GIS Security Innovation - Research and Development
Technology Evaluation Report and Recommendations

Executable Write Blocking for Malware Control Using Abatis HDF (Hard Disk Firewall)

GIS Cyber Defense Matrix Block	**Devices / Protect**
Principal Investigator	**Jon Codispoti**
Report Version	**1.0**
Report Date	**April 22, 2016**

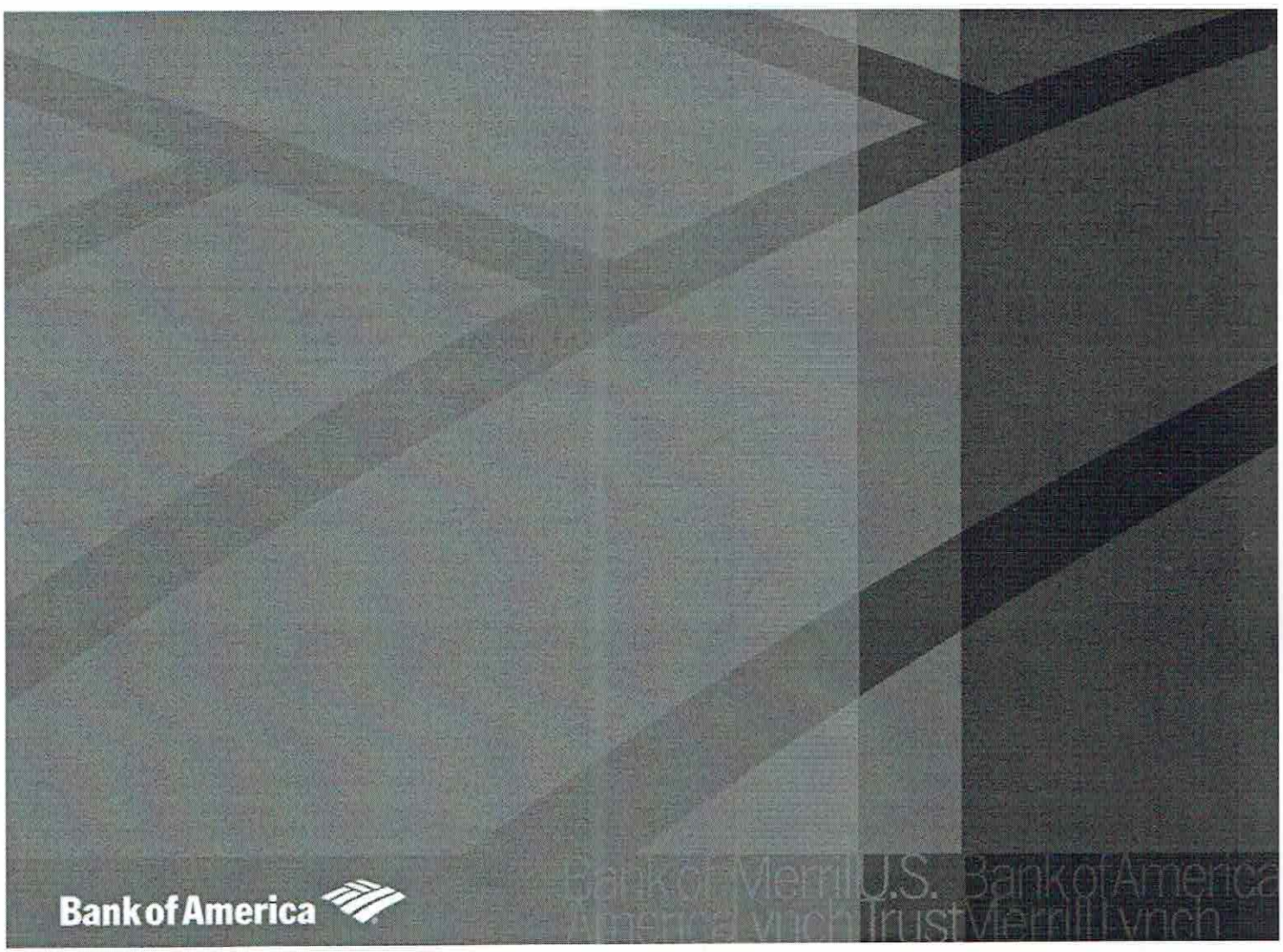

Table of Contents

Executive Summary

Bank of America provides world class protection for its computer systems through various layered controls which are very effective at stopping malware in its tracks. While the number of endpoint system rebuilds due to infection has been drastically reduced over the years, because the Bank runs vendor provided computer operating systems like Microsoft Windows, there will always be the risk of new types of malware attacks that can bypass our security controls. One approach the Bank is not currently utilizing to provide protection to our Windows based systems, is to simply **prevent any executable from being written to disk** to be run at a later time (i.e. prevent malware's ability to persist).

The ***Abatis HDF*** product was tested in our Research and Development lab and was found to offer a simple solution to prevent any unknown executables from being written to disk. While this does not offer full protection against new malware (e.g. it cannot protect from in memory infections), it does offer a new layer of defense not currently being employed by the Bank. This protection fits in the "Devices/Protect" square of the GIS Cyber Defense Matrix. Additionally, ***Abatis HDF*** has the potential to be leveraged by other tools to broaden our sensing capabilities at the endpoint level to further improve our visibility to threats and increase our opportunities to react. In this role Abatis HDF would live in the "Devices-Detect" square of our matrix.

Some general use cases where the **Abatis** product can be considered to augment current security controls include:

1. **Windows Application Servers** – ***Abatis HDF*** provides its protection as a very light footprint (less than 100KB) Windows driver, making it very different from many of the heavy agents in use at the Bank today. ***Abatis*** works particularly well in environments that are fairly static, with all changing executable content already managed by other tools. ***Abatis*** could be added to standard Bank application server build images and the protection it provides enabled as a final step before moving the system to production, preventing any new executable content being written to the server's storage and helping to ensure the overall integrity of the system after initial setup.

2. **Vendor Appliances / Turnkey Servers** – ***Abatis HDF*** has been programmed to work across many different versions of Windows and operate very efficiently. Because its operations are very focused and well defined it does not interfere with any normal system operations. The ability to provide strong protection against malware persistence with no system impact positions ***Abatis HDF*** as one option for the Bank to add protection to vendor provided systems installed on *BARONET*. Providing standard Bank security tools on vendor installed systems can be a large challenge because vendors have their own need to minimize changes and ensure performance and compatibility for their product. This can be particularly contentious when it comes to anti- virus products which have a heavy footprint and require constant updating. ***Abatis HDF*** is well suited to be installed on any vendor provided Windows platform to add an effective security control in situations where we may not otherwise be able to offer any other endpoint protection, in a way that does not impact system performance and requires zero maintenance after installation.

3. ***Abatis* as a Sensor** – Because Abatis HDF has visibility to all low level file operations, it is uniquely positioned to record these changes on any endpoint it is installed on. When operating in "learn mode" it maintains a log of any changes to the system that would violate its rules (policy), but still allow the changes to ensure normal operations. By deploying Abatis' capabilities alongside other tools like Tanium, we could gain the ability to see and control all executable content being saved to disk. When the appropriate Tanium scripts, within seconds of a new attack we'd be able to quickly engage Abatis

controls to prevent new executables from being written to Tanium instrumented systems until other malware controls were updated to repel the attack. If the attack were to have occurred on an Abatis protected system, we'd have records of that attack that could quickly be consumed by Tanium for reporting and investigative purposes.

Problem Overview

What Are We Trying to Solve

Bank of America currently supports multiple flavors of Microsoft Windows computer operating systems throughout the enterprise. To protect these systems from attacks which leverage vulnerabilities that may be present in these operating systems, the Bank installs various software security tools on these endpoints and on the network devices that connect these systems. The primary goal of this layered approach is to detect and defeat threats being delivered through the Bank's network before real damage can occur.

One class of attacks that are difficult to defend against are "zero day" attacks, which use new exploit methods that may not be detected by current endpoint protection products. To prevent damage which may occur from these "zero day" attacks, the Bank deploys various technologies to purge this malware before it ever makes it to the endpoint stations. While these methods work extremely well, there is never a guarantee that a new attack method will not be successful against a Bank endpoint and start spreading from there. New "zero day" attacks which successfully make it through all our various perimeter controls will be facing our last line of defense, our endpoint controls. While the Bank also provides a strong stack of protective tools on our endpoints themselves, there is always a chance for new malware threats to make it into our systems.

Proposed Solution

Assuming a new malware attack is able to be successfully delivered to a Bank endpoint due to a "zero day" exploit being used, for this malware to persist, spread, survive reboots, and inflict the most damage possible, it usually needs to write itself (or secondary stage) to disk storage at some point in its lifecycle. If we are able to simply stop the writing of executable code to disk, we can stop most malware from ever being delivered to our systems. For infections which do gain some sort of foothold in a computer system's memory, their ability to persist will be hampered if they cannot write to disk.

Based on how malware works and its need to write executable code to be successful, the Bank can leverage this knowledge to consider the introduction of a new type of security control to help protect its systems. The purpose of this document is to perform a deeper dive on this topic, to go through the results of an product evaluation conducted to test out these ideas and their applicability in the bank computing ecosystem, to set forth some use cases where this technology may be beneficial, and to present any recommendations for moving forward.

Executable Write Blocking

Exploring the Concept

As explained in the proposed solution paragraph above, the concept of blocking unknown executable content from being saved to disk storage is easy to grasp. If malware can be hampered in its ability to persist on any endpoint device, we can interrupt a very important stage of its lifecycle and in doing so

neuter any attacks that require this capability. Diving deeper into this idea there are some important additional points worth discussing.

1. No Threat Intelligence Required

All standard endpoint security products share a common need for some sort of intelligence in order to differentiate between good and bad. Whether this intelligence is based on something very specific like anti-virus signatures, adaptive as in heuristic based analysis, or based on behavior analysis as occurs in virtual machine detonations, each method used has been developed over time in response to ever changing tactics by our advisories to evade detection, This intelligence is costly to research and maintain and generally only partially effective as the game of technology leapfrog moves forward over time. The concept of blocking the writing of executables requires no specific knowledge of the threats we face other than their one shared need to persist to disk.

2. Proactive vs. Reactive

Traditional anti-virus is reactive not proactive. Infections must first occur in order for the anti-virus software vendors to collect samples and generate signatures. The same holds true for other security products we use that rely on complex algorithms and systems to analyze for behaviors learned only after studying real attacks. When new vulnerabilities are discovered and leveraged by malware authors or new attack methods developed by hackers, our security tool vendors must react before they can offer us any protection. Taking the posture of blocking the writing of all unknown executable content is unique because it is a purely proactive security control. Once the software is initially configured and the write protection enabled, no further attendance is required, even in the face of an ever changing threat landscape.

3. Self–Teaching

While executable write blocking requires no updated knowledge of external threats to continually benefit from the protection it provides, the system does need to be aware of trusted processes and executables in order to allow normal operations and system updates to occur. To make this local intelligence gathering easier the software uses its own unique view into the Windows file system to help automatically capture all the required data. The "learn mode" is used to easily establish a baseline of what "normal" is, allowing it to easily establish when "abnormal" attempts to save executable code to disk later occur. Through simple processes of setting up baseline configurations and recording normal activities, the system can quickly learn what it needs to operate going forward in order to allow the "good" and deny the "bad".

4. Low Impact

Endpoint security tools are programmed to minimize impact to the system and go to great lengths to coexist peacefully with other software on the devices they run on. Over time however these tools have to evolve and as the attack methods and capabilities they work to detect change, these security tools typically only get larger, more complex, and consume more system resources to accommodate these needs and thusly impact the protected systems to a greater level. Computer hardware has evolved over time as well, so there have typically been quick and easy fixes to these issues through hardware upgrades which add more system resources to in turn better support our security tools. This cycle becomes crazy though and only incremental changes in protection are ever achieved while processing requirements typically remain the same or worsen, but do not lessen. By adopting the simple security control of blocking the writing of unknown executable content we have the opportunity to introduce a simple capability that provides a long lasting control that should very rarely need

updating and that offers a very consistent and minimal fingerprint in terms of processing requirements. The software required to implement this control is minimal because its task is minimal, so only consumes less than 100K of system memory. Additionally, because the software is implemented as a Windows kernel mode driver, it operates at a very low level, so has been programmed to Microsoft specifications to be very efficient and not interfere with any normal processing. This small and efficient footprint is in contrast to many modern security agents which typically contain feature bloat that add to processing complexity and use many "user mode" components which slow down overall operations.

5. Low Maintenance

Apart from the required "learning" process which helps differentiate between "good" and "bad", the software which offers disk write protection requires very little maintenance after initial deployment. If deployed to static environments where system executable content does not change, the only maintenance required may be an occasional software upgrade from the vendor. In other environments there will be some management overhead involved to account for system changes; however this can be easily accommodated by integrating this management with the software deployment functions already in place. An added benefit of this technology is that the same protection deployed to protect from malware installations works equally well to prevent rogue user installations, allowing the Bank to further tighten its grip on the control of Non-Permitted Technologies (NPT). For most all other security tools we deploy, there is a stiff maintenance tax paid in terms of signature changes and software updates, simply to keep the software viable. Through careful deployment and development of proper operational procedures, the blocking of unknown executables has the potential to offer tremendous long term "bang for the buck" in terms of the security protection it offers for our endpoint systems.

6. Bonus Features!

Most endpoint security products we deploy provide very good coverage for their specific use cases and reporting that's really only useful for the specific coverage it provides. Because executable write blocking protection operates at a very low level and has unique visibility inside the Windows OS, there are some additional benefits that can be leveraged through this tool deployment:

- Once executable write blocking is enabled the software automatically aids in maintaining general system integrity, which helps in other Bank initiatives to prevent NPT or software licensing abuse.

- Rules can be adjusted to cover additional Bank defined file types, allowing custom write controls to be easily added over time, extending the capabilities of the control to meet Bank specific needs.

- The software can perform extensive audit logging for all file system requests. This positions the tool to be used as a unique sensor that we can remotely control and extend through tools such as Tanium. Having visibility to low level file operations within seconds and enabling more reactive write control features when threat levels increase become capabilities we can potentially leverage by combing these toolsets.

- As the tool is rolled out, there may be extant undetected infections on systems (often referred to as Advanced Persistent Threats or APTs). The unique operation and extensive audit log capabilities of the software can allow this malware to be identified. In some cases it can also reveal "root kit" infections and facilitates the subsequent removal of such programs (vendor claim, not validated in our testing).

- One unique claim for this technology is that it can help save on energy costs and improve laptop battery life. While this sounds somewhat farfetched, there is some easy to follow logic behind it. If disk write requests get ignored, it saves energy because of the CPU processing and electromechanical activity that is not generated. While this may add up during an attack, these savings are not really seen as a selling point. However, if one considers turning off traditional anti-virus tools in favor of this simpler technology, at the Bank's scale significant energy cost savings could be realized due to the massive overhead of AV tools in both CPU and disk activity for constantly inspecting the same content over and over again.

Testing the Concept – Abatis HDF Product Evaluation

In order to perform some hands on testing of the concept of executable write protection, the Research and Development Team engaged with the UK vendor named Abatis, who pioneered this technology and have dubbed it Host Integrity Technology (HIT). In addition the providing executable write protection, their product (Abatis HDF) offers several other features to make it enterprise friendly. Features such as built in tamper protection, a configurable user interface, and optional centralized management make it easy to roll out and control in the environment. Abatis HDF is also being evaluated by some other large very firms, though details on any who have become customers have not been provided.

The evaluation undertaken with the Abatis HDF product for the Bank was designed as just a general review of the overall technology and not a full product evaluation that is intended to drive the selection of a specific tool to deploy. More information covering the scope of the review conducted and implementation specific details on how the evaluation was set up follows.

1. Evaluation Overview

The primary goal of the evaluation conducted was to validate that write blocking of unknown executable content is achievable and most importantly offers value as a protective control for the Bank. While validating the first two points is fairly straightforward, fully testing the protective claims is not really possible because "we don't know what we don't know", however we can test how some of these protective features would stand up against known threats and can extrapolate from there.

2. Scope

Abatis offers its HIT technology for both Windows and Linux platforms. Within each platform Abatis also supports various versions of the target operating system. The scope of the testing completed for this technology review was constrained to a single version of Microsoft Windows Server. Because Abatis works the same no matter the operating system version it is installed on, this limited scope of testing is sufficient to simply evaluate the technology. Additionally, while Abatis does offer a centralized management console, we did not look at this software. The management of the Abatis HDF driver is something the Bank would need to understand for a full evaluation, but was not in scope for our goals in this evaluation process so will only be given a cursory look.

3. Experiment Setup

Based on the limited scope of this technology review, the setup requirements for this experiment were minimal and consisted of:

1. One virtual device in the R&D lab
2. Microsoft Windows Server 2012 Release 2 installed using standard Bank server build process with all default security and management tools pre-installed.

3. Abatis HDF Extended Edition Version 1.0 installed after server build completed and all updates verified as current.

4. Test Plan

A test plan was devised to cover many of product claims made by the vendor which are also key features of the technology in general. Not covered in the test plan are performance tests, which we are not instrumented to evaluate in our lab, tests which would require the introduction of live malware onto the system, and some broad claims which do not lend themselves to lab environment review. The test cases covered in this plan are not exhaustive and are only intended to evaluate the technology's baseline protection to prevent malware persistence. The plan has been broken up into several groups of tests, each summarized below. The full test plan with all detailed test steps is included in the Appendix of this document.

Test Group	Purpose of Tests
Test Group 1	To verify proactive protection against saving unknown executable content to disk (malware persistence)
Test Group 2	To verify prevention of unauthorized system and file modification on a web server computer
Test Group 3	To verify self-resilience protection against hostile attack and shutdown
Test Group 4	To verify proactive prevention of malware and APT persistence/saving on a computer - using real malware

5. Test Results

Based on the test plan that was followed, the Abatis HDF product performed very well. From the core technology perspective, it passed all the tests we threw at it. It is an effective software tool for implementing write blocking of executable content. If the Bank wishes to further evaluate the concept of this new type of security control, Abatis does offer the tools and the support resources we'd need to pursue this further.

The full test plan with results is included as an appendix to this document. A summary of these results is presented below:

Test Group	Test Environment	Group Results
Test Group 1	R&D Lab, BAND Windows Server 2K8	**Passed** all test cases
Test Group 2	R&D Lab, BAND Windows Server 2K8	**Passed** all test cases
Test Group 3	R&D Lab, BAND Windows Server 2K8	**Passed** all test cases
Test Group 4	Air Gapped VM, MS Windows Server 2K8	**Passed** all test cases

Technology Evaluation Results

Results Summary

The ***Abatis HDF*** product was found to offer a simple solution to prevent any unknown executables from being written to disk. While this does not offer full protection against new malware (e.g. it cannot protect

from in memory infections), it does offer a new layer of defense not currently being employed by the Bank. Additionally, ***Abatis HDF*** has the potential to be leveraged by other tools to broaden our sensing capabilities at the endpoint level to further improve our visibility to threats and increase our opportunities to react.

While the protection offered by Abatis does look compelling, this does of course come at a cost of adding some additional complexity to the environment. The requirements to learn about all pre-existing executables on a system and to properly manage the deployment of all new valid executable content would add some overhead to server deployments and necessitate some process changes to ensure there are no software conflicts. The vendor has done a very good job at building in tools to help automate this process though, so integrating this product into standard build processes does appear to be very feasible, especially in "cookie cutter" environments that rarely change.

Possible Bank Use Cases

Due to the nature of this product, any broader proof of concept efforts to further test its capabilities should be carefully considered. The best candidate environments for further testing would be those with infrequently changed systems that also already follow stringent change control for any updates. Much could be learned by first targeting this sort of "low hanging fruit" because it would allow us to further evaluate how to integrate the product with existing Bank processes and controls with minimal risk of any business impact in doing so. Some thought was put into use cases where the **Abatis** product could be considered to augment current security controls at the Bank, and some examples follow.

1. Windows Application Servers

Because the product provides its protection in a very light footprint (less than 100KB) Windows driver, it is very different from many of the heavy agents in use at the Bank today. ***Abatis*** works particularly well in environments that are fairly static, with all changing executable content already managed by other tools. ***Abatis*** could be added to standard Bank application server build images and the protection it provides enabled as a final step before moving the system to production, preventing any new executable content being written to the server's storage and helping to ensure the overall integrity of the system after initial setup

2. Vendor Appliances / Turnkey Server

The Abatis product has been programmed to work across many different versions of Windows and to operate very efficiently. Because its operations are very focused and well defined, it does not interfere with any normal system operations. The ability to provide strong protection against malware persistence with no system impact positions ***Abatis HDF*** as one option for the Bank to add protection to vendor provided systems installed on *BARONET*. Providing standard Bank security tools on vendor installed systems can be a large challenge because vendors have their own need to minimize changes and ensure performance and compatibility for their product. This can be particularly contentious when it comes to anti- virus products which have a heavy footprint and require constant updating. ***Abatis HDF*** is well suited to be installed on any vendor provided Windows platform to add an effective security control in situations where we may not otherwise be able to offer any other endpoint protection, in a way that does not impact system performance and requires zero maintenance after installation.

3. Abatis as a Sensor

Because Abatis HDF has visibility to all low level file operations, it is uniquely positioned to record these events on any endpoint it is installed on. When operating in "learn mode" it maintains a log of any changes to the system that would violate its rules (policy), but still allow the changes to ensure normal operations. By deploying Abatis' capabilities alongside other tools like Tanium, we could gain the ability to see and control all executable content being saved to disk. When the appropriate Tanium scripts, within seconds of a new attack we'd be able to quickly engage Abatis controls to prevent new executables from being written to Tanium instrumented systems until other malware controls were updated to repel the attack. If the attack were to have occurred on an Abatis protected system, we'd have records of that attack that could quickly be consumed by Tanium for reporting and investigative purposes.

4. ATM Systems

Because host integrity and malware protections are just a couple of key features already built into the Bank's ATM systems, Abatis seems a perfect fit for this environment to further augment these two protections. Once integrated into the ATM system build and update process, Abatis would provide an additional layer of protection at the file system driver level and not interfere with any normal operations. In the event of the failure of any higher level write controls which would leave the system vulnerable, Abatis would be another layered control on the device at a level where there is currently no other protections in place.

Conclusions and Next Steps

Abatis HDF offers a very solid level of write protection to prevent persistence of unknown executable content, which in turn thwarts most forms of malware from gaining a stronghold on the endpoint. This protection comes at a very low cost in terms of system impact and execution and with only moderate effort in terms of management (assuming we can integrate with current tools). The simple way Abatis works though, would also be its primary impediment for general acceptance as a core protective control, because the Bank is reliant on the proper execution of software executables to run its businesses. With this in mind, some recommendations on the best way to proceed follow:

- Further evaluate the tool in specific tightly controlled environments, such as in the example use cases already provided. By rolling the tool out into such specific environments and instrumenting them to report back protected writes, we'd be able to gain both peace of mind protection and valuable data on that protection should it ever trigger.

- As GIS begins engaging on the conversations R&D is starting regarding "imagining new security perimeters", start considering the storage connected the Bank's endpoints as a new security perimeter. In this context the Abatis product makes perfect sense, as it is a firewall for storage which implements a "default deny" policy for all executable content and supports rules which can be customized to "open holes" to further support our business needs. When security is considered at this level of granularity, Abatis becomes quite a viable control for this perimeter.

Appendix

Test Plan

The overall test plan was broken down into 5 groups of tests, each focused on testing different types of product functionality or protection. Each test group contains several individual steps with the expected results for each operation.

Test Group 1

Purpose: To verify proactive protection against saving unknown executable content to disk (malware persistence)

Note: For all of the following tests, the user account used is a member of the local "Administrators" group.

Abatis Evaluation Test Plan - Test Group 1				
#	**Test Case**	**Expected Results**	**Test Results**	**Test Objective/Remarks**
1	Installation – run installer and import license key. If not already, configure HDF to run in protection mode – via context menu option->Settings, or using configuration tool, HDFControl.exe	HDF runs in 'normal mode'.	Passed	Installation successful. HDF running in blocking mode
2	Download, copy or move an executable file to any folder, e.g. **C:\WINDOWS\system32\notepad.exe** to **C:\notepad.exe**	Action blocked - cannot save executable to folder	Passed	Prevents malware injecting code or saving executable file on the system.
3	Copy a real executable with a fake file extension, e.g. **notepad.exe** to **notepad.tmp**	Action blocked - cannot copy a real executable with a non-executable file extension.	Passed	Block malware that conceals and hides its file extension
4	Rename to a real executable with a fake file extension and with 'hidden' attributes set, e.g. *[hidden]***notepad.txt** to **notepad.exe**	Action blocked - cannot rename a non-executable file extension to an executable file.	Passed	Block malware that is hidden and/or with read-only attribute set
5	Download or copy a real executable from the Internet, network share and removable device, e.g. **H:\notepad.exe** to **C:\notepad.exe.** Note: **H:** is presumed a network drive or a removable drive such as a USB drive.	Action blocked - cannot save executable to drive from network and removable device	Passed	Prevents drive-by infection, worm spreading over network and removable devices, USB drive.
6	Download or copy a real executable with a fake file extension from the Internet, network share or removable device, e.g. **H:\notepad.txt** to **C:\notepad.exe** or **C:\notepad.tmp**. Note: **H:** is presumed to be a network drive or a removable drive such as a USB drive.	Action blocked - cannot save an executable with a non-executable file extension from network and removable device	Passed	Prevents malware that use camouflage techniques, encrypted code body, fake file extension in drive-by infection, worm spreading over network and removable devices, USB drive
7	Execute an embedded executable in a container file, e.g. Winword document	Action blocked – block auto-run of an embedded executable	Passed	Prevent malware infection via email attachment, hidden in MS Office document, PDF or other application file
8	Check HDF audit log file, HDF.log	All activity is logged – blocked I/O is indicated with a code <1> and allowed I/O with a <0> code	Passed	All block and non-block activities are logged.
9	Uninstall test (HDF must be fully disabled)	HDF uninstall successfully and completely reoved	Passed	HDF uninstalled and all registry entries removed. HDF log file will be removed if not archived.

Test Group 2

Purpose: To verify prevention of unauthorized system and file modification on a web server computer

Note: For all of the following tests, the user account used is a member of the local "Administrators" group.

Abatis Evaluation Test Plan - Test Group 2				
#	**Test Case**	**Expected Results**	**Test Results**	**Test Objective/Remarks**
1	Installation – run installer and import licence key. If not already, configure HDF to run in protection mode using the configuration tool, HDFControl.exe	HDF runs in 'protected mode'.	Passed	Installation. HDF running in blocking mode
2	Copy a new protected file, e.g. <index.htm> or <products.gif>, to a web site folder.	Action blocked - cannot save protected file to the computer/web server	Passed	Prevents website defacement attack - an attacker uploads his version of unauthorized web files and/or graphic files to a web server
3	Overwrite a protected file, e.g. <index.htm> or <products.gif>, with the attacker's version with the same file name	Action blocked - cannot replace protected web files with attacker's files.	Passed	Prevents website defacement attack - an attacker replaces existing web files and graphic files with the attacker's versions
4	Modify the contents of a protected file, e.g. <index.htm> or <products.gif>, in any web site folder after a successful server compromise	Action blocked - no unauthorized & unauthenticated file modification.	Passed	Prevents website defacement attack - an attacker attempts to edit existing web pages to include 'iFrame injection' and Java script type malicious contents.
5	Repeat the above test cases (2-4) to target a web server's (e.g. IIS) system configuration file, e.g. <web.config>	Action blocked - no unauthorized modification of system and file configuration files	Passed	System and file integrity protection
6	Upload or copy an executable or real malware from the Internet, network share or removable device to the computer, e.g. **H:\notepad.exe** to **C:\notepad.exe** - hijack the site as a drive-by download malware distribution server. Note: **H:** is presumed a network drive or a removable drive such as a USB drive.	Action blocked - cannot save an executable file from the Internet, networked drives and removable device	Passed	Prevents a hacker hijacking a web server to host and launch drive-by attack and as a malware distribution server.
7	Download or copy an executable or real malware from the Internet, network share or removable device with a fake non-executable file extension, e.g. **H:\notepad.txt** to **C:\notepad.exe** - hijack the site as a drive-by download malware distribution server Note: **H:** is presumed a network drive or a removable drive such as a USB drive.	Action blocked - cannot save a camouflaged executable file from the Internet, networked drives and removable device	Passed	Blocks any malware that use camouflage, fake file extension and encrypted malware code in drive-by infection, worm spreading over network and removable devices, USB drive etc.

Abatis Evaluation Test Plan - Test Group 2				
8	Check HDF audit log	All activity are logged – blocked I/O is indicated with a code <1> and allowed I/O with a <0> code	Passed	All block and non-block activities are logged.
9	Uninstall test (HDF must be fully disabled)	HDF uninstall successfully and completely removed	Passed	HDF uninstalled and all registry entries removed. HDF log file will be removed if not archived.

Test Group 4

To verify proactive prevention of malware and APT persistence/saving on a computer - using real malware

Note: For all of the following tests, the user account used is a member of the local "Administrators" group

Abatis Evaluation Test Plan - Test Group 4				
#	**Test Case**	**Expected Results**	**Test Results**	**Test Objective/Remarks**
1	HDF operates in blocking mode. A malware infected USB stick is used. The malware under examination is `W32.SillyFDC` (Symantec naming). The USB is inserted to a test computer and the malware will auto run to infect the host. Note: the malware `W32.SillyFDC` executable file is <schost.exe> which is a similar file name to the system component <svchost.exe>. It spreads by USB drive and shared network drive.	The malware executes and attempts to infect the host by copying itself to the host folder in C:\WINDOWS\SYSTEM32\SCHOST.EXE. The malware action is blocked and the infection attack is prevented.	Passed	HDF log shows the malware action is blocked, e.g. code <1>: 2010/06/24 19:06:03 <1> 1888:schost.exe S-1-5-21-823518204-261478967-299502267-1003 C:\WINDOWS\SYSTEM32\SCHOST.EXE Refer to Test Case diagram 1 below.
2	Turn off HDF protection. Repeat the same test case as above.	The malware executes and successfully infects the host by copying itself to the host folder in C:\WINDOWS\SYSTEM32\SCHOST.EXE Since HDF is turned off the malware successfully infects the host. HDF log shows the malware action.	Passed	HDF log shows the malware action is not blocked and successful enters the system, e.g. code <0>: 2010/06/24 19:14:43 <0> 1824:schost.exe S-1-5-21-823518204-261478967-299502267-1003 C:\WINDOWS\SYSTEM32\SCHOST.EXE Refer to Test Case diagram 2 below.
3	HDF operates in blocking mode. Evaluate an Internet worm commonly called Sdbot and Spybot. The worm's executable file is <dns.exe>. To run the test case, copy the executable file to an USB drive and configure it to auto run using <AutoRun.inf>, or execute it manually. The worm can also execute from a network share drive. Detail of Sdbot: www.sophos.com/security/analyses/viruses-and-spyware/w32rbotgky.html	The worm executes and is prevented from infecting the host. It attempts to copy itself <MSWDNS32.EXE >to the host folder The malware action is blocked and the infection attack is prevented.	Passed	HDF log shows the malware action is blocked, e.g. code <1>: 2010/06/24 23:39:41 <1> 1544:dns.exe S-1-5-21-823518204-261478967-299502267-1003 C:\WINDOWS\SYSTEM32\MSWDNS32.EXE HDF log shows the malware action is not blocked and successful enters the system, e.g. code <0>: 2010/06/24 23:43:25 <0> 1528:dns.exe S-1-5-21-823518204-261478967-299502267-1003 C:\WINDOWS\SYSTEM32\MSWDNS32.EXE Refer to Test Case diagram 3 below.
4	Turn off HDF protection by selecting Software install/update option (Standard version) or using the configuration tool, HDFControl.exe (Advanced version). Repeat the test case 3 as above. It is interesting to note that SDBot can survive removal/cleaning action from some anti-virus programs, and persist on the infected computer.	The malware executes and successfully infects the host by copying itself to the host folder in C:\WINDOWS\SYSTEM32\MSWDNS32.EXE Since HDF is turned off the malware successfully infects the host. HDF log shows the malware in action.	Passed	HDF log shows the malware action is not blocked and successful enters the system, e.g. code <0>: 2010/06/24 23:43:25 <0> 1528:dns.exe S-1-5-21-823518204-261478967-299502267-1003 C:\WINDOWS\SYSTEM32\MSWDNS32.EXE Refer to Test Case diagram 4 below.