

Self-Defending Security Software

John E. Kerivan, Ph.D.

nGran, LLC

9 Plain Rd. Westford, MA 01886

and

Kenneth Brothers, CISSP

CleanComputes

745 N. Pennsylvania Ave., Yardley, PA 19067-2019

Abstract - This paper describes a series of tests designed to attack security software in real time as it provides protection for applications and operating system programs on Microsoft Windows 2000 operating platforms. One security program tested fell into the Intrusion Prevention System (IPS) category of security software. A second security solution provided Anti-Viral protection and the third security program provided Anti-Malware protection for the test systems with a principal focus on Spyware and Adware detection and removal. All security programs were run in a variety of conditions including single mode, where only the security application was running through full integration modes where all security applications were running simultaneously. Security program default configurations were used in all tests. The findings indicate that none of the tested software was capable of defending itself against attacks designed to suspend and unload them from memory. As is shown, the IPS software was more robust than the other two solutions, but was easily compromised and actually created numerous false positives and misinformed the user on the running of the other security software.

I. INTRODUCTION

Self-Defense must be a basic requirement for any security software that runs on a Microsoft OS platform. Today's "Zero Day" trojans, worms and viruses have been designed to nullify anti-viral, anti-malware, host based firewall, intrusion detection and IPS software. Unfortunately, most major vendors of host-based security software are NOT comprehensively tested on their ability to defend themselves and thus their intended wards against improper termination and subsequent loss of proprietary data. In other words, if the security software has been stopped and/or unloaded from memory then it is safe to assume that the machine is unprotected.

This paper presents the results of the first in a series of evaluations of various security software solutions and their ability to defend themselves against such attacks and thus provide the required protection for the system in question. In this paper, we provide a set of initial

questions in the Survey Question section that "should" be asked of any vendor who claims to protect their security software from comprehensive termination attacks. Unsubstantiated marketing claims or "security scanning" performance data should not be considered proof of a self-defense concept. To that end, a suite of malware emulators was applied to the three vendor security applications under test. This malware was built to suspend and terminate security tools, infect, steal simulated proprietary information with currently accepted hacking methods and otherwise compromise the integrity of Microsoft OS platforms. The potential user of security software is urged to apply such methods when evaluating security vendor claims. Care was taken in the preparation of this paper to NOT release the vendor names.

The solutions tested were the latest released by the vendors. All of these security products are well along in their release cycles and have received favorable ratings from a variety of industry sources. As shown in the results section, none of the tested software was capable of defending itself against attacks designed to suspend and unload them from memory. As will be shown, the IPS software was more robust than the other two solutions, but was easily compromised and actually created numerous false positives and misinformed the user on the running of the other security software.

The anti-viral major process modules used a significant amount of machine memory, 32.3 MB of private memory to load and run. The anti-malware programs required 13.2 MB of private memory to load and run. Both anti-viral and anti-malware programs consumed roughly 80-95% of the CPU to run scans against local files. Such resource consumption renders a machine unusable for intended purposes while scanning is running. These test results indicate that vendors who use such methods should provide a less intrusive scanning capability in upcoming versions of their products.

Potential users of host based security products should in part, base their choice of a security solution on the ability of a product to resist process, thread and driver suspension / termination as well as the ability to unload

the suspended agents from memory and where applicable, the local Services Control Manager (SCM) database. The vendors of competing security applications are also urged to provide a more robust and better-defined set of default rules with their products to minimize false positives from competing products. It is felt that the anti-viral and anti-malware scanning products need to be redesigned to use significantly less memory and machine resources to provide intended scanning functionality. It would also be advisable that vendors consider adding self-defense capabilities to their respective products.

II. OBJECTIVES

The primary objective of this testing effort was to validate the efficacy and self-protection capabilities of the IPS tool. This host resident security software is a “behavioral” IPS tool that checks for program anomalies on a desktop by comparing running processes to a set of usage rules. The usage rules are controlled by default and user-defined policies. In theory, if a program violates one of the rules then its execution must be blocked.

This type of product makes bold assertions about its ability to detect and block security attacks in real time. These assertions are based on the assumption that the IPS tool cannot be stopped or unloaded from a properly configured desktop. This product is well along in its release cycles and has received favorable ratings from a variety of industry sources. However, none of those reviews have tested the ability of the IPS tool to protect itself from a suspension/termination attack. Thus, the goals of this testing were as follows:

1. To validate the strength and accuracy of a product’s anti-malware claims
2. To ensure that the security products could not be stopped or unloaded
3. To ensure that standard applications typically in use on most desktops are NOT adversely impacted by the use of the selected security application.
4. To ensure that false positives from the use of such a product are minimized.

III. METHODS

The IPS vendor was initially contacted and asked to respond to a series of technical questions regarding its choice of security methods to protect servers, desktops, laptops and network equipment from malware infection and potential compromise. A sample of one of the questionnaires is shown in the Survey Questions section of this paper. The response was to refer us to high-level marketing literature.

A series of malware emulation tests were constructed to test the strength of the security product. Brief descriptions of the “Top 50” are provided in the Appendix. The tests were constructed based on standard malware categories and included the following;

- Destructive Trojans – designed to erase hard drives, disable system BIOS, etc.
- Remote Administrative Trojans – designed to take control of victim hosts via network connections
- Keylogging Trojans & Spyware – designed to capture user logon credentials
- Information Stealing Spyware – designed to capture hard drive contents and ship via Internet connections to a third party site
- Multipartite Worms – designed to infiltrate a site, disable running security software and capture user data and logon credentials
- Polymorphic Malware – designed to be combinations of the above categories and capable of changing component identities, signatures and methods from one instantiation to the next

In the event that the security product was able to deflect an emulation of the Top 50, the actual Trojan, spyware or worm (when available) was launched in a secured test network totally disconnected from all other systems. Due to the capabilities of many of the Top 50 malware used for these tests all systems were re-imaged after the “real infections” were carried out. The only exception to this test methodology was in the use of the BIOS destructive Trojans. In this case a special tool was written to capture such an Interrupt from the Trojan and display it on screen rather than sacrifice a machine. However, it was also re-imaged after exposure.

Prior to the start of testing an isolated lab was configured to provide a representative set of conditions under which the security applications could be exercised. This included standard PC applications in “normal” use, such as Microsoft Office, Internet Explorer as well as other applications. Nine test systems were profiled prior to the start of testing to develop pre-test baselines. Due to the large volume of data collected, summary results are presented in this report. Integration testing results are described in the Compatibility Test section of this document.

All systems were monitored with Microsoft performance monitors as shown in the next section as well as another tool available from an independent third party vendor (www.sysinternals.com) called the Process Explorer v 8.35. The Process Explorer was an important tool to use for this type of testing as it could show the real-time process and thread execution of the Malware emulators. The CPU cycles consumed by each process tree were documented and are shown in the Results section. The Process Explorer showed the amount of Private Bytes

consumed by each running program. It also provided insights into the processes, threads and modules used by the security applications and their instantiations as they responded to Top 50 attacks.

A. Performance Monitor Parameters

Performance monitors were activated on all test systems to track various usage parameters. The monitors used on Win 2K machines were the Microsoft Management Console 1.2 version 5 (Perfmon) and were set to capture the following 9 plus other parameters:

1. Cache - % Data Map Hits: shows the % of data that were resolved without retrieving a page. This is used to check on whether RAM is sufficient on the system in question;
2. Logical Disk – Average Disk Queue Length: shows an inverse relationship between an increase in queue and a drop in hard drive performance (delays);
3. Memory – Pages per second: shows an inverse relationship between increase in paging and lack of sufficient main memory;
4. Network Interface – Total Bytes per second: shows TCP/IP network usage at the local system interface. This is correlated between all systems and network analyzer data;
5. Objects – Processes: shows the number of running processes during the measurement intervals associated with each scenario;
6. Physical Disk – Average Disk Queue Length: shows queue length combined with available physical memory;
7. Processor - % Processor Time: shows CPU usage by percent of available time. If this reaches 75% the processor needs to be upgraded to a higher speed;
8. System - % Total Processor Time: shows the total time usage over all processors (if available);
9. Server – Total Bytes per second: shows how busy the server is while processing system level I/O.

The nine usage parameters identified above were used to provide potential application tuning information and to show bottlenecks as a function of virtual user activities captured by the Process Explorer during the execution of all tests.

B. Injection Methods

Four basic Malware injection methods were used to bracket known infection techniques. They included the following:

1. Local Physical Access as an Interactive User, e.g. from a local disk or Malicious User;

2. Infection by email as well as an attachment, e.g. an embedded link vs. a script;
3. Clicking on an infected Web Link (see figure below);
4. Use of a Browser Helper Object to permit direct memory R/W capabilities, similar to comload or Warez group based attacks.

Within all malware injection methods, emulation programs, that did NOT require Registry writes or hard drive writes, were utilized. When a “Top 50” program used standard silent installations to infect a victim system, emulators used the same packing, obfuscation and unwinding methods (when available). MSI as well as INF based installations were attempted for “Top 50” programs using such methods.

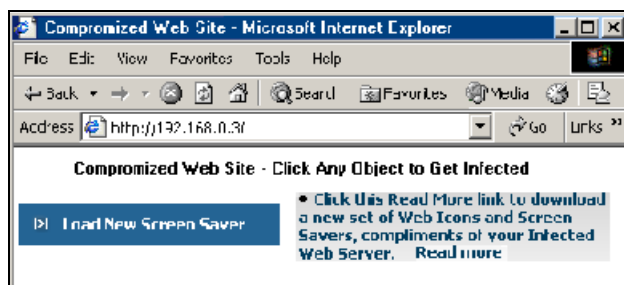


Figure 1: Sample of an Infected Web Page with Multiple Malware Options used during testing.

C. IPS Security Agent Configuration

The default IPS policies were used as the starting point for these tests. The vendor provides a set of mechanisms to perform policy-based configuration for the IPS, which is beyond the scope of this investigation. The general architecture of this model is available on the IPS vendor’s web site. The configuration software is implemented with the Policy server - IPS Access Control Server (ACS) and other third-party vendor policy servers that evaluate endpoint security credentials relayed from a network access device and determine the appropriate access policy to be applied (permit, deny, quarantine, or restrict). The configuration software is freely available from the IPS vendor.

The IPS vendor has policies defined that permit the products shown in the next Table to interoperate with the IPS. For those security vendors who do not participate in the bundling or interoperability program, the false alarms from the other security software were found to be exceedingly high. However, it should be noted that with the increased sophistication of Trojans and Spyware, most anti-viral software is stopped and unloaded at the very onset of an infection. See the next section for additional data on this topic.

TABLE I
Selected security applications that can run without interference from the IPS

IPS Vendor Configuration Agent works with:	IPS Agent Anti-Viral Vendor 1 Anti-Viral Vendor 2 Anti-Viral Vendor 3
IPS Vendor Configuration Agent is bundled with:	IPS Agent Anti-Viral Vendor 3

D. Program Suspension and Termination Techniques

One of the major goals of this study was to evaluate the ability of the IPS to protect itself from being stopped or unloaded from memory. Unfortunately, there are numerous programs and methods available to stop and unload programs while they are running in a Windows environment. Later versions of Bagel worms currently terminate and suspend over 520 security tools at the onset of an infection. The following eight techniques were embedded and used in the Trojan emulators:

1. TerminateProcess function call with SeDebugPrivilege using OpenProcess via kernel32.dll;
2. TerminateThread function call with SeDebugPrivilege using OpenProcess via kernel32.dll, by walking a Thread Tree;
3. ExitProcess function call with the start address in EIP register using CreateRemoteThread via kernel32.dll;
4. Attaching to a process as a debugger, using the DebugActiveProcess function in kernel32.dll;
5. Using the EndTask function in user32.dll;
6. Issuing WM_CLOSE messages to all windows in the target process via user32.dll;
7. Issuing SC_CLOSE messages to all windows in the target process via user32.dll;
8. SuspendThread function call using OpenThread via kernel32.dll.

Once a program, thread, driver or service had been suspended, the underlying service or driver was unloaded from the system using a readily available administrative tool on Windows platforms (sc.exe). This tool allows an administrator to call any of the service control API functions and vary any of the parameters from the command line. In effect this tool provides a convenient method to create and/or delete driver and service

information in the registry and the Service Control Manager database of a local host.

IV. RESULTS

There were 1250 tests conducted over a 3-week period on the systems described in the Methods section of this paper. This included time required prior to testing to setup the isolated test environment, build the Malware emulators, setup monitors and analysis systems.

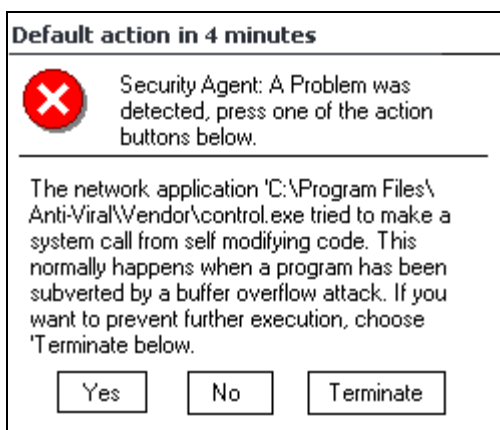
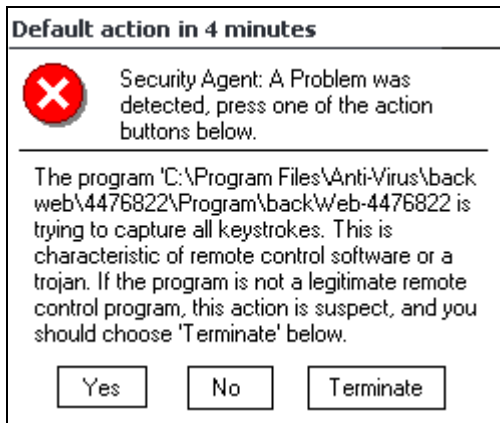
A. General Observations

The following observations are supported by the findings of this investigation:

1. Suspending, in most cases terminating, and then unloading their drivers and services easily compromised the IPS and Anti-Viral software.
2. The IPS blocked any attempts to terminate their processes and threads using the techniques described in the Methods section. However, they permitted the suspension of their processes, and their deletion from the Service Control Manager database.
3. When any infection started with IPS suspension and unloading techniques, the host was totally defenseless.
4. When left running, the IPS blocked 75% of the Top 50 and emulation test executions including those that used multi-step installation methods to infect a system.
5. On the positive side, the IPS consumed minimal resources on the test systems in default modes of operation.
6. However, the IPS was ineffective in thwarting the more sophisticated Top 50 attacks when run in "Monitor & Report" modes. Due to this finding, it is NOT recommended to deploy the IPS in anything less than its "Strict" operational mode.
7. The IPS was blinded by network based Denial of Service (DoS) attacks. However, it was adequate in responding to Browser based attacks that made use of Browser Helper Objects (BHO).
8. The IPS did NOT handle infected and redirected links properly. It could not distinguish between normal Web Server usage and the more insidious link exploitations used in the Trojan emulations (see the Stash data in the next section).
9. The IPS did NOT handle various information stealing Trojans properly. Native transport tools such as the Microsoft tftp client were used to offload data even while the IPS was running.
10. The IPS did NOT handle a variety of keyboard "hooking" mechanisms properly. Data was acquired and sent off-site while the IPS was

running.

11. The IPS had high false positive rates, especially when run in conjunction with anti-viral software from vendors that do not currently participate in the IPS vendor's NAC program. See the next two figures as examples of this finding.



Although the IPS reported on the programs as shown above, it misdiagnosed both conditions. In the case of the Anti-Viral software it did not set a hook to “capture all keystrokes”. It did run a dll to periodically check for new Anti-Viral updates. The IPS vendor should revisit some of its default rule sets if it cannot discriminate between a dll watching for signals indicating a new file update is arriving vs. setting a “keystroke logger”.

In the second case, the Anti-Malware program branched to acquire a new Anti-Malware update. It was not subverted by a “buffer overflow attack”. Again, we recommend that the IPS vendor revisit some of their default rules and add sufficient logic to eliminate such misleading directives, without forcing the user to make such policy corrections.

B. Specific Findings

Once properly configured, it was found that the IPS could be a reasonable host monitor (when left running) with a few significant flaws. The most significant flaw was that it did NOT properly protect itself from suspension and unloading using readily available hacker techniques to exploit a system. It failed to detect that all of its processes, threads, drivers and services had been suspended. It also failed to detect that its drivers and services had been deleted from the system.

Unfortunately, this product also created numerous false positives and was less than intuitive to setup and modify from a policy-based perspective. Under default operations, it blocked several network based attempts to infect a system and thus propagate through a local network. However, it did permit multi-step based attacks to execute and carry out various data stealing techniques. When apprised of these conditions the vendor referred us to company marketing material. The following observations characterize the performance of the IPS against the emulations and “Top 50” attacks:

1. Blocked standard IE browser memory insertion techniques;
2. Permitted multi-step download/installs from infected links;
3. Blocked all standard termination attempts (but did not log them);
4. Failed to block its own processes, threads, drivers and services from suspension, thus rendering the host defenseless;
5. Permitted Data Stealer installed programs to run through a download to disk and a separate instantiation of a remote silent setup function;
6. Permitted retrieval of sensitive data via use of native TFTP client software;
7. Blocked all email transmissions of data either with memory resident or native email transports when it was running;
8. Permitted the use of rundll32 with OpenAsApp based calls to shut down the system;
9. Blocked the execution of Keylogger programs designed to set System Hooks, use of DDE or CreateThreadProcess hook methods;
10. Failed to block and report on network based DoS or BIOS based attacks;
11. Missed specially crafted trojans that capture file, process data. See the next set of figures for an example of a Stash Trojan download and execution while the IPS is NOT reporting on its presence or activity. This event took place prior to the execution of the embedded “Zero Day” termination software in the Trojan. In other words, the IPS was still running.

C. Stash Test Results

As previously described, an emulation of the Stash Trojan was used to further exercise the IPS capabilities to defend itself against a Web based attack. This injection method made use of a custom HTTP service application that was downloaded by an interactive user who was misled by the opportunity to install a new set of screen savers.

To replicate these results a Web server must have the privilege to run and respond to "normal" external requests for services. Such a privilege typically exists in the default IPS policies. The first image was shown as the first figure in the Methods section as an example of an infected Web Page. The next figure shows the pre-test process table on the Web Server. Note that the Anti-virals as well as the IPS monitor were running at the time. This snapshot was taken with another Sysinternals tool called Handle version 2.2. Both the Process Explorer and the Handle tools produce snapshots of running processes and threads. Note that the Security processes are preceded by two asterisks(**).

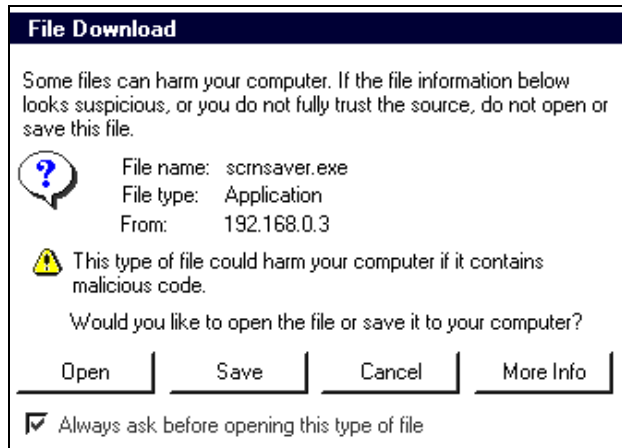
Handle v2.2 – Process Table Output

```
-----  
System pid: 8 NT  
smss.exe pid: 176 NT  
csrss.exe pid: 200 NT  
winlogon.exe pid: 196 NT  
services.exe pid: 248 NT  
lsass.exe pid: 260 NT  
svchost.exe pid: 448 NT  
spoolsv.exe pid: 480 NT  
**IPS Service1.exe pid: 488 NT  
**IPS Service2.exe pid: 502 NT  
Ati2evxx.exe pid: 508 NT  
**SERVIC~1.EXE pid: 520 NT  
svchost.exe pid: 536 NT  
**fsgk32st.exe pid: 552 NT  
**FSGK32.EXE pid: 576 NT  
**fsbwsys.exe pid: 588 NT  
**FSMA32.EXE pid: 620 NT  
**fssm32.exe pid: 636 NT  
**FSMB32.EXE pid: 668 NT  
MSTask.exe pid: 680 NT  
**FCH32.EXE pid: 860 NT  
**FAMEH32.EXE pid: 1016 NT  
**fsav32.exe pid: 1024 NT  
wanmpsvc.exe pid: 1044 NT  
svchost.exe pid: 1060 NT  
**fsdfwd.exe pid: 1416 NT
```

Handle v2.2 – Process Table Output

```
-----  
Explorer.EXE pid: 1480 SECURE1\jek  
**BackWeb-4476822 pid: 1560 SECURE1\jek  
Atiptaxx.exe pid: 1644 SECURE1\jek  
SynTPLpr.exe pid: 1652 SECURE1\jek  
SynTPEnh.exe pid: 1676 SECURE1\jek  
PRPCUL.exe pid: 1692 SECURE1\jek  
directcd.exe pid: 1704 SECURE1\jek  
**FSM32.EXE pid: 1720 SECURE1\jek  
**IPS Tray.exe pid: 1740 SECURE1\jek  
ntvdm.exe pid: 1280 SECURE1\jek  
WINFILE.EXE pid: 1584 SECURE1\jek  
procexp.exe pid: 1780 SECURE1\jek  
cmd.exe pid: 936 SECURE1\jek  
notepad.exe pid: 976 SECURE1\jek  
cmd.exe pid: 1796 SECURE1\jek  
handle.exe pid: 1928 SECURE1\jek
```

The next figure shows the standard download screen provided by IE when a file download has been requested. As a rule, end users should be discouraged from ever clicking the Open button.



Once the user clicks the Open button, the Trojan is delivered to a local directory, unwound, and then executes the operative functions. As shown in the previous section of this report, the IPS did NOT discriminate between a download of a suspicious package and a normal file download with its default rule set.

Once on the victim machine, the primary Trojan function eliminated all running Anti-viral and the IPS software by suspending the processes and threads, and where possible, terminating the processes and threads. The

Trojan finally unloaded any drivers or services components of the IPS and other Anti-viral software by invoking an admin tool (sc.exe) that deletes their entries from the Service Control Manager (SCM) database. Note the absence of running security programs in the next slide.

```

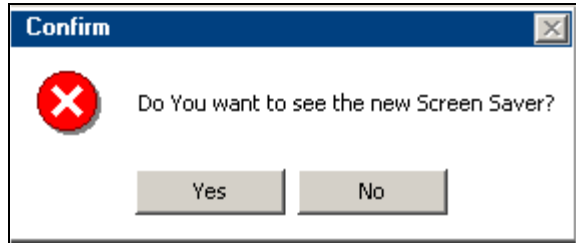
Handle v2.2 – Process Table Output
-----
System pid: 8 NT
smss.exe pid: 176 NT
csrss.exe pid: 200 NT
winlogon.exe pid: 196 NT
services.exe pid: 248 NT
lsass.exe pid: 260 NT
svchost.exe pid: 448 NT
spoolsv.exe pid: 480 NT
Ati2evxx.exe pid: 508 NT
svchost.exe pid: 536 NT
MSTask.exe pid: 680 NT
wanmpsvc.exe pid: 1044 NT
svchost.exe pid: 1060 NT
-----
Explorer.EXE pid: 1480 SECURE1\jek
Atpfax.exe pid: 1644 SECURE1\jek
SynTPLpr.exe pid: 1652 SECURE1\jek
SynTPEnh.exe pid: 1676 SECURE1\jek
PRPCUI.exe pid: 1692 SECURE1\jek
directcd.exe pid: 1704 SECURE1\jek
ntvdm.exe pid: 1280 SECURE1\jek
WINFILE.EXE pid: 1584 SECURE1\jek
procexp.exe pid: 1780 SECURE1\jek
cmd.exe pid: 936 SECURE1\jek
notepad.exe pid: 976 SECURE1\jek
cmd.exe pid: 1796 SECURE1\jek
handle.exe pid: 1928 SECURE1\jek

```

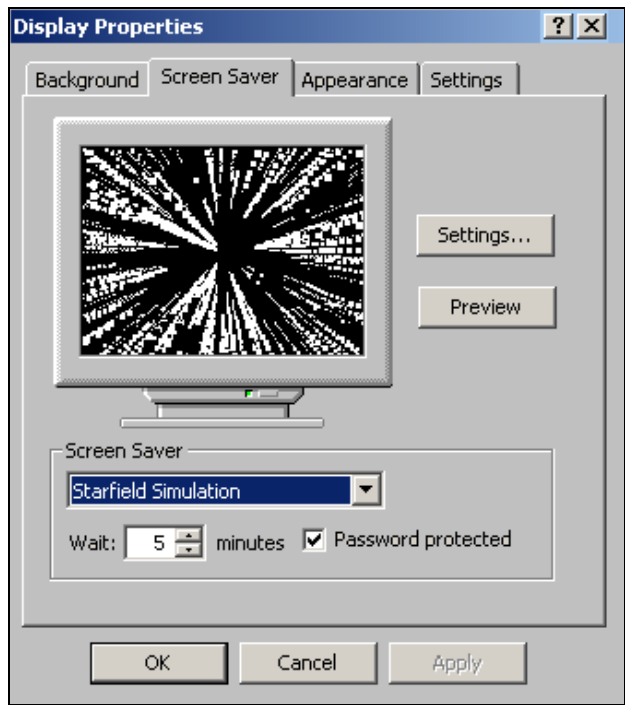
In addition, as the missing IPS drivers and services were properly unloaded from the SCM, upon reboot the user is given the first impression that all is well and a status function still shows a presence in the System tray. The IPS status program starts up but it does NOT report on the missing drivers and services that are no longer loaded and running. This condition should NEVER be permitted by any vendor to ensure that a user is not operating an infected system with a false sense of security.

While the unloading functions are running, a companion thread in the Trojan is free to execute any previously protected software on the victim machine. In the next figure, the user is presented with the option to Display the “New Screen Saver”. At this point, vital process and personal information have been captured and are being offloaded with an internal SMTP mailer via the browser

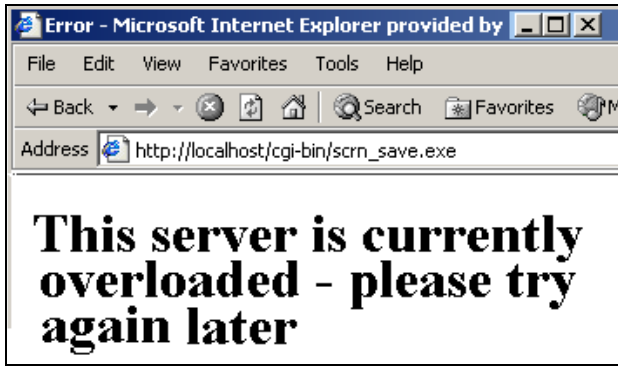
connection or via a local tftp client to an embedded tftp server off-site.



An inquisitive mind would naturally click ‘Yes’. Note that these displays are invoked locally using regsrv32. That program is normally restricted by the IPS which was disabled as the operative Trojan function began to run.



Finally, the last part of this example shows the end results from the browser perspective. Note that the victim machine has been destabilized and will require a reboot to be usable. The IPS functions that do NOT perform self-integrity checks have no driver to communicate with and as such are frequently locked in a non-usable state.



Also note in this example that the Trojan was added to the local machine and was running from a localhost instantiation. The IPS did nothing to block the infection with its default rules or stop the offsite transmission of proprietary information.

D. Compatibility Test Results

Before compatibility testing started, application profiles were collected for standard MS products in use at the site where the investigation was taking place. Baselines were collected for all measured products and are defined as opening the product in its most minimal modes and letting it sit idle for a minimum period of ten (10) minutes. Limited secondary tests were also performed where up to three (3) applications were opened and left to sit idle for the same period of time as in the baseline.

These tests focused on the amount of memory consumed by any given application and whether the application stayed within its Peak Working Set of Memory (PWS). Memory Leak Offenders (MLO) were those programs, as measured by the Process Explorer, that ramped PWS without releasing or freeing initial working sets of memory.

The monitor parameters of value were chosen based on their System Performance Impact (SPI), rated as Acceptable or Unacceptable. For the purposes of these tests, the SPI was also correlated to the running Performance Monitor data. Unacceptable performance ratings were given when the % Processor Time and Total CPU time for a given application was higher than 75% utilization.

Applications were also judged based on the amount of memory leak directly attributable to their process trees (includes parent process and all children threads and objects). The latter measure is important in judging the use of any given application over time. Due to the fact that some applications are opened and run idle for several hours each day, usage should be

modified for MLO to improve over all usability of a machine. In the event that a baseline application continuously exceeded its initial working set of memory, the application was judged to be undesirable for routine usage. The next 2 tables show the performance for the IPS Security Agent and the baseline applications. The first table shows standard mode IPS settings defined as basic logging. The second table shows the same baseline measurements with the IPS Agent in strict (kill & block) mode. Where a triple asterisk is present in the MLO column, the application was judged to be one where usage should be limited.

TABLE II
Baseline performance of selected applications with the IPS running in standard mode

Program	PWS	Tot CPU	Agent CPU	Agent Memory	MLO	SPI
Outlook	16 MB	3%	3%	14MB		A
MS-Word	16MB	3% - 9%	3%	14MB	***	U
	122MB					
Excel	5MB	2%	3%	14MB		A
Powerpoint	3MB	1%	3%	14MB		A
Iexplorer	12MB	6%	3%	14MB		A
MS Access	4MB	2- 3%	3%	14MB		A
Outlook Express	7MB	1- 2%	3%	14MB		A
MS Paint	2MB	1%	3%	14MB		A
Calculator	308KB	0- 1%	3%	14MB		A
Visio	19MB	3- 4%	3%	14MB		A
WordPad	4MB	1%	3%	14MB		A
TextPad	1MB	0- 1%	3%	14MB		A

TABLE III
Baseline performance of selected applications with the IPS running in strict mode

Program	PWS	Tot CPU	Agent CPU	Agent Memory	MLO	SPI
Outlook	18 MB	3%	3%	14MB		A
MS-Word	9MB	3%	3%	14MB		A
Excel	5MB	5 - 15%	3%	14MB		A
Iexplorer	12MB	5 - 90%	3%	14MB		U
Visio	19MB	15%	3%	14MB		A

The next table shows the performance of 5 applications while the IPS agent was running in strict mode and the applications were being exercised in a light load scenario.

TABLE IV

IPS running in strict mode with Programs running in light load e.g., 1 document, 1 web site, sending 1 message, creating 1 spreadsheet, creating 1 drawing

Program	PWS	Tot CPU	Agent CPU	Agent Memory	MLO	SPI
Outlook	16 MB	3%	1%	11MB		A
Anti-Virus	32 MB	28-95%				U
Anti-Malware	13 MB	5-85%				U
MS-Word	8MB - 122MB	3% - 9%	1%	11MB	***	U
Excel	5MB	2%	1%	11MB		A
Powerpoint	3MB	1%	1%	11MB		A
Iexplorer	12MB	6%	1%	11MB		A
MS Access	4MB	2-3%	1%	11MB		A
Outlook Express	7MB	1-2%	1%	11MB		A
MS Paint	2MB	1%	1%	11MB		A
Calculator	308KB	0-1%	1%	11MB		A
Visio	19MB	3-4%	1%	11MB		A
WordPad	4MB	1%	1%	11MB		A
TextPad	1MB	0-1%	1%	11MB		A

One interesting finding shown in this table is the dramatic change in MS-Word performance when an active session with one document is running. Contrast this with the baseline performance where the application was leaking memory at a rate of 6MB per minute in idle mode until the machine ran out of available memory and began to page.

Also note the performance differences between Internet Explorer in baseline (well behaving) mode and minimal mode with one URL connection. Like the baseline MS-Word condition, IE running in minimal mode burst between 5 and 100 % CPU utilization that directly correlated to scrolling through web pages.

The good news is that IE would free the memory. However, the CPU consumption led to intermittent machine slowdowns. In the event that a user opens IE, connects to a web site with high refresh and scroll rates and moves to another application, the machine will periodically pause for IE to refresh screen dump images even when running in the background. This indicates that users might want to consider opening the IE browser only when in direct use and closing it or moving to a more benign web site when not in use. From this data it can be seen that the IPS software is remarkably consistent in its use of memory and Total CPU utilization. It is estimated that the IPS software can run successfully with “normal” Company applications tested in a 768 MB memory footprint with a 1GHZ or greater processor speed.

V. RECOMMENDATIONS

Potential users of host based security products should, in part, base their choice of a security solution on the ability of a product to resist process, thread and driver suspension / termination as well as the ability to unload the suspended agents from memory and, where applicable, the local Services Control Manager (SCM) database. The vendors of competing security applications are also urged to provide a more robust and better-defined set of default rules with their products to minimize false positives from competing products. It is clear that the anti-viral and anti-malware scanning products need to be redesigned to use significantly less memory and machine resources to provide intended scanning functionality. It would also be advisable that vendors consider adding self-defense capabilities to their respective products, in that these types of attacks have been steadily increasing over the past two years.

VI. SURVEY QUESTIONS

1. How do you protect the IPS running processes and associated threads on a client?
2. How many processes does the IPS instantiate on each client?
3. How much main memory do the IPS processes take when fully loaded on a client? Include all modules, objects, components.
4. When sending data to central management servers do the clients use encryption? If so, what type of encryption?
5. What is the typical CPU cycle consumption when encryption is running on a client? How is that measured?
6. Do all processes on a client stay resident? If not, how many are running at most points in time?
7. Where is the main IPS Control Engine Process loaded in client memory? e.g., is it hidden, kernel side, user side.
8. Can the major IPS processes and threads be detected with the Task Manager or Process Viewer on a client? If not, where are they placed in memory?
10. Do any of the client side IPS functions monitor the process table for messages associated with any of the standard function calls? And if so, do they take actions when detected, and if so what actions? e.g., ToolHelp32Snapshot(), EnumProcesses(), EnumProcessModules(), DdeInitialize()
11. Are thread properties and thread states of processes evaluated by the IPS Control Engine? If so, is privileged time for threads monitored?
12. Are Window Classes monitored on clients? If so, are any signatures or behaviors assumed for one class or another?

13. Are Message Groups of running processes on a client evaluated? If so, do any groups receive precedence over others?, e.g. DDE, MDI, AFX/MFC, IP Address, WM_USER e.g., are some classes privileged, if so what are the evaluation criteria/
14. Does the IPS software sweep network connections for any listening agents?
15. How does the IPS Control Engine distinguish between normal debuggers and potential hacking processes? For example, DDESpy or Spy++ vs ProcKill or PWSHooker?
16. Do the IPS programs fail normally open or normally closed?
17. How does the IPS evaluate un-handled OS exceptions?
18. What programming languages and development environments are used to build the IPS programs? Include release levels and any known compiler or library bugs yet unfixed.
19. Are any "expert system" or standard AI programming methods used to construct the IPS? If so, do they require a learning period to evaluate "standard" client behaviors?
20. Have the IPS modules been certified as using Safe Programming Practices? If so, are any prone to buffer overflows, stack smashing, etc.?
21. Does any of the IPS software rely on a third party component, function, module, dll, etc? If so, have the third party components been evaluated with Fault Injection and Safe Programming Test Methods? And if so, can you share the test results?
22. Can the IPS protect the integrity of it's processes and threads when attacked by unknown processes? If so, How?
23. What notification model does the IPS Control Engine follow when major OS services are disabled on a client, (e.g., another module that disables a NIC used to send messages to the management server)?

VII. APPENDIX

A. "Top 50" Trojan, Worm & Spyware

- T1. Senna Spy Trojan Generator
- T2. Win32.Executor trojan (doesn't register) is a RAT
- T3. Hack´a Tack v.TE - RAT,Keylogger,Steals passwords,IP Scanner
- T4. Pretty Park - RAT,Steals passwords,Worm,IRC trojan,Mail trojan
- W5. Agobot - DDoS Attacker
- S6. Spyware - Dataminer from Web 3000
- W7. Muma Backdoor, Data Stealer & AV killer
- T8. Sockets des Troie - RAT,ICQ trojan,Virus, Alias
Lame Bug

- T9. JammerKillah - Anti-protection,RAT,Hacking tool,Trojan dropper
- T10. NetSphere - Backdoor Trojan w RAT, Keylogger, ICQ trojan, Process Killer
- T11. SubSeven - RAT, ICQ, IRC & Network Trojan
- S12. Spyware - Dataminer from Surf+
- T13. Mitglieder.AI - Proxy Trojan & mail relay
- T14. Deep Throat - Replaces system tray
- T15. Acid Shivers - Anti-protection trojan,RAT,Steals passwords, uses Telnet
- S16. Spyware - BHO, Drive-by downloader & Dataminer from Timesink
- T17. NetBus derivative - RAT,Keylogger,Eavesdropper
- W18. Welchi - RPC Worm
- W19. Sasser Worm exploit of LSASS as Internet Worm
- W20. Code Blue IIS Hack Tool
- W21. Mimail Worm - DoS launcher
- W22. Sober Worm – email worm disguised as a security warning
- W23. Nyxem - (multi component) email attach jpg, prockiller
- W24. NetSky.X - DDoS launcher spreads with email
- T25. UrlSpooF.E - trojan spy drops dumaru worm
- W26. Dumaru - Worm w Trojan Dropper, Password stealer, prockiller
- W27. Bugbear.E - Keylogger,prockiller, mailer
- T28. Dluca - Trojan downloader
- W29. Bagle.AY - Mitglieder trojan dropper & prockiller
- S30. Stealth Group - Keylogger & mailer
- S31. Keylogger, stealth process, prockiller, mailer (Stealth Group)
- T32. Hook Dumper - Keylogger & mailer (Stealth Group)
- S33. KeyKey - Keyboard logger
- T34. Brutus HoobieNet - Password Stealer w NetBus
- S35. Web Spyder tool - Eavesdropper, info stealer
- S36. Passgrab – Active X password stealer
- S37. BackWeb Backdoor downloader, eavesdropper
- S38. ProBot Activity Monitor - KeyLogger, mailer like StealthLogger 1.6
- S39. PC Spy - Keylogger & Screenshot capture
- S40. Spector Pro 3.1 - Keylogger & Surveillance utility
- S41. Ghost Keylogger – Webcam & screen capture, etc.
- T42. Trojan.Spy.Win32.Logger - (Stealth Group)
- T43. Global Killer - Remote Admin Trojan
- S44. ComLoad - ActiveX control to run auto-dialers
- T45. Stash Trojan - data stealer, & mailer
- T46. Doly Trojan - RAT,Keylogger,IRC trojan w mailer
- T47. GirlFriend Trojan – Password stealer w mailer
- T48. BIOS Password Stealing Trojan w mailer
- T49. Polymorph - Trojan Generator constructor
- T50. Bios_killer - Anti-machine Trojan

Note: The numbers preceded by (T) are normally classified as Trojans, (W) as Worms and (S) as Spyware

Disclaimer: nGran believes that the information in this document is accurate as of its publication date; such information is subject to change without notice. nGran is not responsible for any inadvertent errors or your use of this information. nGran and the nGran logo are registered trademarks of nGran. All other trademarks and registered trademarks are property of their respective owners.

©2005 nGran. All rights reserved.