Windows Filtering Platform (WFP)

# What is it?

Tập hợp các dịch vụ hệ thống và API cho phép ứng dụng có thể xử lý và tuần tự lọc các gói dữ liệu truyền trên mạng. Nó cung cấp các tính năng như để giao tiếp tích hợp và có thể được configured để gọi các xử lý logic từ các ứng dụng. Nó chủ yếu được sử dụng bởi firewall và các thành phần xử lý gói hoặc kiểm soát kết nối của ứng dụng. WFP có từ phiên bản Windows Vista.

WPF cho phép người phát triển ứng dụng có thể tương tác với việc xử lý các gói dữ liệu ở nhiều tầng khác nhau trong network stack của hệ điều hành. Các dữ liệu có thể được lọc và thay đổi trước khi đến được đích.

Các công nghệ lọc gói trước WFP: Transport Driver Interface (TDI) filters, Network Driver Interface Specification (NDIS) filters, and Winsock Layered Service Providers (LSP). Trong đó WFP cung cấp 1 platform đơn giản hơn để phát triển. Windows Filtering Platform is a development platform and not a firewall itself.

WFP có thể hỗ trợ người phát triền hiện thực firewalls, IDS, antivirus, công cụ kiểm soát mạng, …. Nó cũng cung cấp các nền tảng để quản lý chính sách IPsec, thay đổi các cảnh báo, network diagnostics, ….

# Requirements

The Windows Filtering Platform is supported on servers running Windows Server 2008 or Windows Server 2008 R2 and on clients running Windows Vista or Windows 7.

# Benefits and usage

|  |  |
| --- | --- |
| • | You have a fine level of access control to the TCP/IP packet processing path. This control differs from the filter and firewall hook methods that are supported in Windows XP and Windows Server 2003, which provide limited access to the TCP/IP processing path. |
| • | Because WFP already provides a filtering engine, you are not required to build your own filtering logic and engine. You can just tap into the WFP filtering engine and concentrate on the value that your component adds. |
| • | When you use WFP, there is much less risk of affecting your component with a future service pack release. |
| • | Implementing a firewall or packet filtering value-added solution is easier because the filtering logic and the hooks into the various layers of the TCP/IP protocol already exist. |
| • | Depending on the filtering and processing needs of your component, you might be able to move your component from kernel mode into user mode, which makes it easier to develop your component. Additionally, a crash in the user-mode component does not affect the entire Windows system. |
| • | Because all the applications and services use the same filtering engine, it is easier to determine whether other applications or services exist that perform the same function. |
| • | It is easier to develop filtering solutions that can coexist with other WFP-based filtering solutions, all of which follow WFP filter arbitration rules. |

You should use WFP in the following situations:

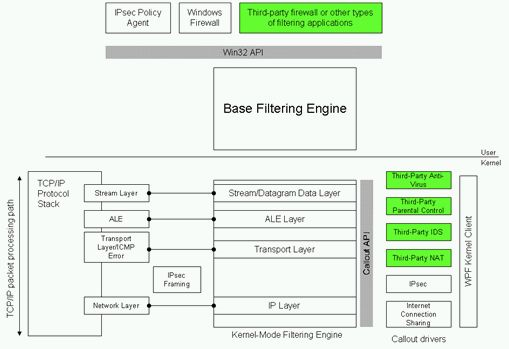
|  |  |
| --- | --- |
| • | Your component must examine TCP/IP traffic at a specific layer of the TCP/IP protocol stack. |
| • | Your component works with IPsec-protected traffic. |
| • | You want to perform packet processing after IPsec decryption. |
| • | You want to use the built-in IPv6 filtering engine to perform IPv6 packet filtering, rather than building an IPv6 filtering engine yourself. |

# Architecture

The WFP API consists of a user-mode API and a kernel-mode API.

Phân thành các thành phần chính:

* **Win32 API**
* **Base Filtering Engine**: user mode service. It implements the filter requests that user-mode filtering applications make by adding filters to the Kernel-Mode Filtering Engine. It accepts filtering rules, and enforces the security model of the application. It also maintains statistics for the WFP and logs its state.
* **Shims**, which exposes the internal structure of a packet as properties. Different shims exist for protocols at different layers. The filtering engine filters the packets by verifying the data against the specified set of rules. WFP comes with a set of shims, shims for other protocols can be registered using the API.
* **Kernel-Mode Filtering Engine:** stores the filters that filtering applications create through the Base Filtering Engine and interacts through filtering layers with the TCP/IP stack and the set of installed callout drivers.
* **Callout drivers:** a callback function exposed by a filtering driver. The filtering drivers are used to provide filtering capabilities other than the default block/allow. During registration of a filter rule, the callout function is specified. When the filter is matched, the callout is invoked which handles what needs to be done. Used when simple packet filtering - checking the packet against the predefined WFP filtering conditions to determine whether the packet should be permitted or dropped - is not enough. To perform deep inspection of packet contents or data modification, you must have callout drivers.



# WFP sample project links:

<http://blogs.msdn.com/onoj/archive/2007/05/09/windows-filtering-platform-sample.aspx>

<http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=1637&SiteID=1>

<http://msdn.microsoft.com/en-us/library/aa366504.aspx>

IPTables

# What is it?

IPTables là một chương trình ứng dụng( có thể được xem là một firewall ) cho phép người quản trị hệ thống có thể kiểm soát được các packet tương tác với hệ thống. Chương trình này cần phải có quyền root của hệ thống mới có thể thực hiện được. Chương trình này được viết bằng ngôn ngữ C, chạy trên hệ điều hành Linux.

- Bất kỳ Packet nào muốn đi vào PC của bạn đều phải đi qua **Input Chain.**

- Bất cứ Packet nào từ PC của bạn muốn đi ra ngoài **Network** đều phải đi qua **Output Chain**.

- Bất cứ Packet nào mà PC của bạn muốn gửi đi một **Destination** khác đều phải đi qua **Forward Chain**

Incoming / \ Outgoing

-->[Routing ]--->|FORWARD|------->

[Decision] \\_\_\_\_\_/ ^

| |

v \_\_\_\_

\_\_\_ / \

/ \ |OUTPUT|

|INPUT| \\_\_\_\_/

\\_\_\_/ ^

| |

----> Local Process ----

Tất cả những điều trên đều được quản lý bởi iptables, ta có thể thiết lập các rules cho các chains.

Usage

(by **N.X.Bi O==(=========> ^($)^ Administrator - Moderator Of HKCVN**)

Link: <http://antihacker.50webs.com/ky%20thuat%20hack/Iptables.htm>

**-s** là tuỳ chọn để ngăn chặn một địa chỉ IP hay DNS nguồn. Ta có dòng lệnh:

***iptables -s 192.78.4.0***

Nếu bạn muốn xử lý các Packet một cách chi tiết hơn. Thì tuỳ chọn **-j** sẽ giúp bạn thực hiện điều đó như: **ACCEPT**, **DENY** hay **DROP**

***iptables -s 192.78.4.0 -j DROP***

Chúng ta còn có thể bỏ qua một PC nhất định trên một mạng. Nếu bạn không muốn những PC trong mạng liên lạc và nói chuyện với PC đó hay liên lạc ra ngoài. Bạn chỉ cần thay đổi tham số **Input**, **Output** và thay đổi tuỳ chọn **-s**, **-d**

Nếu chúng ta muốn bỏ qua yêu cầu phản hồi Telnet từ máy PC này. Trong trường hợp này có ít nhất 3 giao thức có thể được chỉ rõ: **TCP, UDP** và **ICMP**.

Tuỳ chọn **-p** được sử dụng để chỉ rõ chi tiết giao thức cần xử lý. Telnet là một giao thức hoạt động trên Port 23/TCP lên chúng ta sẽ có dòng lệnh:

**iptables -A INPUT -s 192.78.4.0 -p tcp --80 telnet -j DROP**

Các Command trên là thao tác cho 1 địa chỉ IP (**Single IP**). Nếu bạn muốn thao tác với nhiều địa chỉ IP cùng một lúc (**Multi IP**) thì sẽ có chút thay đổi nhỏ như sau:

- **192.78.4.0/84** = = > Tất các các IP từ **192.78.4.0** cho đến **192.78.4.84**

**- 192.78.4.\*** = = > Tất cả các IP thuộc lớp mạng D. Từ **192.78.4.0** cho đến **192.78.4.255**

Hướng dẫn sử dụng Iptables trên Ubuntu: <https://help.ubuntu.com/community/IptablesHowTo>

Note: The iptables tool inserts and deletes rules from the kernel's packet filtering table. This means that whatever you set up, it will be lost upon reboot. You can try the iptables-save and iptables-restore scripts to save them to, and restore them from a file.