Windows Filtering Platform, engine for local security

Zoran Spasov, Ana Madevska Bogdanova[1]

[1]Faculty for Natural Sciences and Mathematics, Institute of Informatics
Skopje, Macedonia
zocisp@gmail.com, ana@ii.edu.mk

Abstract. The goal of this paper is to analyze the functionality and usability of the Windows Filtering Platform introduced by Microsoft Corporation as set of API functions that are controlling the network stack of the Windows operating systems. The applications build on it are intended to be used as the windows firewall replacement, with additional functionality in terms of the logic of packet filtering. Some examples demonstrating the usage of the platform will be described. In order to check the durability and exploitability of the systems built on this platform we will use the QualysGuard infrastructure by Qualys, specialized for vulnerability and open ports scanning. The results provided by the tests are analyzed and some conclusions are given.

Keywords: Windows Filtering Platform, QualysGuard, windows security, port scanning, vulnerability testing.

1 Introduction

Windows Filtering Platform (WFP) is a set of API functions and system services that provide a platform for creating applications that control the flow of packets through the networking stack. The code that uses the WFP API, allows developers direct interaction with the packet processing which is done at several layers in the networking stack of the operating system. Network data can be filtered and also modified before the packets reach the destination. WFP offers many application programming interfaces introduced with Windows NT 6.0 that allows applications to tie into the packet processing and filtering pipeline of the new network stack. It provides features such as integrated communication, so the applications coded on top of this interfaces can utilize the powerful processing logic that is based per application or per user bases. It is intended for use by firewalls and other packet-processing or connection monitoring systems.

WFP is a simple development platform, designed to replace previous packet filtering technologies such as Transport Driver Interface (TDI) filters, Network Driver Interface Specification (NDIS) filters, and Winsock Layered Service Providers (LSP). Starting in Windows Vista and Windows Server 2008, the firewall hook and the filter hook drivers are not available; applications that were using previously mentioned drivers should use WFP instead. With the WFP API, developers can implement firewalls, intrusion detection systems, antivirus programs, network monitoring tools,

M. Gusev (Editor): ICT Innovations 2010, Web Proceedings, ISSN 1857-7288

and some parental controls based on application or user policies. WFP integrates and provides support for firewall features such as installing filters at different stages of the tcp or udp communication, authenticated communication and also there is a dynamic configuration of firewall policy based on applications' use of sockets API (application-based policy). This platform also governs the area of secure communication like controlling and managing the IPSec infrastructure using different policies, raising change notifications, some network diagnostics, and stateful filtering of the connectivity [1].

Windows Filtering Platform as a development platform should not be considered as a standalone firewall engine. The existing firewall application Windows Filtering and Advanced Security (WFAS) built into the newer operating systems, such as: Windows Server 2008, Windows Server 2008 R2, Windows Vista, and Windows 7, is implemented using WFP. So, all these firewall applications utilize the WFP and WFAS features and the powerful processing logic for packet based filtering.

2 WFP: Terms, Definitions and Features

WFP using a set of components acts on the networking stack in order to control the packet flow that goes through the network interfaces on the computer. It is composed from these main components:

- *Filter Engine*. It represents the core multi-layer filtering infrastructure, hosted in both kernel-mode and user-mode, which replaces the multiple filtering modules in the Windows XP and Windows Server 2003 networking subsystem. This engine is responsible for packet filtering based on the firewall policy depending on the data it receives from the shim. Also if there are multiple filter policies that are enforced on a same packet data it provides arbitration between different policy sources and returns the proper action: “permit” or “block”.
- *Base Filtering Engine (BFE)*. It is responsible for controlling the operation of the filtering platform, accepts new filters, creates some statistics and reports the current state of the systems and enforces some configuration on different modules of the system, i.e. the IPSec negotiations.
- *Shims*. These are kernel-mode components that reside between the network stack and the filter engine. They make the filtering decision by classifying against the filter engine. There are many different types of shims: application layer enforcement (ALE), transport layer module, network layer module, etc.
- *Callouts*. They represent a set of special function that are called by a driver and they are used for special filtering, besides the basic “permit” or “block” based on a simple rule. Usually they are used by IPSec processing, adjust stateful filtering behavior, make some intrusion detection processing of packets, etc.
- *Application Programming Interface*. This is a set of data types and functions available to the developers to build and manage network filtering applications.

Windows Filtering Platforms offers many features and extensions to the developers. It also offers connectivity to a third-party filtering systems. Many of the key features that WFP offers are: provides a packet filtering infrastructure where independent software vendors can add some own specialized modules, it works with both IPv4 and IPv6, has a capability for data filtering, modification, and re-injection, performs both packet and stream processing, allows packet filtering to be enabled per application, per user, and per connection in addition to per network interface or per port, provides security during the boot-up process until the firewall engine is transferred to the user environment, enables stateful connection filtering, handles both pre and post IPSec-encrypted data, allows integration of IPSec and firewall filtering policies, provides a policy management infrastructure to determine when specific filters should be activated. This includes mediating conflicting requirements from multiple filters provided by different vendors, has a monitoring and reporting capability of the filtering process and many more [2].

2.1 Windows Filtering Platform architecture

The following picture explains the windows filtering platform architecture in detail:

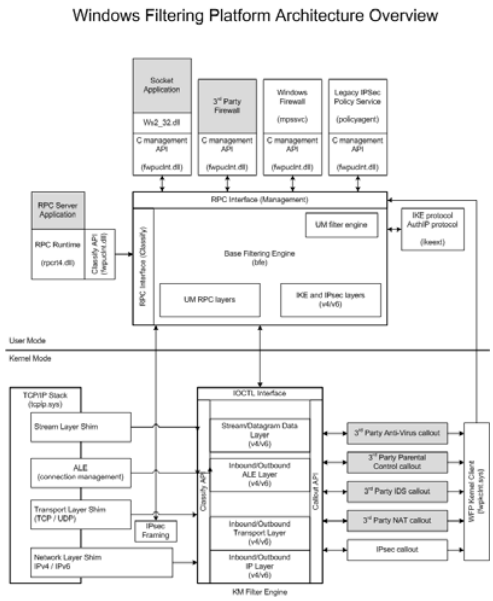

Fig. 1. WFP architecture and communication, courtesy of Microsoft Corp.

The filter engine contains two components that operate in different modes: user-mode and a kernel-mode component, which together perform all of the filtering operations on network packets or streams. The filter engine contains multiple filtering layers for every network layer in the networking stack of the operating system. The

division of the filter engine layers into user-mode layers and kernel-mode layers is based on the filter engine component that owns them.

The user-mode component performs RPC and IPsec filtering. The filter engine contains approximately 10 user-mode filtering layers.

The kernel-mode component performs filtering at the network and transport layers of the TCP/IP stack. This component is also responsible for the calls to the available callout functions during the classification process. The filter engine contains maximum of 50 kernel-mode filtering layers.

The Base Filtering Engine (BFE) belongs to the user-mode component and coordinates the WFP components. The main operation performed by BFE as mentioned earlier is adding and removing filters from the system, storing filter configuration, and enforcing WFP of the security policy. The communication between the application and BFE is done through specific WFP management functions.

Callout drivers provide additional filtering functionality by adding custom callout functions to the filter engine at one or more of the kernel-mode filtering layers. Callouts support deep inspection and packet as well as stream modification. After a callout driver has added its callout functions to the filter engine, filters that specify a given driver's callout function can be added to the filtering process. Such filters can be added by either a user-mode management application or by the callout driver itself.

2.2 Filter Arbitration

When there are multiple filters acting on the same layer there is a need for filter arbitration in order to produce the proper action. Filter arbitration is the logic built into the WFP that is used to determine which filter will have the priority over another in the security policy when making network traffic filtering decisions.

The filter arbitration behavior is done over three rules: no network packet can escape a firewall filter, it passes through all configured filters; packets can be blocked by a Veto from some callout driver even if a filter with higher priority permits it; multiple filters can examine the network traffic at a same layer.

Each filter layer is divided into sub-layers ordered by priority which can be done by attaching weight to a particular layer. Network traffic passes through the sub-layers from the highest priority to the lowest priority. Within each sub-layer, filters are ordered also by weight. Network traffic is indicated to matching filters from highest weight to lowest weight. As we mentioned earlier the filter arbitration algorithm is applied to all sub-layers within a layer and depending on the decision after the packet passes all the filters, a decision is made by this algorithm. This is very suitable if we want to make multiple filters for the same network traffic.

On Fig. 2 is shown a sample diagram of traffic flow through the arbitration algorithm.

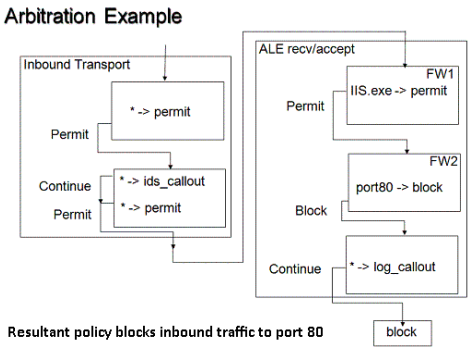

Fig. 2. Filter arbitration example, courtesy of Microsoft Corp.

In this example the firewall policy is intended to block all inbound traffic coming to port 80.

3 Overview of the WFP functionality

Overall the Windows Filtering Platform represents a convenient way to make a security solution based on the operating systems core functions and thus providing the appropriate protection to the users. The fact that this machine directly interacts with the OS networking stack almost on all networking levels eliminates the need for developing generic drivers for the network interface cards that extract the raw network packets.

By introduction of the callout drivers in the WFP architecture opens the possibility for development of a third-party security tools, like intrusion detection systems (IDS), post configuration for secure communication components, network flow monitoring tools, "sniffers" processing on different levels of a network communication, etc. This is also possible due to a good arbitration logic provided by this platform.

Worth mentioning is also the application and the user logic built in the WFP. It provides the means of network traffic filtering based per application or by a specific logged on user. All the filters available through this platform APIs can exist during the boot-up process and after the user profile is up and running they can switch operating in user mode. This transition is atomic and very fast so the local machine is never operating unprotected.

It should be noted that all of this functionality is available in the latest operating systems starting from Windows Vista desktop editions and the Windows Server 2008 family editions. From the developers' point of view, it is important that the API is available only in the C++ programming language [3].

4 QualysGuard solution

In this section we will explore the QualysGuard solution for vulnerability and malware scanning. This is one of the best security scanning solutions on today's market. It searches for open ports on the target machine and tests if its exploitable with all known vulnerabilities and exploits. The scanner will examine the sample application build upon this platform.

The QualysGuard Security and Compliance Suite eliminates network auditing and compliance inefficiencies by leveraging organization's core IT security information. In one consolidated suite, groups with different responsibilities can utilize similar information for their specific needs. The QualysGuard Security and Compliance Suite automates the process of vulnerability management and policy compliance across the enterprise, providing network discovery and mapping, asset prioritization, vulnerability assessment reporting and remediation tracking according to business risk. Policy compliance features allow security managers to audit, enforce and document compliance with internal security policies and external regulations.

QualysGuard vulnerability management enables organizations to effectively manage their vulnerabilities and maintain control over their network security with centralized reports, verified remedies, and full remediation workflow capabilities with trouble tickets. It provides comprehensive reports on vulnerabilities including severity levels, time to fix estimates and impact on business and analysis on security issues.

It has the most up-to-date KnowledgeBase of vulnerability checks in the industry, and the solution comes with external and internal scanners that safely and accurately detect security vulnerabilities across the entire network. As an on demand service, new signatures are delivered weekly, giving users the ability to scan for the latest threats. QualysGuard's is extremely accurate in the scans with almost zero percent false positives, false negatives and host crashes [4].

On Fig. 3 is shown the QualysGuard vulnerability lifecycle.

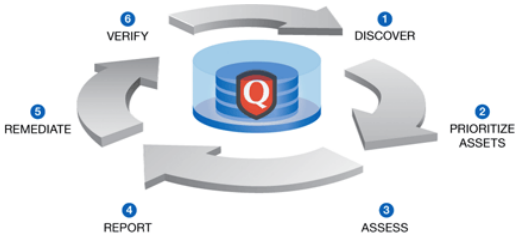

Fig. 3. QualysGuard Vulnerability Management Lifecycle

5 Analysis of WFP applications

In the next section we will present two firewall applications based on the Windows Filtering Platform. The first one is a firewall with two filters at a same sub-layer and the second one will be a demonstration of adding a rule on already started engine. All

the code is written in C++, due to the unavailability in the C# language (.Net framework).

5.1 A firewall application

This firewall application blocks all inbound traffic on the ports less than 10000. The only IP address that is allowed to connect to these ports is the loopback address 127.0.0.1, because it is needed by some processes as a part of the operating system.
First we need to register the firewall engine in some user defined dynamic session, as can be seen from the code snippet bellow.

```
FWPM_SESSION0     session;
RtlZeroMemory(&session, sizeof(FWPM_SESSION0));

session.displayData.name = L"FW Session";
session.displayData.description = L"Fltr";
session.flags = FWPM_SESSION_FLAG_DYNAMIC;

error =  FwpmEngineOpen0(NULL,RPC_C_AUTHN_WINNT,
NULL, &session,&engineHandle);
```

After we open the firewall engine for configuration, we must create a sub-layer in which we can add the filters for the network traffic. In our example we must add two filters to correspond to our security requirement mentioned above. The following code represents the filter that allows the loopback address to connect on all the ports.

```
Filter2.subLayerKey=SubLayer.subLayerKey;
Filter2.layerKey= FWPM_LAYER_INBOUND_TRANSPORT_V4;
Filter2.action.type= FWP_ACTION_PERMIT;
Filter2.weight.type=FWP_UINT8;
Filter2.weight.uint8=0x0F;
Filter2.filterCondition= &Condition2;
Filter2.numFilterConditions= 1;

Condition2.fieldKey=
FWPM_CONDITION_IP_LOCAL_ADDRESS;
Condition2.matchType= FWP_MATCH_EQUAL;

Condition2.conditionValue.type=FWP_V4_ADDR_MASK;
Condition2.conditionValue.v4AddrMask =
&AddrAndMask; /// local ip
```

The IP address must be defined as a Hexadecimal number without the dots. Also because we added two filters on the same layer, we must add priority to each filter (weight) in order for the filter arbitration to work properly and the final decision to be correct. From the code above we can also see that we set the firewall engine to process data from the transport layer of the network stack.

5.2 Adding rules

As stated by Microsoft Corp. the rule adding procedure should be very easy and very fast in terms of switching to the new security policy. We will add a rule to our previously mentioned example, which will allow traffic only from one specific address. Bellow is the code snippet.

```
Filter3.subLayerKey=SubLayer.subLayerKey;
Filter3.layerKey= FWPM_LAYER_INBOUND_TRANSPORT_V4;
Filter3.action.type= FWP_ACTION_PERMIT;
Filter3.weight.type=FWP_UINT8;
Filter3.weight.uint8=0x0A;
Filter3.filterCondition= &Condition3;
Filter3.numFilterConditions= 1;

Condition3.fieldKey=
FWPM_CONDITION_IP_REMOTE_ADDRESS;
Condition3.matchType= FWP_MATCH_EQUAL;

Condition3.conditionValue.type=FWP_V4_ADDR_MASK;
Condition3.conditionValue.v4AddrMask =
&AddrAndMask; /// remote ip
```

In order to add the new filter to an existing engine, we need to know the unique sub-layer key. Usually when installing or also known as "pushing" of the new policy, all the policy rules are deleted and rewritten with the new ones. This action should be atomic and done in some sort in a transaction process (in case something goes wrong). WFP API has support for both issues, which was tested and seems to be working, although it need some improvement if one should choose to make serious security solution on top of WFP.

5.3 Testing and analysis

Environment. The example applications were running on Windows 7 operating system, on virtual machine with one gigabyte of RAM and on virtual core 2 duo processor working on 2,20GHz. The native firewall application was turned off.
Tests. Two scans were done with the QualysGuard solution, one with the firewall turned on, and one with the firewall switched off. A third scan was done with a freeware non-commercial application called "NMap" for scanning open ports, in order to have two separate scan results [5]. During the scans the main factor that was observed was the machine performance including CPU usage and memory allocation.
The QualysGuard scan profile was set to scan all tcp and udp ports, thus performing a three-way handshake at the process of establishing a tcp connection. This inspection is also known as very intrusive and may cause a denial of service to some services, so it's recommended to be used with extreme caution. The NMap profile was set to scan all the ports, because it is only a port scanner. It lacks the ability for vulnerability

scans unlike the QualysGuard, but in the firewall application testing scenario, vulnerability scans are not necessary.

Analysis. As mentioned earlier, during the scans, the hardware performance of the machine was monitored, because these tests are known to be very "cruel" to the targets, resulting in their total unresponsiveness. There were CPU usage peaks to maximum of 70% of 10 to 15 seconds period of time. This is not very high, taking into account the scanning profile, particularly the QualysGuard one, and the fact that the firewall was working with virtual hardware.

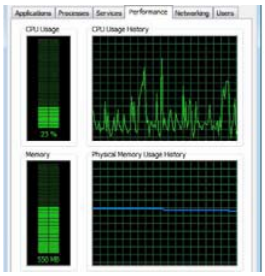

Fig. 4. System performance during the scan process

Memory performance was constant during the whole scanning process. Also all other processes that were running on the target machine did not show any unresponsiveness. Taking into account previously mentioned, the firewall application took the "beating" pretty well. This is one of the major requirements for a firewall application: reliability and availability.

The first scan taken with QualysGuard at the end although showed that the target machine was listening on two tcp ports, it cannot scan for vulnerabilities on these ports.

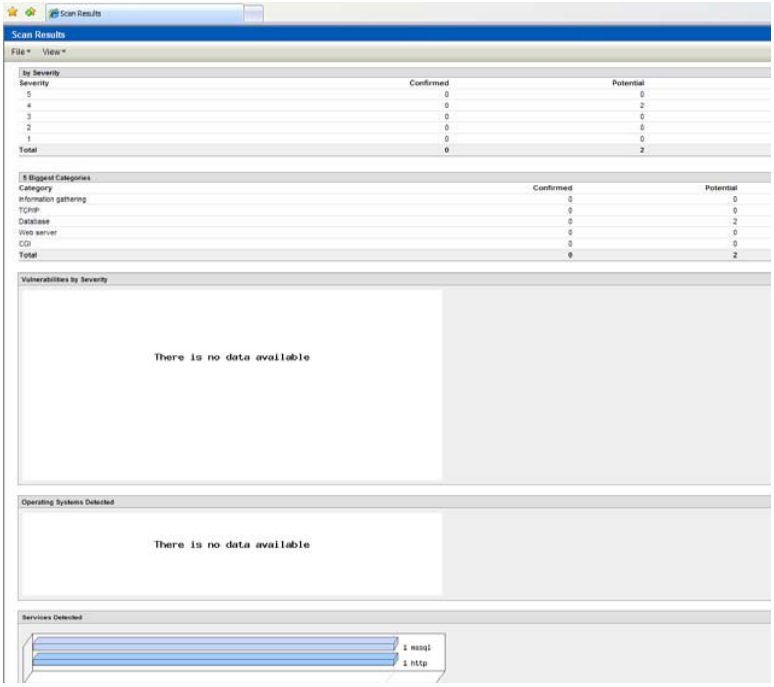

Fig. 5. Scan results from QualysGuard scan

This is mainly because the QualysGuard solution works on probing with TCP RST packets i.e. sends a the first packet for opening a tcp session and goes quite waiting for reset packet from the target machine in order to see if it's listening on that port. Because of the firewall it cannot make a tcp connection and make the necessary vulnerability tests. This port scanning can be block by a callout filter that makes a silent drop at the transport layer using the WFP engine. Blocking of UDP port scanning is done by blocking ICMP destination unreachable messages [6].

The NMap port scanning solution did not find any open ports bellow 10000 when the firewall was turned on.

Adding a rule (deleting rules and installing the new ones) is an atomic procedure, like noted by the platform vendor. The new security policy immediately (bellow one second) was put into operation.

After switching off the firewall the scan taken with the QualysGuard reported a lot of information about open ports and the vulnerabilities on services running on these ports.

6 Conclusion

The Windows Filtering Platform offers solid ground for security software developers in order to quickly produce stable and reliable firewall software. This new technology has proven to have the key requirements needed for serious security solution like: stability, robust engine, easy usage, configurability and option diversibility. Also one of the biggest advantages of this platform is the fact that the platform is available on every new operating system by Microsoft. This means that a piece of software can turn almost any machine in a powerful firewall. Worth mentioning is the support of third-party applications (IDS, VPNs, etc.) through the callout functions.

There are still some disadvantages like: the inability of ARP request processing, some minor bugs in the API functions and the most important one is maybe that until this moment this great platform is not available in the .Net framework.

The platform is in its constant development and even there is announcement that soon will be included in the newest .Net framework. Finally, all these disadvantages are thrown in shadow by the fact that firewall developers will never need to write generic network interface drivers in order to process data for a common application like a firewall.

References

1. Roger A., Grimes, Jesper M. Johansson: Windows Vista Security: Securing Vista Against Malicious Attacks (2007)
2. Microsoft Corporation, online library, http://msdn.microsoft.com
3. Mark E. Russinovich, David A. Solomon with Alex Ionescu: Windows Internals 5th Edition (PRO-Developer) (2009)
4. NMap tool official site, http://nmap.org
5. QualysGuard solution official site, http://www.qualys.com
6. Jan Kanclirz Jr.: How to Cheat at Microsoft Vista Administration (2007)