Abatis Hard Disk Firewall Technical Overview

by

Platinum High Integrity Technologies UK

ABSTRACT

Abatis Hard Disk Firewall (HDF) takes a different approach and uses several simple strategies, but very powerful in the never-ending fight against malware.

Contents

Introduction

To affect a device, malware typically needs to execute code on the CPU. Traditional anti-malware defences try to examine executable code and determine if it is good or bad. 'Good' code can run, and 'bad' code is blocked. The anti-malware process uses a combination of signatures, algorithms and heuristics to make the choice. This results in an arms race between the anti-malware providers and the bad actors. With half a million new pieces of malware being released a day (July 2023)[1], this is a race the anti-malware vendors can never win.

Abatis Hard Disk Firewall (HDF) takes a different approach and uses several simple strategies, but very powerful in the never-ending fight against malware.

The first and most important strategy is to freeze the device's operating system and application programs. By stopping runnable program files from being written to permanent storage, HDF stops malware from gaining a foothold on the device. If it is not on the device, it is very difficult for malware to run.

The second strategy is to prevent the execution of a program unless it resides in permanent storage on the device. This stops bad actors from plugging in a USB or connecting to a network drive and executing malware from storage under their control.

The third strategy is to stop fileless attacks by blocking the execution of 'Living off the Land' programs. Some malware gains a foothold by exploiting programming errors such as buffer overruns, out-of-bounds reads/writes and other code failures. Typically, the code which can be executed like this is limited in size. So, for a fileless attack to succeed, it co-opts programs already on the target device – known as 'Living off the Land' programs. Stopping the execution of these programs helps block fileless attacks.

The fourth strategy is using anti-tamper features to make it difficult for bad actors to turn off, alter, or disable HDF. This means they cannot bypass HDF to accomplish their intentions.

Together, these strategies provide a robust defence against malware.

These strategies mean that HDF does not need to download signature files and is suitable for online and air-gapped devices. HDF's decision-making process does not depend on huge numbers of signatures, heuristics, or rule sets. This means that the core code occupies a small amount of space – typically under 100k, and because the decision tree is relatively simple, system performance has negligible impact. HDF is especially suitable for low-powered machines with limited memory, IoT devices, etc.

HDF supports legacy Windows Workstations and Windows Servers from XP/NT4 to all current Windows versions on 32-bit and 64-bit platforms. It is, therefore, also especially suitable for legacy devices that are out of support for operating systems or anti-malware applications.

---

[1] https://dataprot.net/statistics/malware-statistics/

How HDF Stops Malware

Most malware exists as runnable programs, which on Microsoft Windows platforms have a specific format known as a PE (Portable Executable) file. PE files typically also have a well-defined set of file extensions. Some samples of common executable file extensions on Microsoft® Windows® platforms are *.exe, *.dll, *.ocx *.sys and others. HDF detect and blocks all attempts to create new Portable Executable (PE) files or alter existing ones.

A typical malware attack life cycle goes as follows:

- The attack begins with what virus researchers call a Warhead attack. A warhead is a very small part of the malware code that exploits known/unknown system vulnerabilities, e.g., buffer overrun attacks, to enable it to run on a computer. The warhead code is necessarily small with very limited functions; the infamous SQL Slammer has a warhead of 376 bytes.
- The next step is to transfer the Payload code to the victim's computer. This payload could be anything – e.g., ransomware, botnet, wiper, etc. The warhead may download the payload from the internet or open a command shell so the attacker can download it by hand. The payload may also contain propagation code so the malware can continue spreading.

HDF intercepts malware's life cycle at the most critical point of propagation, i.e., saving its payload in a program file for permanent storage. Without propagation, the malware cannot run its payload or spread further. It is, therefore, effectively terminated.

If the malware attempts to run the payload without copying it to local storage first (e.g., by running it from a remote server on the internet), then HDF will block the program from being executed.

If the malware attempts to use a fileless attack, it will typically do this by co-opting other programs already existing in a Living of the Land" (LotL) attack. Most of these programs are not needed in day-to-day usage. HDF will block these attacks by preventing the LotL programs from executing.

Architecture and Technical Description

Under the Windows operating system, all file operations must use system kernel services, and no programs, including malware, can write to the hardware directly.

Briefly, Windows enforces privilege-checking mechanisms to guarantee system integrity and has the concept of kernel mode (Ring 0) and user mode (Ring 3). The CPU enforces access so that a user-mode program cannot directly access kernel memory and can only use kernel services by a well-defined set of APIs. The OS prevents direct access to all hardware devices, such as a hard disk, by a thin layer of system software called the Hardware Abstraction Layer (HAL). When Windows is running, all input and output (I/O) accesses are provided by kernel service calls.

HDF integrates into the kernel as a file filter. It uses well-defined Windows APIs and conforms to the file filter specifications. It does not use any undocumented Windows structures or APIs. It is, therefore, a well-behaved part of the Windows operating system. It loads early in the boot process, reading its initial operating parameters from the Registry.

All file operations pass through HDF, and the operating system enforces this. HDF analyses them, and either lets them through or blocks them. Most operations are let through. However, HDF will stop attempts to alter or create a PE file.

HDF does not rely on file extensions to block write attempts to PE files, although it does use file extensions to speed the process. However, HDF will also look at the structure of the file. PE files have a specific format, and if they do not have this format, the operating system will refuse to load and execute them. If an attempt is made to write to a file, HDF will check the format and block the write if the file has the structure of a PE file.

It is, therefore, impossible for malware to write a PE file to storage controlled by the device or to alter an existing PE file. Figure 1 on the last page illustrates HDF's operation flow.

When the operating system attempts to execute a file, it opens it with the 'execute' property. This is a file operation and thus gets passed to HDF. If the file exists on removable media, then HDF blocks the access. Thus, malware and hackers with access to the device cannot execute arbitrary code not already on the machine. An example might be the attacker connecting to a remote file share on the internet under their control or inserting a USB which contains malicious programs.

HDF will also check the file being executed to see if it is on the list of banned LotL binaries, and if it is, it will also block access. Thus, malware cannot use LotL binaries to carry out any actions, e.g., in a so-called Fileless and Memory-only attack.

HDF also contains tamper-resistant code, which uses operating system features to protect itself, its configuration files, and its registry entries. It denies attempts to shut it down by malicious processes and bad actors even if they have the highest system privileges, e.g., Local_System.

Functional Features

The technical specifications of HDF are:

- Loads and runs as part of the operating system
- Loads during system boot-up before any user logs in.
- Resists attack attempts to shut down operations by hostile malware/hacker attacks and user mistakes.
- Able to identify and block creation/modification of executable files without pattern matching.
- Functions autonomously irrespective of user account system privileges, caller processes and threads, i.e., non-by-passable protection.

- Functions transparently to users and applications. No user interaction is needed. All existing applications run as usual without modification.
- Supports local, networked, removable storage devices and plug-and-play mounted devices, e.g., external USB devices.
- Operations are logged for audit trail purposes.
- Small application footprint, under 100k
- Negligible degradation of system performance.
- Compatible with antivirus programs, personal firewalls, and file encryption tools.
- No day-to-day user maintenance necessary and reliable operation.

Configuring HDF

HDF creates an immutable system where changes to the operating system and applications are not allowed. This is not necessarily useful in the long term. HDF, therefore, provides a rich set of methods by which its operation can be tailored to an organisation's specific requirements while still retaining control by the IT department.

HDF can be temporarily turned on and off to permit software installation or update. This operation is controlled so that only permitted users can do this.

Some security software needs to update itself periodically, and protection will degrade if this is not allowed. HDF can be configured via policy files so that this can still happen. The policy files allow tailoring by process name, file path and file name. Wildcards are allowed.

Depending on risk appetite, some organisations may want to allow other software to update itself (e.g., via Windows Update). The policy files can also be configured to let this happen.

HDF Logging

HDF creates a text log file which records the actions it has taken. This can be used for audit and logging purposes.

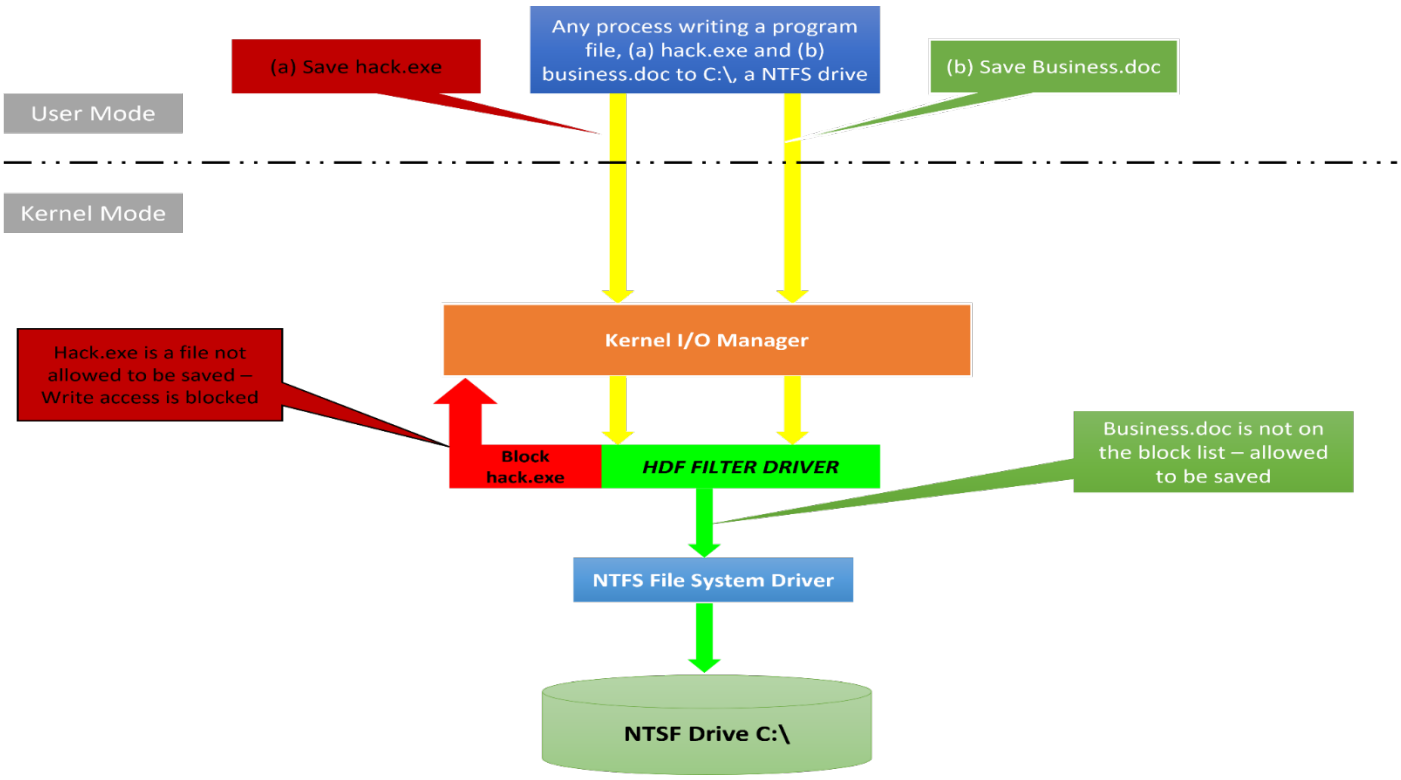

Diagram 1 HDF Filter Workflow