables and the size of all p-code instructions for this function.

The ProcCallEngine loads various internal data, sets the error handler, allocates sufficient space on the stack and then starts processing the p-code instructions:

```
ProcCallEngine:
…
66104abe mov    ebx,edx
…
66104b7f movzx  esi,word ptr [ebx+08] ; p-code size
66104b83 neg    esi
66104b85 add    esi,ebx
…
66104b8d xor    eax,eax
66104b8f mov    al,[esi]
66104b91 inc    esi
66104b92 jmp    dword ptr [tblDispatch+eax*4]
```

From the code above we see the p-code instructions precede the ProcDescription structure directly. The first byte of

every instruction encodes the instruction type. This leads to 256 different encodings, though not all of these are used. Furthermore, opcodes 0xfb – 0xff, named Lead0 – Lead4, have a second opcode byte and thus constitute a new set of 256 encodings each. Consequently, there are hundreds of different p-code instructions. The opcode bytes may be followed by one or more arguments. The majority of the instructions have a fixed number of arguments but there are a few exceptions.

Knowing all that, we can inspect the p-code for our example Main() function itself:

```
0000 27FCFE      LitVar_Missing   loc_0104 ; Context
0003 271CFF      LitVar_Missing   loc_00E4 ; HelpFile
0006 273CFF      LitVar_Missing   loc_00C4 ; Title
0009 F500000000  LitI4            00000000 ; Buttons
000E 3A6CFF0000  LitVarStr        loc_0094,
                                  "Hello World!"
0013 4E5CFF      FStVarCopyObj    loc_00A4
0016 045CFF      FLdRfVar         loc_00A4 ; Prompt
0019 0A01001400  ImpAdCallFPR4    rtcMsgBox,14
001E 3608005CFF3CFF1C  FFreeVar
                 loc_00A4,loc_00C4,loc_00E4,loc_0104
0029 14          ExitProc
```

Let us explain this code briefly. We already mentioned that the VB p-code instructions are stack-oriented. The operation of an instruction can be derived from its name, but often it is necessary to inspect the actual code in the runtime DLL that interprets it.

The name of the first instruction, LitVar_Missing, can be divided into three parts. 'Lit' states that a value will be pushed onto the top of the stack. 'Var', which stands for 'Variant', not 'variable', tells us that the instruction works with a variable of the 'Variant' type (Variant is a special data type that can contain any standard kind of data). Finally, 'Missing' specifies that the variable will be loaded with a special DISP_E_PARAMNOTFOUND value.

The instruction takes one argument which is the offset of a local variable from the top of the stack. It fills it with the value and pushes its address onto the stack (the Variant type occupies more than four bytes in memory and therefore is not pushed directly onto the stack; instead a reference to it is added). LitI4 takes the I4 (Long) argument and, as expected, pushes it onto the stack also.

The second argument of the LitVarStr instruction is an index to a special table (ReferenceTable) which contains the addresses of various data such as strings, imported functions, COM objects and so on.

Every module in VB project has its own ReferenceTable. In the case of LitVarStr the corresponding address in this table points to a string in BSTR format (Unicode zero-terminated string preceded by its length). The string is contained in the read-only '.text' section of the PE executable, so a copy of it is created first by FStVarCopyObj. A reference to the copy is then pushed onto the stack by the FLdRfVar instruction as the first (Prompt) parameter of the MsgBox() function. We did not specify the remaining four parameters (Buttons, Title, HelpFile and Context), so they were generated automatically. The last three are loaded by LitVar_Missing and the Buttons parameter is vbOKOnly (defined as 0) by default.

The ImpAdCallFPR4 instruction takes two arguments. The first is an index to ReferenceTable, the second is the size of parameters on stack. The instruction takes an address from ReferenceTable and sends a call there. The code at this address follows:

```
00401020     jmp    MSVBVM60:rtcMsgBox
```

As can be seen, the code simply jumps to the routine statically imported from VB virtual machine DLL. After rtcMsgBox returns, the FFreeVar instruction checks all the Variant variables used and frees any allocated resources (in our example the copy of the 'Hello World!' string). Finally, ExitProc does all the necessary de-initializations, removes the error handler, restores the stack (removes all possible arguments) and returns to the caller of ProcCallEngine.

The example above has shown that, from the VB runtime point of view, there is not a big difference between a project compiled to native code and p-code. It can call p-code functions just like native code and the small chunks of code do all the necessary work to ensure the p-code will be interpreted correctly.

This example was very simple. Most VB projects have no Main() function; instead they have a starting form. Finding the entry points of the starting form events (for example 'Load()') is more complicated and requires knowledge of several data structures which can be traced from the EXEPROJECTINFO structure.

Imported Functions

Applications written in VB are able to call external procedures in DLLs. These can be imported either statically (during load time via the PE import section) or dynamically (during run time). Usually the procedures of the VB runtime library are imported statically. This includes procedures like ThunRTMain or ProcCallEngine, which are referenced automatically by VB compiler, as well as a special set of functions available to the VB programmer. They can be used in code without explicit declaration.

The names of these functions have the 'rtc' prefix (the prefix is added by the compiler, so it does not appear in the source code). Some of them are interesting in terms of generic malware detection – rtcCreateObject, rtcFileCopy and so on. The way these functions are called was illustrated in the previous example.

Dynamic Imports

Of more interest are calls to external API functions declared using the 'Declare' statement. The information about such functions is stored in a special table (ImportTable). Each entry contains the name of the function and DLL together

with a short piece of native code that invokes the function. The following example illustrates the use of external functions. A similar code can be found in many backdoors:

```
Public Declare Function GetCurrentProcessId _
  Lib "kernel32" () As Long
Public Declare Function RegisterServiceProcess _
  Lib "kernel32" (ByVal dwProcessID As Long, _
  ByVal dwType As Long) As Long

Public Sub MakeMeService()
Dim pid As Long
Dim regserv As Long

On Error Resume Next
pid = GetCurrentProcessId()
regserv = RegisterServiceProcess(pid, 1)
End Sub
```

This source code compiles to:

```
0000  0002        LargeBos           0002
0002  0005        LargeBos           0007
0004  4BFFFF      OnErrorGoto        Resume Next
0007  0011        LargeBos           0018
0009  5E00000000  ImpAdCallI4
                    kernel32:GetCurrentProcessId,00
000E  7170FF      FStI4              loc_0090
0011  3C          SetLastSystemError
0012  6C70FF      ILdRf              loc_0090
0015  7178FF      FStI4              loc_0088
0018  0019        LargeBos           0031
001A  F501000000  LitI4              00000001
001F  6C78FF      ILdRf              loc_0088
0022  5E01000800  ImpAdCallI4
                    kernel32:RegisterServiceProcess,08
0027  7170FF      FStI4              loc_0090
002A  3C          SetLastSystemError
002B  6C70FF      ILdRf              loc_0090
002E  7174FF      FStI4              loc_008C
0031  0000        LargeBos           0031
0033  14          ExitProc
```

Using the 'On Error Resume Next' statement is a little trick. It forces VB compiler to put the LargeBos instruction in front of the compilation of every statement, so we can easily see how a particular statement is compiled.

The loc_0088 is the pid variable and loc_008C is regserv. The way both API functions are called is very similar to the way the rtcMsgBox function was called in the previous example (although, now, both functions return Long value so it is reflected in the mnemonic code of the ImpAdCallI4 instruction). However, the associated code (remember its address is in ReferenceTable) is different:

```
0040134C     mov    eax,[004022E4]
00401351     or     eax,eax
00401353     je     00401357
00401355     jmp    eax
00401357     push   00401334
0040135C     mov    eax,00401020
00401361     call   eax
00401363     jmp    eax
...
00401020     jmp    MSVBVM60:DllFunctionCall
```

First the code checks whether the address of the imported function has been located already. If it has, it simply jumps there, otherwise the DllFunctionCall function is invoked with a pointer to the entry of ImportTable as an argument. DllFunctionCall reads the name of the imported function and DLL from there and tries to get the address using LoadLibraryA and GetProcAddress API. If successful it will save this address (to variable at 4022E4 in this case) and return it. If not, it invokes the error handler internally.

COM Technology

One of the reasons why VB is so popular must be the fact that it makes programming with COM objects so simple. Often a program that would take a lot of coding (and perhaps cause a lot of headaches) in languages like C++ can be written in a few lines of VB. Of course we pay a price for the comfort but sometimes it is very elegant.

In contrast to the VBA or VBScript members of the Visual Basic family, VB supports late binding as well as both types of early binding – vtable and DispID bindings.

Late Binding

While there is usually no need to use pricey late binding in VB, source codes of the infamous Melissa (VBA) and LoveLetter (VBScript) worms are so popular that the majority of VB email worms we see use this kind of binding. It means that no type library information about the COM object used is available during compilation so this must be collected during run time.

The example will show the way a few lines of code in VB are compiled to p-code. They use *MS Outlook* COM object to create an email. As we have selected no additional references in the VB project settings, nor have we explicitly defined the type of object variables, the compiler has no other option than to use the late binding. The source code:

```
...
Set out = CreateObject("Outlook.Application")
Set mail = out.CreateItem(0)
mail.Recipients.Add "marko@eset.sk"
...
```

The corresponding p-code:

```
...
0007  001A        LargeBos      0021
0009  F500000000  LitI4         00000000
000E  1B0000      LitStr        "Outlook.Application"
0011  046CFF      FLdRfVar      loc_0094
0014  0A01000C00  ImpAdCallFPR4 rtcCreateObject2,0C
0019  046CFF      FLdRfVar      loc_0094
001C  045CFF      FLdRfVar      loc_00A4
001F  FE4E        SetVarVarFunc
0021  0018        LargeBos      0039
0023  284CFF0000  LitVarI2      loc_00B4,0000
0028  25          PopAdLdVar
0029  045CFF      FLdRfVar      loc_00A4
002C  FF426CFF02000100 VarLateMemCallLdVar
                                 loc_0094,CreateItem,1
0034  042CFF      FLdRfVar      loc_00D4
0037  FE4E        SetVarVarFunc
```

```
0039  001C         LargeBos      0055
003B  3A4CFF0300   LitVarStr  loc_00B4,"marko@eset.sk"
0040  25           PopAdLdVar
0041  042CFF       FLdRfVar      loc_00D4
0044  FF3D1CFF0400 VarLateMemLdRfVar
                                 loc_00E4,Recipients
004A  FD9F         LdPrVar
004C  FE9805000100 LateMemCall   Add,1
0052  351CFF       FFree1Var     loc_00E4
...
```

All the communication with the COM object is realized via the IDispatch interface. The rtcCreateObject2 function converts input ProgID to CLSID, creates an instance of the specified class and requests a pointer to the IDispatch interface.

The methods and properties of this interface are then invoked by 'LateMem' instructions (i.e. their names contain the 'LateMem' string). All take a method/property name as an argument, convert it internally to DispID using IDispatch::GetIDsOfNames and then invoke the method/property using IDispatch::Invoke.

There are several 'LateMem' instructions. Methods are invoked by the 'LateMemCall' instructions, 'LateMemLd' instructions read properties and 'LateMemSt' write properties. If the instruction name is preceded by 'Var', the instruction takes the pointer to the class instance from the reference at the top of the stack. Otherwise, the pointer is read from a special internal variable set by the LdPrVar instruction (actually there are more 'Pr' instructions). On the other hand, if 'Var' is appended to the name, the instruction stores a result value in the specified variable. The 'LateMemCall' instructions also receive the number of arguments for the invoked method while the arguments themselves are prepared on the stack (PopAdLdVar).

VTable Binding

In the fastest form of early binding, vtable binding, Visual Basic uses an offset into a virtual function table (vtable). It needs to know the layout of vtable, IIDs of interfaces used etc., so appropriate references need to be selected in the project properties. The following code can be found in many worms, backdoors and other malware:

```
PathName = App.Path & "\" & App.EXEName & ".exe"
```

While the source code is pretty simple, the compiled code is a little more complicated and lengthy:

```
0000  0474FF       FLdRfVar      loc_008C
0003  0478FF       FLdRfVar      loc_0088
0006  050000       ImpAdLdRf     Global:VBGlobal
0009  240100       NewIfNullPr   Global:VBGlobal
000C  0D14000200   VCallHresult  VBGlobal:App
0011  0878FF       FLdPr         loc_0088
0014  0D50000300   VCallHresult  _App:Path
0019  6C74FF       ILdRf         loc_008C
001C  1B0400       LitStr        "\"
001F  2A           ConcatStr
0020  2368FF       FStStrNoPop   loc_0098
0023  046CFF       FLdRfVar      loc_0094
0026  0470FF       FLdRfVar      loc_0090
0029  050000       ImpAdLdRf     Global:VBGlobal
002C  240100       NewIfNullPr   Global:VBGlobal
002F  0D14000200   VCallHresult  VBGlobal:App
0034  0870FF       FLdPr         loc_0090
0037  0D58000300   VCallHresult  _App:EXEName
003C  6C6CFF       ILdRf         loc_0094
003F  2A           ConcatStr
0040  2364FF       FStStrNoPop   loc_009C
0043  1B0500       LitStr        ".exe"
0046  2A           ConcatStr
0047  4644FF       CVarStr
004A  FCF654FF     FStVar
...
```

In vtable binding methods and properties are invoked by the 'VCall' instructions. All these instructions take an offset to vtable as the first argument. They take the pointer to the class instance from the internal variable (see previous section) and invoke a method/property at the specified offset in vtable.

Since the return value is usually HRESULT the VCallHresult instruction is used most often. This is lucky indeed because this particular instruction takes a second argument. It is an index to ReferenceTable and points to the corresponding interface IID. The instruction verifies the returned HRESULT status internally and if it fails it invokes an error handler. The IID is supplied as an argument to the error handler. With both the vtable offset and the interface IID, it is relatively easy to look up a method/property name.

DispID Binding

The slower form of early binding is more similar to late binding and is seldom used in VB applications. The Automation COM components usually support dual interfaces (i.e. support both vtable and DispID binding), and VB compiler uses vtable binding whenever possible. There is a set of the 'LateId' instructions which are analogous to the 'LateMem' instructions, but they take the DispID argument instead of method/property name. There is probably no easy way to force VB to use DispID binding, but it is possible. For example, we modified the *MS Outlook* type library where we changed some of the dual interfaces to pure dispinterfaces. However, since this is clearly a 'laboratory' result we will not discuss it here.

Conclusion

The study of the internals of Visual Basic executables is a rather complex issue. It is also a time-consuming endeavour, since the analysis requires extensive work with debuggers, disassemblers, hex browsers and similar tools. Visual Basic, being one of the most popular programming languages in the world, may well justify all the effort. The outline of our approach indicates that time-consuming and detailed analysis may, in fact, provide important clues to the VB application's interaction with the outside environment and prove to be instrumental in the development of efficient heuristic rules to identify malicious software generically. The future will show how successful we will be.

FEATURE 1

System Disinfection

Jong Purisima and Vincent Tiu
TrendLabs, Trend Micro Inc., Philippines

The prevalence of malware that target systems as a whole, rather than the traditional 'files in a system', has risen in the past few years. We have seen numerous methods of exploitation, while some malware programs combine the traditional file infections with system infections that can result in catastrophic problems.

With these changes, additional technology has been implemented to provide better detection for all the components of contemporary malware. As new exploits and techniques are discovered, the ability to support detection of their malicious byproducts is all the more vital.

System Disinfection

As the battle for malware detection continues, a new one has emerged: system disinfection.

For a user, the problem with an infected system does not end with detection of the malware. Detection merely provides a reference point for restoring the system to its accepted functional state and, in general, more time will be invested in disinfecting it.

System disinfection can be defined as 'restoring a system to an acceptable and working state'. Of course, it is possible to carry out a 'full' restoration, however, the diversity of systems and system configurations makes this very difficult. The aim of disinfection is to enable the system to work as it did in its pre-infected state, without having to perform any re-installation.

Since *Windows* is currently the most commonly attacked system, it will be the focus of this article.

Detection of an Infected System

A very important part of system disinfection is the correct identification of an infected system. When a scanner detects a malware program that is supposedly a system infector, it does not necessarily follow that the system is already infected. Detection of such a program should, however, be used as a basis for carrying out further checks.

The opposite is also true: if a scanner has not detected any system-infecting malware, it does not necessarily mean that the system is not infected.

In order to understand the underlying concepts of disinfection, infection should be defined. Currently there are two types of system infection. The first are Memory-File-Registry infectors. These use any combination of the three (memory, file and Registry) in order to take control of a system. To check for infection of this type, a memory scanner, file scanner and a Registry scanner may be used.

The second type of system infector uses memory only – a good example is CodeRed. This type has no physical trace (in a file or in the Registry) and is a little harder to identify.

In a similar manner to its file counterparts, misidentifying a system as infected can cause problems, since attempting to clean a system that is already clean can lead to further damages – caution is advised.

Once a system has been identified as infected, the disinfection process can begin.

Memory

A critical aspect of system disinfection is removing the malware from memory. This is critical since the operating system will not allow the removal of files if a process or service is still resident in memory. Furthermore, removing the malware from memory removes the malware program's control over the system, thus preventing it, for example, from restoring its physical counterpart or re-infecting.

Memory residence can be classified as either a process or a service. Processes should be killed, while services should be stopped. To kill a process, we can use the KillProcess() APIs, while, to stop a service, we can use the SCManager() APIs.

Memory cleaning should start with a memory scan to identify where the malware resides. Once its location is identified, based on the malware analysis, you can define whether it is a process or a service and kill the process or stop the service accordingly.

It is good practice to re-scan the memory after supposedly removing the malware from memory. This is to make sure that the malicious program has not spawned another process or service before you have killed or stopped it – for example, the Bymer.A worm spawns multiple copies of itself in memory and then executes them one at a time.

Another technique for remaining resident in memory has been seen in Magistr.A, which attaches itself as a thread to the explorer.exe process. Yet another method is to exploit buffer overflow, as in the case of CodeRed which exploited the buffer overflow in IIS to reside in the same process as its victim. In both cases, the thread or the process should be terminated in memory to stop the malicious program and, if necessary, restart the original service. Remember that all processes and services started by the malware should be killed or stopped in order to ensure that the program is no longer resident in memory.

Registry

Another method of taking control of a system is to add or modify Registry entries. Malware programs often use this method together with a file which is run upon system startup as the result of adding a link to it in the AutoRun Registry folders.

Cleaning up the Registry entries should always be malware-specific. We must perform the exact reverse of the changes the malware has made in the Registry, which should be as simple as removing what was added or restoring what was modified. Removing and restoring Registry entries can be done using the Registry APIs.

Another method that malware programs use to make sure that they will be executed is the modification of entries inside the HKEY_CLASSES_ROOT folder in the Registry. Inside this folder, the malware can select any extension, which it uses to trigger itself when files with this extension are opened. A good example is the worm Yaha.A, which modifies the default Registry entry for .EXE files so that, whenever files with EXE extensions are executed, Yaha.A is executed first. Restoring the original default value in the Registry will clean this.

Some malicious programs try to modify the Registry entries with the intention not only of gaining control over systems, but also of compromising network security and rendering infected systems more vulnerable to Trojan attacks. An example is Nimda.A, which modifies the Registry to share all drives from C to Z as well as adding the user 'guest' to the 'administrators' group on *NT* systems. Again, restoring the default value on those entries that have been modified should clean up the system.

Other malicious programs attempt to modify the default home page, *Office* Macro Warnings, Internet browser's security settings and other miscellaneous settings that are contained within the Registry. Since restoring these to their previous value would be difficult, unless the old values have been recorded, restoring them to their default values should be sufficient to allow the system to function in its clean state.

Files

A more prevalent method in system infection is the use of a physical file, paired with a modification to the Registry. This modification results in the file being executed at startup or during any other special event. In either case, the 'dropped file' must be removed as part of the system cleanup.

Most AV scanners should be able to remove a dropped file by selecting 'delete' as the post-detection action. However, if the file is protected by memory residency or other special techniques, it may not be straightforward to accomplish such a task. This is why cleaning the memory and the Registry must be carried out first, removing any links which might prevent the file from being deleted or modified.

Some malware programs try to append themselves to system files in order to gain control of systems. Such is the case for MTX.A, which modifies the system file WSOCK32.DLL in order to utilize its mail routines. Again, most AV scanners should be able to clean the newly-infected file, but may have some problems with sharing violations. In this case, system disinfection would be the same as traditional file cleaning techniques, but must be executed once the link mentioned above has been broken.

The worm Ska.A, on the other hand, renames the existing WSOCK32.DLL to WSOCK32.SKA and then replaces WSOCK32.DLL with itself. In cases like this, deleting the Ska.A detected file (WSOCK32.DLL) is one part of the cleaning process, and restoring the original file's pre-infected name completes the process.

Of course, there are other file-related malware modifications. For example, modifying *Windows* configuration files such as WIN.INI, SYSTEM.INI and Autoexec.bat by adding a link to a malware file. In these cases, the system configuration files should be cleaned after the physical files they link to have been deleted.

Other system modifications, like planting backdoors and security policy compromises introduced by malware programs should be disinfected as well. FunLove, for example, will patch NTOSKRNL.EXE to disable the system's access rights checking and at the same time patches NTLDR, forcing it to skip the integrity checking of NTOSKRNL.EXE. These modifications must be reversed.

As with Registry cleaning, additions or changes to system configuration files should be deleted or restored to their default values (if the original values can no longer be found).

Inoculation

After an infected system has been cleaned, it would be advantageous to introduce some means of preventing re-infection. A good example is the case of FunLove, in which a dummy file or directory named 'FLCSS.EXE' can be added in the Windows System directory. FunLove will cease to infect any system that has this entry in the System folder of Windows. Another good example is to put a dummy file or folder named 'NOTWORM' in the C drive. This renders CodeRed.A dormant – its payload and replication mechanisms will not be triggered.

However, such preventive measures should be considered only during crisis and should not be used as a sole method of combating re-infection: future variants may not use the dummy files or folders that you have positioned.

However, this is a good opportunity to inform the user of exploit patches that can combat the re-infection of their systems. For example, using the patch provided in *Microsoft* Security Bulletin MS01-033 would block any IIS vulnerability exploited by CodeRed. Remember that the use

of security patches to prevent infection in the first place is a much better solution than the workarounds. Security patches should be used whenever possible.

Fix Tools

Standalone fix tools have become accepted solutions for system disinfection during alerts or outbreaks. Advantages of fix tools are that they are small (most would fit on a standard 1.44 MB floppy disk), can be downloaded quickly, and can be executed on their own.

A downside, however, is that the tools are malware-specific, which means that there is one tool for every malware program or family of malware programs.

Another major drawback of using this approach is that it can not be deployed from a central location, which means that if an enterprise of 10,000 infected machines is to be cleaned, you may need to deploy the fix tool on each of those machines individually.

Furthermore, since most of these fix tools do not have their own fully-developed scanners, false positives in identifying infected files or an infected memory may occur. As mentioned previously, cleaning an uninfected file or memory may result in even more catastrophic damages.

Other Problems

One current problem of system disinfection is that sometimes a system reboot is required to remove traces of the malware totally. This should cause little concern on desktop environments but may be a big issue when it comes to servers and production-related machines.

The reliability of disinfection should also be considered. Disinfection methods should be able to work in all *Windows* flavours, in all language versions and should not affect any other software installed on the system.

Conclusion

It would be safe to say that future system disinfection techniques will depend on the behaviour of future malware. However, improvements to current techniques can be made in the following areas:

- Reliability in identifying the infection.
- Minimizing server downtime during infection cleanup.
- Expandability and manageability of disinfection tools.
- Deployment of tools and centralization, within large-scale infections.
- Vulnerability assessment tools which provide information on possible malware entry points.
- Integration of these tools into AV products as an additional solution.

And the battle rages on …

FEATURE 2

CIRCA 2002: Austrian Incident Response

Joe Pichlmayr
Ikarus Software, Austria

[As businesses, governments and societies become increasingly dependent on computer systems and computer-based networks, governments across the world are showing an increasing interest in IT security. In the following features VB *looks at the approaches two European countries, Austria and Belgium, have adopted for nationwide incident reporting, handling, prevention and recognition. Ed.]*

Over recent years the Internet has begun to play an increasingly significant and sometimes vital role economically and socially. As a consequence, these areas have become exposed to attacks by malware.

Protection of the Internet infrastructure, businesses and their customers against computer viruses, flooding attacks, denial of service attacks, hacking and other destructive assaults is one of the major and most complex tasks of Internet infrastructure operators.

CIRCA

In order to address this issue in Austria, the Austrian association of Internet service providers (ISPA: Internet Service Providers Austria) set up the CIRCA Project (Computer Incident Response Coordination Austria) in cooperation with the Austrian Infrastructure Networks.

The main purpose of this project is to identify and disarm all types of attack on Internet infrastructures, ISPs and their customers as quickly and as reliably as possible. It is also intended to be a pro-active measure for security and defence. Furthermore, it is an attempt to establish a link to similar organizations both in Europe and in the global arena.

Web of Trust

As a physical means of communication, a 'web of trust' was set up to facilitate interaction between network and security officers of the ISPA members (almost all ISPs in Austria), associated companies or institutions and respective software producers. The 'web of trust' is coordinated using ISPA resources.

The network is the most vital element of the project, since a single Internet service provider cannot detect a large number of critical incidents, regardless of whether it is a commercial or non-commercial ISP. In almost every case, the malware that causes critical incidents (distributed denial

of services, computer viruses etc.) are discovered only once they have become widespread. Initially, several locally restricted incidents may pose no problem for the Internet; however, it is possible that these could change to become, collectively, a very significant and widespread threat.

In order to set up the network and to establish the necessary trust between all participants, regular meetings are organized, allowing face-to-face contact between the members, and a community has been established to assist the idea of a 'web of trust'.

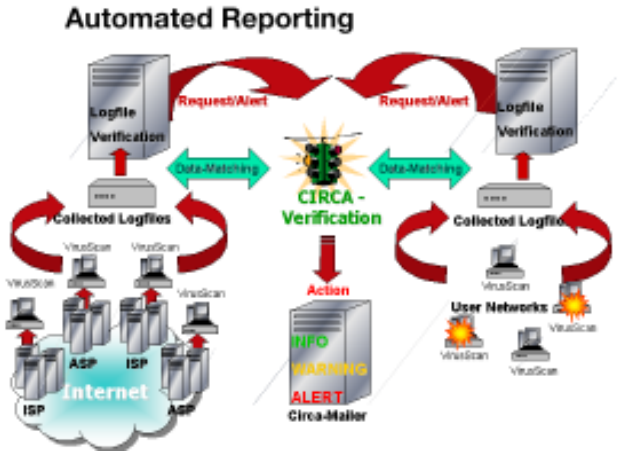
Incident Reporting

An incident reporting network is currently the most effective way to detect 'high-risk' virus outbreaks and other potentially dangerous incidents in their early stages. Timing is the most important factor for protective incident reporting. A detection and alerting system that is both fast and reliable may save a country millions of Euros.

One important component for a working solution is the successful integration of so-called 'galvanic frog legs'. These are sensors which are formed principally by the users (including corporate networks), as those who will feel the impact of the incidents directly, and by the providers (ISPs and ASPs) as those who are able to identify outbreaks by traffic anomalies and the use of a variety of scanner techniques.

For all parties involved in the project, CIRCA offers some very attractive benefits. From the individual user's point of view, centrally-activated virus protection nodes increase security because these will prevent many viruses and other malware from reaching their original destination. The centrally-based approach enables the corporate user to set individually scaled filters directly at the provider, placing him in the position to prevent threats before they are delivered to his networks.

Besides the original purpose of these measures, the provider is able to offer active security management, allowing him to diversify his portfolio of application service provision and improving his position in a very competitive market.



A central incident verification server sends alerts and advisories based on the analysis of results provided by scanning SMTP traffic and HTTP/FTP/NNTP. In addition, data from telecommunication providers is scanned in response to increasing concern regarding the possibility of malware for PDA and mobiles.

Practical Details

There are three levels involved in the incident reporting process.

Level One

The first level consists of the anti-virus (incident) scanner servers, which act directly at the providers' streams. These form the backbone of the incident detection system. In a database, the scanner servers register all virus alerts and incidents as well as traffic peaks (which may be indicative of a high-risk outbreak). In addition, the result of N-N scanning (the ratio of senders to recipients) is recorded.

The enormous amount of information generated at the first level requires an automatic central verification process. This takes place at the second level server.

Level Two

At the second level the collated logfiles from the first-level scanners are interpreted, determining whether incidents are caused by viruses, Trojans or worms and rating the level of the threats.

A process-based comprehensive body of rules filters the incident data. The better the network at the first level, the easier to verify automatically. However, the verification process cannot be fully automated since a manual rating is required, particularly in the case of providing further advice. The final decision (the result of virus/incident analysis) must be made manually in order to support the reliability of any third-level advisories.

The second level is carried out by a group of companies which have the resources to verify such threats (for example anti-virus or data security companies). The results of this level are passed on to the third-level server.

Level Three

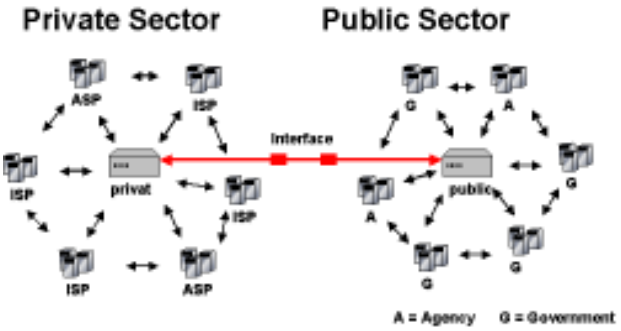
At the third level, the alerts produced at the second level are classified. Different forms of reaction may be triggered; from simple advice and recommendations sent directly to the end user and provider, to pro-active measures in the case of high-risk outbreaks (such as placing directly activated filters at the provider, which act as floodgates against highly distributed threats).

The second and third levels work in a very similar manner, the difference being that the second level does not have executive rights. The third level consists of the members of ISPA.

National Project Realization

Project CIRCA is carried out with the direct cooperation of the computer department of the Austrian Chancellery under Dr. Otto Hellwig as a part of Austria's measures within the framework of the eEurope2002 Action plans (see http://europa.eu.int/information_society/eeurope/action_plan/index_en.htm)

This project is a good example of private-public partnership. The advantage of the partnership lies in the close and profitable cooperation of private and public sectors on the one hand while, on the other hand, there is sufficiently clear-cut separation between the two sectors that the basic trust of each participant in the 'web of trust' is enhanced.



The computer security and communication nets have also become one of the main subjects of the EU Initiatives. The original security guideline of eEurope has developed to become a comprehensive guideline of network and information security (eEurope Benchmarking Report, 05.02.02).

The CIRCA project has been introduced within the framework of the EWIS initiatives (European Warning and Information System: http://ewis.jrc.it/). It has encountered overall acceptance as one of the first well-engineered and practical early warning systems of its kind in Europe, providing a service for both public and private infrastructure operators.

In a European task force with participation and/or involvement of FCB and ISPA, the possibility of the expansion of the CIRCA system in the direction of 'Computer Incident Response Coordination *Europe*' should be reinforced and evaluated.

Conclusion

Austria has tried a somewhat different approach to this problem by getting the ISPs (Internet Service Provider) and Infrastructure Networks involved in the solution and linking the process to the country's emergency services (points of contact across the country).

The system has yet to prove how effective it will be in helping to protect the country's critical information infrastructure in the event of a computer disaster or mass attack (e.g. spoofing, distributed attacks, etc.) – but the system is prepared!

FEATURE 3

Belgian E-Security: the Start of a European Initiative?

Eddy Willems
Data Alert International, Belgium

Despite my busy schedule of consultancy work, giving presentations, analysing viruses and dealing with *EICAR* matters (as EICAR Director Information and Press), another job was thrown at me a little over two years ago.

After the VBS/LoveLetter outbreak, the Belgian Government felt that something needed to be done to protect the Belgian people and businesses against such attacks. The Belgian Minister for Telecommunications, Rik Daems, reacted promptly to the virus, introducing immediate countermeasures. After a day of brainstorming he came up with the idea of creating a website on which anyone could find virus alerts and information about other security issues.

But this was only part of the plan. The goal was to alert the Belgian people before anything else was published on the Internet and *before* a new outbreak could begin. An attractive goal and an interesting new approach – that was the idea, anyway.

In order to run the website and its associated alarm system, the minister's cabinet decided to set up a security team of relevant experts. Thus, on 5 May 2000, the e-Security team of BIPT (Belgian Institute of Postal services and Telecommunications) was established.

e-Security Team

Despite the fact that Belgium does not boast a large number of anti-virus or IT security experts, the security team was formed after just two days. Within the e-Security team there are three groups: the minister's cabinet, external specialists and the advisory team, also called external analysts. The external specialists – a group of about 30 people – comprise individuals from ISPs, some TV stations, several large corporations and two security companies.

So, where are the security experts? My employer, *Data Alert International*, volunteered me as the only anti-virus expert within the advisory team. The other security company involved in the project elected a general security expert. Other members of the advisory team include a representative from the Federal Crime Unit (Federal Police)

and some members of BIPT itself. The advisory team consists of five people.

Alerting Procedure

There are eight steps involved in the alerting procedure:

1. An alarm may be initiated by members of the population (by contacting their ISP) or by the team of external specialists.
2. The call point of the e-Security team screens the alarm and determines whether it should be continued.
3. The staff of the Minister of Telecommunications are warned and the external analysts (the advisory team) are contacted by email, SMS or by telephone.
4. The members of the advisory team forward an analysis of the threat to the e-Security team, and a formal notification is drawn up.
5. The notification is forwarded to the staff of the Minister of Telecommunications.
6. The Minister of Telecommunications decides what action should be taken. If a press conference is appropriate, the staff of the Minister start the procedure.
7. Members of the e-Security team add information to the BIPT website: the problem, the dangers and what to do if your computer is infected.
8. The press and other media become involved. This includes the RDS system, in which programmes on the radio are interrupted for a virus announcement. Also, specially designed anti-virus banners may be added within minutes at some ISP portals (see below).

There is a timeframe of two hours for this list of actions. In my opinion, the most important part is that it is possible to use other media (i.e. radio and television) to warn the population of the potential problems. A national warning using such different kinds of media has an incredibly large impact. Can you imagine how it feels to be a system administrator driving to work and hearing between traffic alerts that 'AnnaKournikova' is in an outbreak situation?

Problems

Like every new system, a number of teething problems were encountered with the procedures.

For instance, at first, email was used to contact the advisory team and with that email was an attachment in the form of a *Word* document in .DOC format, containing details of the problematic new virus, worm, Trojan or hoax. Imagine how I felt when I saw the first email they sent to me! Had anyone in the chain not had adequate protection they could have sent a virus round within the email itself! Immediately, I asked that some other distribution method be used.

On receipt of the information the members of the advisory team must respond within one hour. If the process runs according to plan, an article can be published on the BIPT web page within that time, and a national security alert can be prepared – nice, if it works.

However, the details of the first message I saw were rather disappointing; it seemed that some members of the team could not tell the difference between a hoax and a virus. Neither could they distinguish whether something was of low or high importance. In fact, on one occasion I was just in time to prevent a warning being issued for a non-existent virus!

More recently, some of the alerts posted on the website have been superfluous and they vary, depending on who is responsible for editing the site at that time.

A number of links to a selection AV developer websites have been added to the site. Would it not be more helpful to point to all AV websites and select only genuine alerts? (I realise that the definition of a genuine alert is always a troublesome issue – which is something every anti-virus developer will appreciate.)

At those times when I am out of contact with the e-Security Team, the members tend to gather information about 'problematic' viruses direct from anti-virus developers' websites. The information is then consolidated and added to the official BIPT website even if it has not been verified. I am afraid I would say that, at such times, the system certainly does not act as an early warning.

Do not misunderstand me. I believe that the original concept of this early warning system has great potential – and the system has worked very well in the past. I have outlined the problems as I see them to other members of the team and improvements are being made already.

It occurs to me that the BIPT e-Security team is, in fact, more of a political game than it appears. Nevertheless, the idea behind the project should be honoured, because it is a responsive and a helpful one.

The question remains, however, could a new virus or worm with all the latest and perhaps unknown spreading techniques be quicker than this system? I think the answer is inevitable: yes, but the e-Security team can act within a defined timeframe to inform a very large number of pc-users about a potentially serious security problem.

There have been more than 100 alerts over the past two years; more than 60 of these were added to the website. Of those 60 about 20 were translated into the advisory procedure. About five alarms were given using the RDS radio system or some other notification system.

Over the Borders … the Euro Way?

The Belgian Government elaborated on their experience and chose to implement a permanent structure for an early

warning system. During the Belgian Presidency of the European Union, Belgium prioritized information and network security, resulting in a resolution which was adopted at the Telecommunication Council of 6 December 2001. This resolution includes detailed measures and initiatives to be conducted by both the member states and the European Commission.

But of course this is not enough. Belgium is not an island in cyberspace and viruses don't have a sense for geographical borders. It's clear that no truly effective warning system can exist without being global, without having connections with other countries and other continents.

To that end, in September 2001, the Belgian Minister for Telecommunications signed a Memorandum of Understanding with his Singaporean counterpart. This is a formal agreement on the sharing of knowhow and information between the countries and is the framework within which joint actions can be taken. Since Singapore lies in another time zone, a virus outbreak could hit there first, or vice versa. We hope that the people of Belgium will be prepared if such a situation arises.

Of course, involving the whole of Europe in such a project is easier said than done. A lot of political negotiation is necessary at this stage, which is why the Belgian Ministry has already started to make contact with closely related Ministries from countries like The Netherlands and Luxembourg.

The goal is to start as quickly as possible with an evolution of the e-Security Team within the BeNeLux countries to see whether it is possible to have a working solution across the borders. This could be the way to a new European e-Security platform; EICAR (European Institute for Computer Anti-Virus Research) has offered to provide advice and other input to such a system.

Of course, it may not be straightforward to realize an integrated European system like this – for example, will every 'expert' work on a voluntary basis? In Belgium it is done in this way, which is quite unique.

Bullet-Proof System

Despite my hectic schedule it seems that I always have several channels open so that I can be contacted when my help is really needed.

Oops! Another SMS message from BIPT arrived on my mobile phone as I was finishing this article. By coincidence (as part of one of my consultancy jobs) I'm writing this article very close to the area where shootings between the Palestinians and Israelis are going on. Even here, the system of the Belgian e-Security Team carries on working, just as long as my mobile phone keeps working …

The BIPT website can be found at http://www.bipt.be/, where security warnings are posted in Dutch and French, along with other virus descriptions and information.

COMPARATIVE REVIEW

Windows XP Professional

Matt Ham

It was with a certain degree of trepidation that I embarked upon this review: new hardware, a new platform and some new products to do combat with – and my anticipation of troubles was well founded.

The platform, *Windows XP Professional*, has an unhealthy obsession with attempting to contact the outside world, causing it to complain a few times in the process of testing. At first, boot-up of *XP* seems remarkably speedy, however the illusion is soon dispelled since, for several minutes, no network access is available, and on-access scanner components took up to five minutes to begin under normal circumstances. Under some circumstances, on-access functionality vanished or took literally hours to appear.

In addition to the new software, this month's Comparative saw an improvement in hardware. This has one major effect upon the results of the tests in that the results of past tests are no longer directly comparable with those produced from now on. Other than these (admittedly fundamental) changes, the testing procedure remained the same as it has been in the past. For details of the test regime please refer to past Comparative reviews. As ever, the results are specific to *VB*'s particular configuration and a product which proved impossible to test on our machines may show friendlier behaviour for other users.

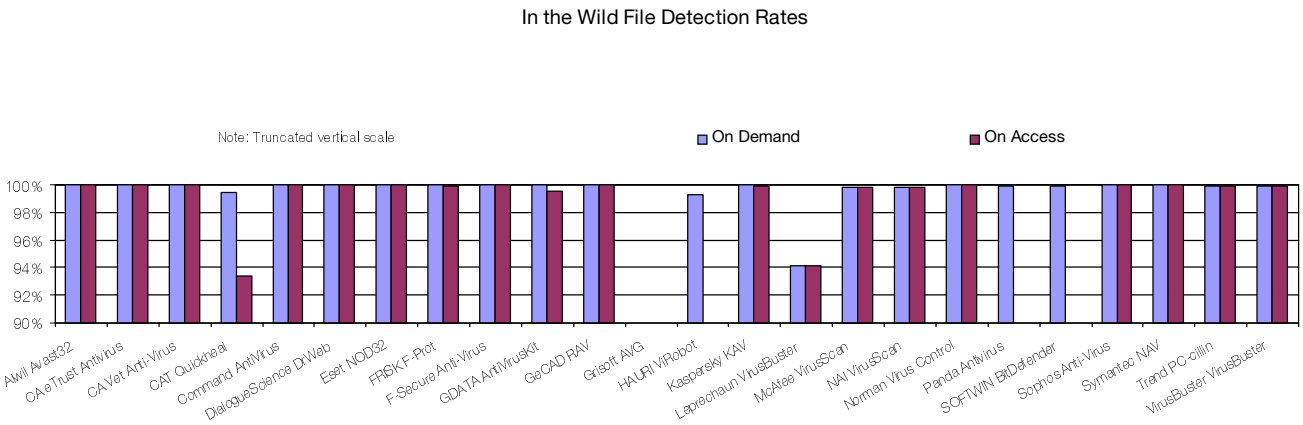
The Products

The line-up of products was the largest that has ever been assembled for a *Virus Bulletin* Comparative. Only one product proved totally untestable: *Ggreat*'s offering – a small downloader program which relied upon an Internet connection for the installation of its files. It was decided that giving test machines laden with viruses free access to the Internet was not an ideal plan. Of the remaining 24 products two were stated by their developers to have known bugs in the versions tested: *HAURI*'s *ViRobot* and *Grisoft*'s *AVG*. These were tested nonetheless. Of those products which were both new and testable there were offerings from *Leprechaun* and *CAT Computer Systems*.

Alwil Avast32 3.0.459.3

ItW Overall	100.00%	Macro	99.55%
ItW Overall (o/a)	100.00%	Standard	99.46%
ItW File	100.00%	Polymorphic	93.61%

Alwil's *AVAST32* is unfortunate in that it is the first product which will be described in less than awed tones. In what became something of a running theme, the on-access



scanner of *Avast32* did not perform as well as expected at first. This is a kind way of describing an on-access component which refused to start up, complaining of time-out errors. The answer to this problem lay in the traditional magic trick of rebooting the computer.

Once into its working state, *Avast32* scored full detection both on access and on demand in the Wild. Since no false positives were detected, *Alwil* gains the first VB100% of this review. As far as overall detection was concerned, results were good in all areas, with polymorphics being the only area where improvements might be hoped for.

CA eTrust Antivirus 6.0.96

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.87%
ItW File	100.00%	Polymorphic	99.94%

eTrust continued its tradition of being the product with the largest collection of mandatory patches. It was sometimes a little less than clear as to where the files involved should be placed, an area where improvement might be delivered.

However, *eTrust* put in another sterling performance, resulting in a VB100% award. One theme which became notable during the tests was the general increase in scan speed as produced by the combination of new hardware and operating system. Although the increase in machine specification would give an obvious boost to scan rates, the effect of the operating system is as yet difficult to gauge.

CA Vet Anti-Virus 10.4.7.0

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.81%
ItW File	100.00%	Polymorphic	97.00%

Vet has long been the only remaining product to be submitted to *VB*'s tests on floppies. This time, however, a CD was the medium of choice for *Vet*. This may be a sign of the increasing size of the products. Not including those three products supplied on CD, the size of the files submitted averaged over 30 MB per product. *Vet* still gains prizes for its small footprint in comparison, and walks away with a VB100% award for its detection performance. Weaknesses in detection were few, though the performance on the polymorphics is, again, an area where things could be improved.

CAT Quickheal 6.06

ItW Overall	99.45%	Macro	95.45%
ItW Overall (o/a)	93.35%	Standard	59.01%
ItW File	99.42%	Polymorphic	29.08%

CAT impressed from the start, having none of the technical difficulties so often associated with a new product. This was slightly tarnished by its inability to perform on-access boot sector scanning, despite the on-access file scanning being clearly operational. As far as detection was concerned, the results were decidedly mixed. When faced with a modern virus *Quickheal* is much more effective at detection than when faced with one of the older in the *VB* test sets. As an example, W32/CTX was detected in all of its samples in the test set, while older polymorphics were passed by without a mutter. Quite how disturbing this lack of detection is, will be very much a matter of opinion.

On the small niggles front, a couple of features surfaced. First, the performing of floppy scans was remarkably irksome, with the settings in need of constant adjustment. Secondly, the scanner would not delete infected archive files – despite detecting infections within these files.

Command AntiVirus 4.64.3

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.68%
ItW File	100.00%	Polymorphic	95.43%

Command AntiVirus is the fourth product to be worthy of a VB100% award. This is not without a measure of anger at the product due to the nature of its log files, which reported scanned files in a manner requiring some degree of

On-demand tests	ItW File		ItW Boot		ItW Overall	Macro		Polymorphic		Standard	
	Number missed	%	Number missed	%	%	Number missed	%	Number missed	%	Number missed	%
Alwil Avast32	0	100.00%	0	100.00%	100.00%	18	99.55%	116	93.61%	18	99.46%
CA eTrust Antivirus	0	100.00%	0	100.00%	100.00%	0	100.00%	1	99.94%	2	99.87%
CA Vet Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	73	97.00%	3	99.81%
CAT Quickheal	2	99.42%	0	100.00%	99.45%	181	95.45%	12408	29.08%	784	59.01%
Command AntiVirus	0	100.00%	0	100.00%	100.00%	0	100.00%	99	95.43%	6	99.68%
DialogueScience DrWeb	0	100.00%	0	100.00%	100.00%	34	99.20%	0	100.00%	1	99.98%
Eset NOD32	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
FRISK F-Prot	0	100.00%	0	100.00%	100.00%	0	100.00%	99	95.43%	10	99.65%
F-Secure Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	1	99.96%	3	99.85%
GDATA AntiVirusKit	0	100.00%	0	100.00%	100.00%	0	100.00%	1	99.96%	0	100.00%
GeCAD RAV	0	100.00%	0	100.00%	100.00%	0	100.00%	49	97.60%	5	99.73%
Grisoft AVG	115	86.06%	0	100.00%	86.91%	106	97.32%	242	86.05%	81	96.20%
HAURI ViRobot	1	99.75%	1	92.31%	99.30%	185	95.02%	10890	36.11%	628	67.55%
Kaspersky KAV	0	100.00%	0	100.00%	100.00%	0	100.00%	1	99.96%	0	100.00%
Leprechaun VirusBuster	32	93.72%	0	100.00%	94.10%	220	95.15%	1478	82.68%	131	92.10%
McAfee VirusScan	1	99.83%	0	100.00%	99.84%	0	100.00%	8	99.86%	3	99.85%
NAI VirusScan	1	99.83%	0	100.00%	99.84%	0	100.00%	8	99.86%	3	99.85%
Norman Virus Control	0	100.00%	0	100.00%	100.00%	0	100.00%	623	90.81%	30	98.65%
Panda Antivirus	1	99.92%	0	100.00%	99.92%	2	99.93%	1091	86.33%	22	99.39%
SOFTWIN BitDefender	1	99.92%	0	100.00%	99.92%	14	99.63%	128	90.93%	61	97.62%
Sophos Anti-Virus	0	100.00%	0	100.00%	100.00%	12	99.71%	64	95.54%	18	99.41%
Symantec NAV	0	100.00%	0	100.00%	100.00%	0	100.00%	14	99.76%	0	100.00%
Trend PC-cillin	1	99.95%	0	100.00%	99.95%	0	100.00%	263	93.32%	7	99.84%
VirusBuster VirusBuster	1	99.95%	0	100.00%	99.95%	0	100.00%	140	90.98%	10	99.73%

cajoling when extracting results. This and the log files of other products were the primary cause of reviewer rage in the analysis of this Comparative.

DialogueScience DrWeb 4.28

ItW Overall	100.00%	Macro	99.20%
ItW Overall (o/a)	100.00%	Standard	99.98%
ItW File	100.00%	Polymorphic	100.00%

DrWeb seems prone to producing strange results of late, and in this test managed to miss a selection of *Excel* viruses which the product has detected in all previous tests.

Whether this is due to some virus database bug or an overzealous trimming of heuristic triggers, will likely remain a secret known only by the *DrWeb* team. Other than this momentary flash of excitement, *DrWeb* performed in just the manner expected, notching up yet another VB100%.

Eset NOD32 1.256

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	100.00%
ItW File	100.00%	Polymorphic	100.00%

Once more comes the arduous task of noting details of *NOD32*, before declaring it to have gained yet another VB100% award. Speed and detection rate were maintained once more for *Eset*'s product leading to a predictable, but no doubt welcome, result for the Slovak team.

FRISK F-Prot 3.12

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	99.92%	Standard	99.65%
ItW File	100.00%	Polymorphic	95.43%

The second of the two purely *F-Prot*-based products in this month's line up, *FRISK*'s product showed both similarities to and differences from *Command F-Prot*. Similar were the overall detection rates and speed of scanning. The difference was that W32/Nimda.A samples were missed due to extension issues in the Wild on access. This was sufficient to deny *FRISK F-Prot* a VB100% award.

F-Secure Anti-Virus 5.40

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.85%
ItW File	100.00%	Polymorphic	99.96%

F-Secure Anti-Virus remained the slowest of the three products utilizing the *F-Prot* engine, no doubt due to its use of the *AVP* engine in parallel. This extra line of defence proved worthwhile, with the detection rate of the two engines combined being predictably higher than either component. This was sufficient to gain *FSAV* a VB100% award.

GDATA AntiVirusKit Professional 11.0.4

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	99.53%	Standard	100.00%
ItW File	100.00%	Polymorphic	99.96%

AntiVirusKit is another of those products offering two scanning engines in one package – in this case the *RAV* and *KAV*. Although *AVK* is not the speediest of scanners, it has not suffered too much by the additional lag that such a combination can produce. In this case the combination has also proved fruitful in detection rate – with file detection In the Wild being perfect. However this was spoiled by the product's failure to detect Michelangelo in the boot sector tests on access and a false positive in the clean set. With a performance so close to a VB100%, however, it seems likely that such an award is not far around the corner.

GeCAD RAV 8.5.8.0

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.73%
ItW File	100.00%	Polymorphic	97.60%

RAV's on-access scanner failed to operate at all when first installed. Reinstallation solved this problem, and the results were well worth waiting for. With full detection of In the Wild viruses and no false positives, *RAV* duly gained a VB100% award. Other results were solid, with the polymorphic set showing good signs of improvement.

Grisoft AVG

ItW Overall	86.91%	Macro	97.32%
ItW Overall (o/a)	86.91%	Standard	96.20%
ItW File	86.06%	Polymorphic	86.05%

In my last review of *AVG* I noted the longevity of the CD supplied – and the remarkable manner in which the updater for *AVG* had managed to cope with this antiquated base program. Alas, a piece of history vanished in this review, with this version of *AVG* causing a blue-screen and proving impossible to update. *Grisoft* provided a slightly less ancient copy of the base software which did not have these problems – but warned that the update software had serious, now corrected, bugs which would render the detection rates 'interesting'. This proved well founded, with results being far below those expected from the *AVG* product. Since, effectively, a crippled version was on test, comments on detection do not seem relevant to current performance.

HAURI ViRobot Expert 4.0 2002-05-07

ItW Overall	99.30%	Macro	95.02%
ItW Overall (o/a)	N/A	Standard	67.55%
ItW File	99.75%	Polymorphic	36.11%

The second of the self-declared crippled products, *HAURI* got off to a predictably unhappy start, as it locked up the machine when the on-access scanner was activated. This was corrected easily by removing the on-access component – a drastic but effective measure. *HAURI* state that current versions have been altered, and this problem does, indeed, seem to have been remedied in the products shipping currently.

The performance of *HAURI* on the last few tests caused considerable woe both for reviewer and developer, so it was pleasant to note significant improvement in detection rate. Detection was definitely in the acceptable range for In the Wild viruses, though narrowly missing complete detection. Akin to *Quickheal*, this improvement in detection rates has been applied with the seeming priority of more recent viruses over aged zoo specimens. As was noted for that product, the decision as to whether these files are worthy of detection lies with the individual user.

On-access tests	ItW File		ItW Boot		ItW Overall	Macro		Polymorphic		Standard	
	Number missed	%	Number missed	%	%	Number missed	%	Number missed	%	Number missed	%
Alwil Avast32	0	100.00%	0	100.00%	100.00%	18	99.55%	112	93.42%	18	99.48%
CA eTrust Antivirus	0	100.00%	0	100.00%	100.00%	0	100.00%	1	99.94%	2	99.87%
CA Vet Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	73	97.00%	5	99.62%
CAT Quickheal	2	99.42%	13	0.00%	93.35%	181	95.45%	12408	29.08%	931	46.09%
Command AntiVirus	0	100.00%	0	100.00%	100.00%	0	100.00%	99	95.43%	6	99.74%
DialogueScience DrWeb	0	100.00%	0	100.00%	100.00%	34	99.20%	0	100.00%	1	99.98%
Eset NOD32	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
FRISK F-Prot	1	99.92%	0	100.00%	99.92%	0	100.00%	99	95.43%	12	99.60%
F-Secure Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	1	99.96%	3	99.85%
GDATA AntiVirusKit	0	100.00%	1	92.31%	99.53%	0	100.00%	1	99.96%	0	100.00%
GeCAD RAV	0	100.00%	0	100.00%	100.00%	0	100.00%	49	97.60%	5	99.73%
Grisoft AVG	115	86.06%	0	100.00%	86.91%	106	97.32%	410	83.75%	81	96.20%
HAURI ViRobot	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Kaspersky KAV	3	99.88%	0	100.00%	99.88%	19	99.60%	1	99.96%	2	99.87%
Leprechaun VirusBuster	32	93.72%	0	100.00%	94.10%	220	95.15%	1478	82.68%	131	92.10%
McAfee VirusScan	1	99.83%	0	100.00%	99.84%	0	100.00%	8	99.86%	3	99.85%
NAI VirusScan	1	99.83%	0	100.00%	99.84%	0	100.00%	8	99.86%	3	99.85%
Norman Virus Control	0	100.00%	0	100.00%	100.00%	0	100.00%	623	90.81%	30	98.65%
Panda Antivirus	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SOFTWIN BitDefender	n/a	n/a	0	100.00%	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Sophos Anti-Virus	0	100.00%	0	100.00%	100.00%	12	99.71%	64	95.54%	18	99.41%
Symantec NAV	0	100.00%	0	100.00%	100.00%	0	100.00%	14	99.76%	0	100.00%
Trend PC-cillin	1	99.95%	0	100.00%	99.95%	0	100.00%	263	93.32%	7	99.84%
VirusBuster VirusBuster	1	99.95%	0	100.00%	99.95%	0	100.00%	140	90.98%	12	99.60%

Kaspersky Anti-Virus 4.0.50

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	99.88%	Standard	100.00%
ItW File	100.00%	Polymorphic	99.96%

With the arrival of the new hardware in the *VB* offices there came a new peril – the new machines came complete with sound cards and an audible, if tinny, internal speaker. When faced with viruses, *Kaspersky Anti-Virus* squeals like a pig. This had the effect of attracting a great deal of astonished attention, followed by a rapid clearance of the environs. Should this not be the desired effect, of course, the sound may be turned off as a program option.

As has so often been the case in the past *KAV* missed out on a VB100% award by the slimmest of margins – and, once more, through opting not to scan extensionless programs on access. Other than deliberate non-scanning, detection rates were impressive.

Leprechaun VirusBuster II

ItW Overall	94.10%	Macro	95.15%
ItW Overall (o/a)	94.10%	Standard	92.10%
ItW File	93.72%	Polymorphic	82.68%

This product, as its name suggests, is a rebadging of the *VirusBuster* product. *Leprechaun* has been a player in the Australian market for a considerable time now, yet it is the first time that I have reviewed the product. Unfortunately, first appearances did little to thrill. *VirusBuster* lacks that most vital of reviewer utilities – an obvious version number.

All functions operated as expected and scanning was easy, if a trifle inelegant. However, the detection rates were none too impressive and definitely worse than the performance of the other versions of *VirusBuster* tested in the past. With a plethora of misses In the Wild, *VirusButer* managed to rank bottom in this category for detection, despite competing against a product declared by its manufacturer to be defective. It also appeared not to have an option to scan inside archive files – rendering the archive scanning speed tests impossible. With evident potential in the engine, it remains to be seen whether this incarnation can fare better in future.

McAfee VirusScan 6.02.1019.1

ItW Overall	99.84%	Macro	100.00%
ItW Overall (o/a)	99.84%	Standard	99.85%
ItW File	99.83%	Polymorphic	99.86%

With a greed for possible VB100% awards, the Siamese twins that are *Network Associates* and *McAfee* submitted two products on this occasion. Unlike *CA*, whose products use different default engines, these two offerings use the same engine, differing in interface and functionality. Despite this it is possible to treat them as separate products, since history has shown that an interface can exercise all manner of influence upon the program lying behind it.

In fact, the results for the two programs were essentially identical, though missing out on a VB100% award as a result of missing a sample of W32/Gibe.A In the Wild. The *McAfee* product is the retail version of the software and, as such, the complexities of the program are remarkably hidden under the interface – lest casual users be scared away by the options which do exist.

NAI VirusScan 4.51

ItW Overall	99.84%	Macro	100.00%
ItW Overall (o/a)	99.84%	Standard	99.85%
ItW File	99.83%	Polymorphic	99.86%

Having summed up much of the two sister products in the preceding paragraph the matter of scanning speed remains. Most interesting is the comparative speed of *McAfee VirusScan* as opposed to *NAI VirusScan*. The two versions of the program offer almost identical throughput rates on both archived and unarchived files.

Norman Virus Control 5.3

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	98.65%
ItW File	100.00%	Polymorphic	90.81%

Norman Virus Control proved somewhat enigmatic in its on-access behaviour, which on first installing was most notable by its absence. Tweaking and cajoling had no effect, though on-access component error messages were generated, but performing the same actions upon an identical freshly re-imaged machine resulted in a fully functional on-access scanner.

In the past few reviews *NVC* has been castigated for its lack of a log facility and wondered at for the slow speed of scanning the clean set. There is now a log file facility, though the sluggish clean set scanning remains on the uncompressed executable set. This aside, *NVC* put in a good performance – good enough to result in a VB100% award. Distinct holes do remain on detection, however, with the polymorphic set continuing to give intriguing half detection in several of the sets.

Panda Antivirus Platinum 6.25.90

ItW Overall	99.92%	Macro	99.93%
ItW Overall (o/a)	N/A	Standard	99.39%
ItW File	99.92%	Polymorphic	86.33%

Panda Antivirus was another whose on-access component proved unpliable. Despite trying two versions of the software, the on-access component remained greyed-out permanently, with no amount of manipulation resulting in any on-access alerts or activity. In the remaining tests detection was by and large good, though there were weaknesses in the polymorphic set.

SOFTWIN BitDefender Professional 6.4.1

ItW Overall	99.92%	Macro	99.63%
ItW Overall (o/a)	N/A	Standard	97.62%
ItW File	99.92%	Polymorphic	90.93%

The subject of a recent standalone review, *BitDefender* behaved much as it did on its last inspection. In terms of on-access testing, however, Murphy Shield demanded a confirmation after every infected file had been detected. This was too tedious a key-press challenge and thus on-access file scanning was untested on this occasion. As far as detection was concerned, *BitDefender* was caught out In the Wild by samples of W32/Nimda.A which prevented it from achieving a clean sweep in that set. Elsewhere, the polymorphics were somewhat weak on detection and

Hard Disk Scan Rate	Executables			OLE Files			Zipped Executables		Zipped OLE Files	
	Time (s)	Throughput (MB/s)	FPs [susp]	Time(s)	Throughput (MB/s)	FPs [susp]	Time (s)	Throughput (MB/s)	Time(s)	Throughput (MB/s)
Alwil Avast32	93.0	5881.0		13.0	6102.6		63.0	2530.4	19.0	3926.7
CA eTrust Antivirus	89.0	6145.3		5.0	15866.8		44.0	3623.1	10.0	7460.7
CA Vet Anti-Virus	112.0	4883.3		6.0	13222.3		63.0	2530.4	12.0	6217.3
CAT Quickheal	125.0	4375.5	1	27.0	2938.3		50.0	10938.6	26.0	3051.3
Command AntiVirus	100.0	5469.3		4.0	19833.4		51.0	3125.8	6.0	12434.6
DialogueScience DrWeb	118.0	4635.0	[15]	8.0	9916.7		53.0	3007.9	9.0	8289.7
Eset NOD32	33.0	16573.7		3.0	26444.6		27.0	5904.3	7.0	10658.2
FRISK F-Prot	86.0	6359.7		4.0	19833.4		54.0	2952.2	6.0	12434.6
F-Secure Anti-Virus	204.0	2681.0		9.0	8814.9		124.0	1285.6	33.0	2260.8
GDATA AntiVirusKit	450.0	1215.4	1	23.0	3449.3		220.0	724.6	56.0	1332.3
GeCAD RAV	326.0	1677.7	[1]	11.0	7212.2		29.0	5497.1	14.0	5329.1
Grisoft AVG	171.0	3198.4		7.0	11333.4		68.0	2344.4	10.0	7460.7
HAURI ViRobot	32.0	17091.6	[1]	17.0	4666.7		40.0	3985.4	38.0	1963.4
Kaspersky KAV	154.0	3551.5		12.0	6611.1		85.0	1875.5	24.0	3108.6
Leprechaun VirusBuster	130.0	4207.2		40.0	1983.3	12	n/a	n/a	n/a	n/a
McAfee VirusScan	94.0	5818.4		5.0	15866.8		40.0	3985.4	9.0	8289.7
NAI VirusScan	100.0	5469.3		5.0	15866.8		42.0	3795.6	8.0	9325.9
Norman Virus Control	2222.0	246.1		4.0	19833.4		186.0	857.1	9.0	8289.7
Panda Antivirus	95.0	5757.2		6.0	13222.3		43.0	3707.4	7.0	10658.2
SOFTWIN BitDefender	701.0	780.2	4	8.0	9916.7		516.0	308.9	10.0	7460.7
Sophos Anti-Virus	64.0	8545.8		9.0	8814.9		36.0	4428.2	10.0	7460.7
Symantec NAV	152.0	3598.2		21.0	3777.8		87.0	1832.4	21.0	3552.7
Trend PC-cillin	67.0	8163.2		5.0	15866.8		51.0	3125.8	18.0	4144.9
VirusBuster VirusBuster	119.0	4596.1		9.0	8814.9		84.0	1897.8	14.0	5329.1

BitDefender had the dubious privilege of being one of the slowest of the scanners tested.

Sophos Anti-Virus 3.57

ItW Overall	100.00%	Macro	99.71%
ItW Overall (o/a)	100.00%	Standard	99.41%
ItW File	100.00%	Polymorphic	95.54%

Starting with the good news, the detection rates with which *Sophos Anti-Virus* has been blessed in the past are continuing to improve, the polymorphic set in particular showing improved detection in the trickier samples there. This, combined with the usual lack of false positives in the clean sets, resulted in a VB100% award for *Sophos Anti-Virus*.

On the irritating side, however, *SAV* has some of the least pleasant log files to deal with of those tested. Along with *Command AntiVirus* and the Hungarian version of *VirusBuster*, it converts file names in the logs to 8+3 format.

SAV also adds some compressed file details directly onto these file names. Not so relevant to testing, but unpleasant in the real world, the results for each sample in the test set

also span several lines. In comparison with the ease of use of the rest of the product, these irritations are magnified.

Symantec Norton Anti-Virus Corporate 7.61.935

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	100.00%
ItW File	100.00%	Polymorphic	99.76%

Having gained a string of VB100% awards recently, *Norton Anti-Virus* remained true to form, putting in a perfect detection rate for In the Wild files. Across the remaining test sets the results were similarly good, with only one miss in the usually recalcitrant polymorphic test set.

Trend PC-cillin 2000 7.61

ItW Overall	99.95%	Macro	100.00%
ItW Overall (o/a)	99.95%	Standard	99.84%
ItW File	99.95%	Polymorphic	93.32%

Having pulled in two VB100% awards in as many reviews, *Trend* brought forward a promising record of success. Unfortunately, this run was brought to a halt by the less than perfect detection of W32/CTX.A. The polymorphics as a whole remained *PC-cillin*-resistant to more than a comfortable extent. This was notable in both newer and older polymorphics despite relatively recent claims that polymorphic detection was in the process of improvement. Perhaps future reviews will bring new heights of polymorphic detection to brighten *Trend*'s corporate visage.

VirusBuster VirusBuster 3.009-14

ItW Overall	99.95%	Macro	100.00%
ItW Overall (o/a)	99.95%	Standard	99.73%
ItW File	99.95%	Polymorphic	90.98%

VirusBuster scored good rates of detection across the board. Unfortunately one spectre loomed to haunt *VirusBuster*: W32/CTX. This was detected on occasion, but not infallibly so, and patchy detection of this virus denied *VirusBuster* a VB100%. With the differences in detection seen between the two versions of *VirusBuster* it is a mystery quite where the differences lie. It is to be hoped that it is simply a case of an antiquated version having been supplied by *Leprechaun*.

Logs, Logs and More Logs

With close to 50 logs to deal with in the space of this review, the quality or otherwise of those provided became quite a pressing concern. Each of these logs must be deciphered into a form which provides the basic information – a virus was or was not detected in file x. Unfortunately, in many cases the developers who have designed these log files have added extraneous information, contorted the information present or simply rendered it all but impossible to interpret.

A prime example is the degree to which packed files are explained in some log files. Consider the case of a *Powerpoint* file infected with a virus in one of its several OLE streams. In the log file this may simply be presented as the file being infected, though this does not present all the information available. Some log files include a detailed breakdown of the contents. In the best case this leads to additional entries describing the subcomponents, yet clearly labelling the file itself as infected. In many cases, however, the logs declare the infected file to be clean, before embarking on several descriptions of areas which are infected – despite these areas being within the 'clean' file. When parsing log files this results in a file which is infected being declared clean, in addition to the appearance of infected objects which are not in the test set – hardly ideal.

Much less forgiveable are those log files which use reporting methods which, although easily readable to the human eye, are obscure as far as machine parsing is concerned. Such log files are most commonly of the form where multiple lines are used to report one scanning event.

Next we reach those log files which alter the scanned file names or paths. Log files which change the case of scanned files or path descriptions are a prime example.

Exactly the same horror greets those products which transform file names into 8+3 format. The platform here was *Windows XP*, which does not use 8+3 format under any circumstance. Despite this, three of the products reviewed converted file names in their listings to this ancient format. Not only does this make log file parsing difficult, it also makes it impossible to determine exactly which files have been declared infected.

Conclusion

While the detection rates in the products reviewed were variable, the trend in detection rate is upwards, if not at an outstanding rate.

The problems encountered in this test were more of a practical nature than a lack of detection. However, if a product does not load reliably or cannot be cajoled into loading at all, it is of little use. Customers tend to be somewhat put off if a product sits on their machine unable to perform as advertised.

Technical Details

Test environment: Three 1.6 GHz Intel Pentium 4 workstations with 512 MB RAM, 20 GB dual hard disks, DVD/CD-ROM and 3.5-inch floppy, all running *Windows XP Professional* Version 2002.

Virus test sets: Complete listings of the test sets used are at http://www.virusbtn.com/Comparatives/WinXP/2002/test_sets.html. A complete description of the results calculation protocol is at http://www.virusbtn.com/Comparatives/Win95/199801/protocol.html.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YP, England

Tel 01235 555139, International Tel +44 1235 555139
Fax 01235 531889, International Fax +44 1235 531889
Email: editorial@virusbtn.com
World Wide Web: http://www.virusbtn.com/

US subscriptions only:

VB, 6 Kimball Lane, Suite 400, Lynnfield, MA 01940, USA

Tel (781) 9731266, Fax (781) 9731267

END NOTES AND NEWS

The 11th Annual EICAR Conference and 3rd European Anti-Malware Forum takes place 8–11 June 2002 at the Forum Hotel, Berlin, Germany. See http://www.eicar.org/ for full programme or to register online.

Internet Security takes place as part of Internet World UK 2002, at Earls Court, London, UK, 11–13 June 2002. A series of seminars and case studies accompanies the exhibition. For information see http://www.internetworld.co.uk/london/.

The Black Hat Briefings and Training 2002 take place from 29 July to 1 August 2002 at the Caesar's Palace Hotel in Las Vegas, USA. The briefings will consist of eight separate tracks over two days (31 July to 1 August), with ten different classes on offer for training (29–30 July). For further details or to make an early reservation see http://www.blackhat.com/.

Information Security World Australasia 2002 will be held 19–21 August 2002 in Sydney, Australia. The conference and exhibition represent the region's largest dedicated IT security show. For full details see http://www.informationsecurityworld.com/.

The 9th International Computer Security Symposium, COSAC 2002, takes place 8–12 September 2002 at Killashee Hotel, County Kildare, Ireland. Cost of registration includes your choice of 40 symposium sessions, five full-day master classes, and the COSAC International Peer Group meeting, in addition to full-board accommodation and meals. Register at http://www.cosac.net/.

The 12th International Virus Bulletin Conference will take place at the Hyatt Regency, New Orleans, LA, USA from 26–27 September 2002. Take advantage of special *VB* subscriber rates and register now. Contact Bernadette Disborough; tel +44 1235 544034, or email VB2002@virusbtn.com. Visit the *VB* website for full programme details: http://www.virusbtn.com/.

Black Hat Asia 2002 takes place in Singapore, 1–4 October 2002. For further information see http://www.blackhat.com/.

Information Security Systems Europe 2002 will be held in Disneyland, Paris, from 2–4 October 2002. For more information visit http://www.isse.org/.

The Third Annual RSA Conference 2002, Europe is to take place 7–10 October 2002 at Le Palais des Congrès de Paris, France. As well as keynote presentations there will be more than 85 individual breakout sessions on topics ranging from enterprise security to hacking and intrusion forensics. See http://www.rsaconference.com/.

The CSI 29th Annual Computer Security Conference and Exhibition will be held 11–13 November 2002 in Chicago, IL, USA. The conference is aimed at anyone with responsibility for or interest in information and network security. For more information email csi@cmp.com or see http://www.gocsi.com/.

US private equity and buyout firm *TA Associates* has invested £41 million in *Sophos*. Following the firm's minority investment, two TA Associates senior executives have been appointed by *Sophos*, one to the *Sophos Plc* board in the UK and the other to the board of *Sophos Americas*. Founders and joint CEOs of *Sophos* Drs Peter Lammer and Jan Hruska retain overall control of the company. See http://www.sophos.com/.

The British Government is to be protected against virus threats by *MessageLabs*. In a deal worth over £1 million, *MessageLabs* will supply the GSI (Government Secure Intranet), which provides the secure network infrastructure for all Government departments, with its *SkyScan AV* managed anti-virus service. For more details of the contract see http://www.messagelabs.com/.

In a bid to increase market share in the European retail market, *Kaspersky Labs* has taken on business development company *MediaGold*. As part of the cooperation MediaGold has been assigned the role of *Kaspersky Labs*' representative and retail partner in Germany, Austria, Switzerland, Spain and the UK. Meanwhile, signs are that *Kaspersky* is poised to enter the US market. For more information see http://www.kaspersky.com/.

***GFI*'s Email Security Testing Zone has added three new tests**, enabling administrators to find out whether their networks are protected against malware using the Iframe Remote and Object Codebase exploits, and to check the efficiency of their AV software against the Eicar testfile. Administrators can register their names and email addresses on the *GFI* website and the tests will be emailed to them. See http://www.gfi.com/emailsecuritytest/.