WebKilit

Installation & Configuration Guide
Version 2.2

Document Revision 2.0

https://www.kaplansoft.com/

WebKilit is built by Yasin KAPLAN

Read "Readme.txt" for last minute changes and updates which can be found under application directory.

Table of Contents

Introduction

WebKilit is an IP content filter and redirector and a HTTP interface for Windows Firewall *(Vista/7/8, 2003/2008/2012 Server)*. Major features;

- Simple design and easy to use user interface.
- Simple interface for user definitions.
- Real time monitoring of connected users.
- Real time monitoring HTTP connection from local network to the Internet and HTTP access logging.
- Redirection of blocked URLs.
- Blocking files based on their extensions.
- Support for TZSP *(TaZmen Sniffer Protocol)*.
- Customizable HTTP interface

WebKilit consists of a GUI and a service application. WebKilit has its own built-in HTTP server.

You can provide username based access through HTTP interface. If you have remote users with dynamic IP addresses, firewall configuration is a major issue. You just need to open a single port for HTTP interface of WebKilit in Windows Firewall. WebKilit will enable firewall rules in user profile, when user is authenticated by the HTTP interface. You can specify a session duration for every user. WebKilit will disable all rules when the session timeout expires.

System Requirements

- A Windows system with at least 2 GBytes of RAM.
- Microsoft.NET Framework v4.0 Client Profile *(Min.).*
- Administrative privileges.
- Replication of LAN traffic to the Ethernet port of WebKilit installed machine in an Ethernet switching environment for HTTP URL blocking. You need to have a secondary LAN connection the then network if your switch does not accept incoming traffic from the monitoring port.

Installation

Unzip “WebKilit.zip” and click “Setup.exe” comes with the distribution. Follow the instruction of setup wizard. Setup will install WebKilit Manager and WebKilit Service, add a shortcut for WebKilit Manager to desktop and the start menu.

Configuration

Run WebKilit Manager from Start Menu / Program Files / WebKilit. WebKilit automatically configures itself at first run.

Settings Tab

Click Settings Tab to start configuration. Enter following information:

- **HTTP Port:** Set HTTP port for login and redirection form.
- **TLS Enabled:** You can use HTTPS for login form when you enable TLS. You must select a server certificate after enabling TLS. You also need to enable TLS in order to block access to SSL secured web sites.
- **Logging / Startup:** Select logging level of WebKilit. Select "None" if you do not want logging, select "Errors" to log errors and select "Sessions" to log session information and errors. Log files are located under <Application Directory>\Logs directory. Set WebKilit service startup mode, Manual or Automatic. You can also disable service startup.
- **Server Certificate:** Select server certificate for TLS.
- **Root Directory**: You can set directory where alternative redirection, login, info and error message html resource files reside. Please see User Defined Login / Redirection Interface section of this manual.

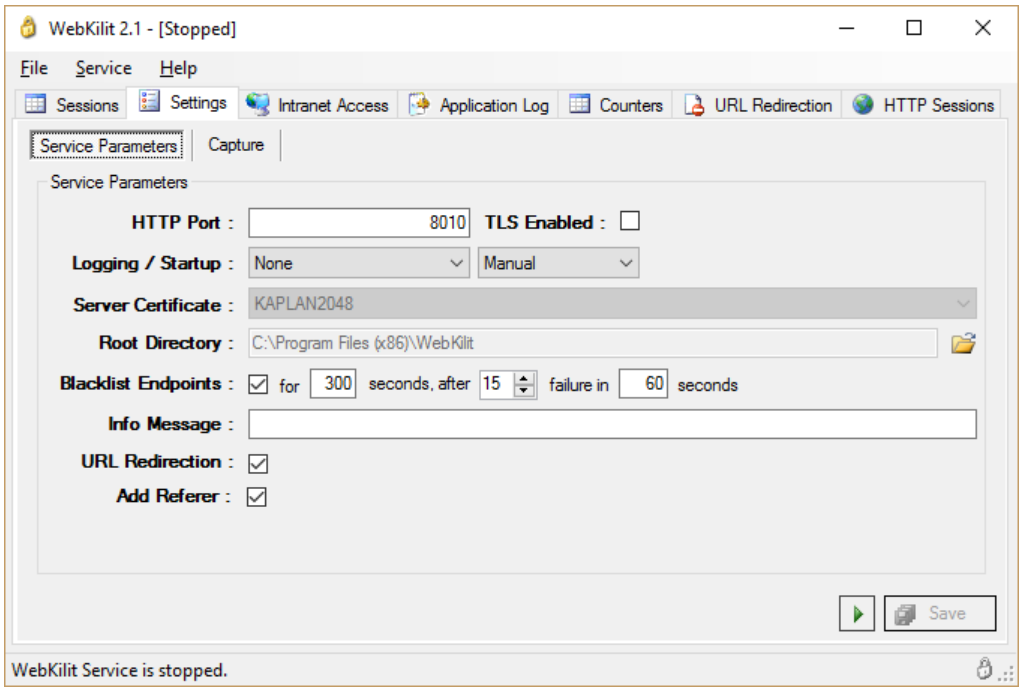

Figure - 1. WebKilit Settings / Service Parameters tab

- **Blacklist IP Endpoints:** If selected, WebKilit monitors failed login attempts from suspicious endpoints and blacklists them.
- **Info Message:** You can define a message can be displayed when user logged in.
- **URL Redirection:** WebKilit can redirect HTTP requests based on URL redirection polices specified in URL redirection tab. Click to enable URL redirection.

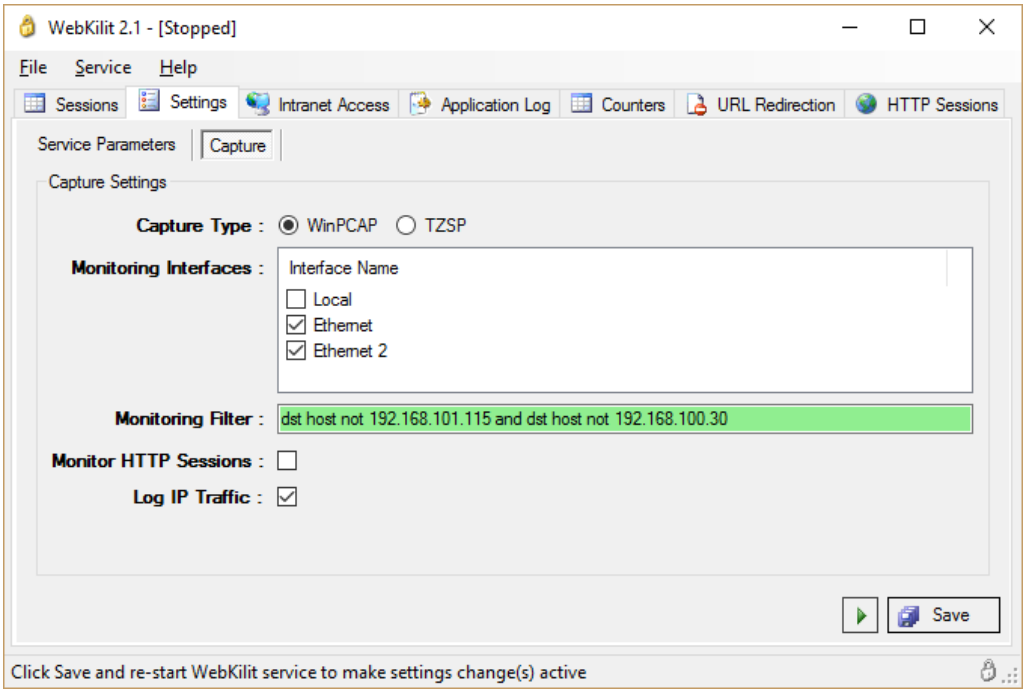

Figure - 2. WebKilit Settings / Capture tab

- **Capture Type:** Select capture type. WebKilit uses UDP port 37008 for incoming TZSP[1] packets. Please set following filter in Mikrotik for proper operation when you use TZSP;

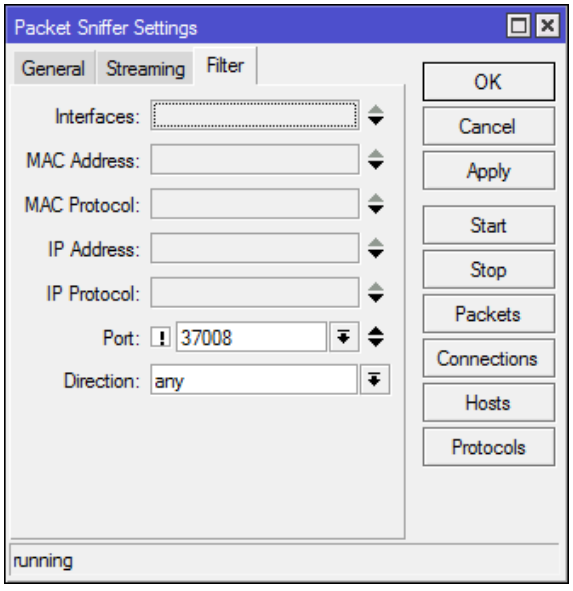

Figure - 3. Mikrotik Packet Sniffer Settings

- **Monitoring Interfaces:** Select monitoring interfaces for redirection HTTP requests from Intranet to the Internet. You need to mirror traffic in your LANs to these ports.
- **Monitoring Filter:** Please see Monitoring Filter section of this manual.
- **Monitor HTTP Sessions:** You can monitor HTTP sessions in real time through WebKilit GUI when this option is enabled.
- **Log IP Traffic:** You can log captured IP traffic in daily rotated log files.

[1] Please see https://en.wikipedia.org/wiki/TZSP for more information.

Intranet Access

You can define users to be allowed to Intranet from the Internet in “Intranet Access” tab. Enter a username in the bottom leftmost textbox, enter the password to the textbox at the right of the username entry and a session duration in seconds. You can add allowed firewall rules in the user profile.

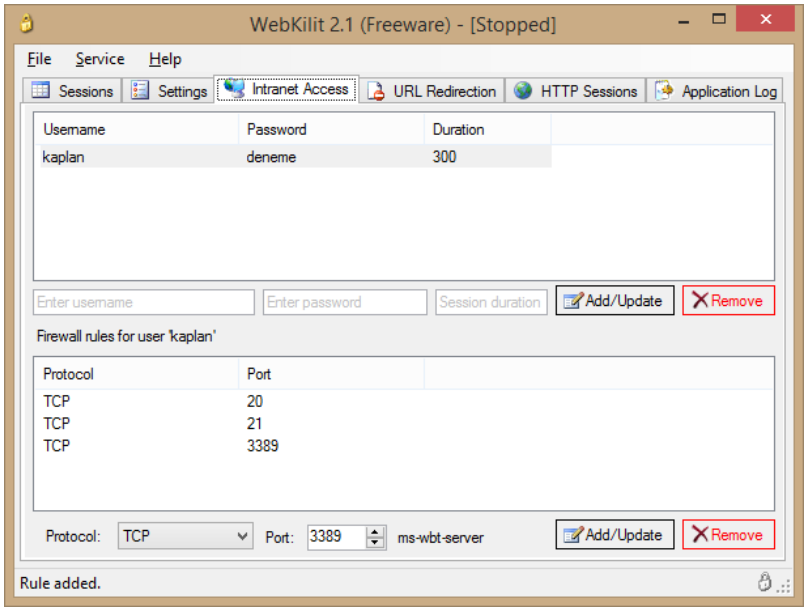

Figure - 4. Intranet Access Tab

WebKilit login form can be accessible through http://<System IP address>:<HTTP port>/tslogin

If your IP address is set 192.168.1.1 and HTTP Port is 8080 then URL will be http://192.168.1.1:8080/tslogin

URL Redirection

You can specify URL redirections in “URL Redirection” tab. WebKilit will redirect HTTP requests to HTTP URLs in the list to the redirection URL configured. You can batch import URLs in a text file. You can also specify application extension to filter like exe.

You can specify source host/subnet for a redirection entry. All HTTP requests from specified host/subnet will be redirected to redirection URL if you set Default as URL. You can also set redirection URL to WebKilit served URL if you set redirection URL to Default.

WebKilit sends artificial Ethernet packets to provide URL redirection to the LAN. You need to have two separate Ethernet connection to the LAN if your Ethernet switch does not allow incoming Ethernet packets from the monitoring port. WebKilit gets user traffic from the monitoring port and injects artificial packets from the other LAN port.

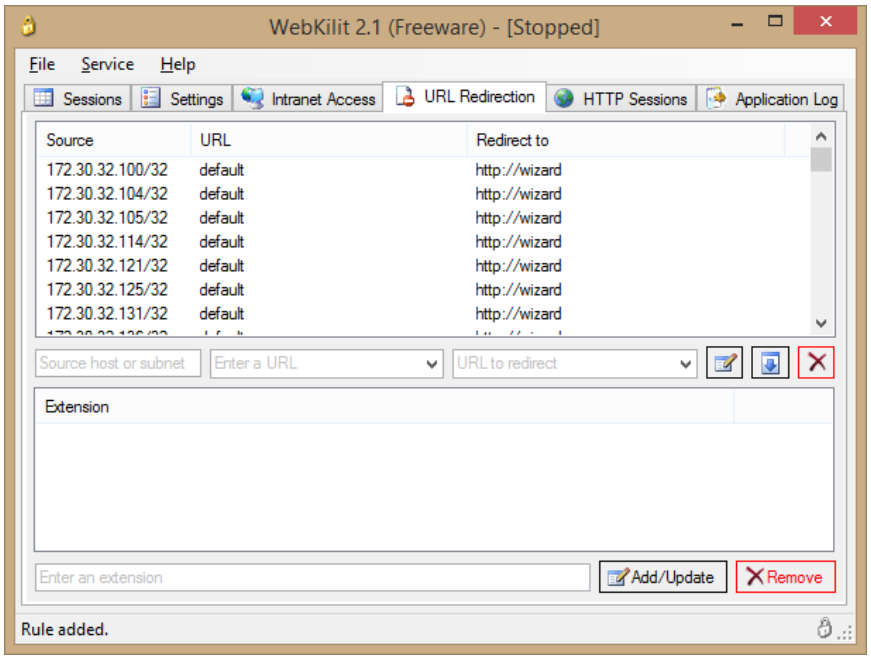

Figure - 5. URL Redirection Tab

HTTP Sessions

You can monitor HTTP connections from Intranet to the Internet in “HTTP Sessions” tab.

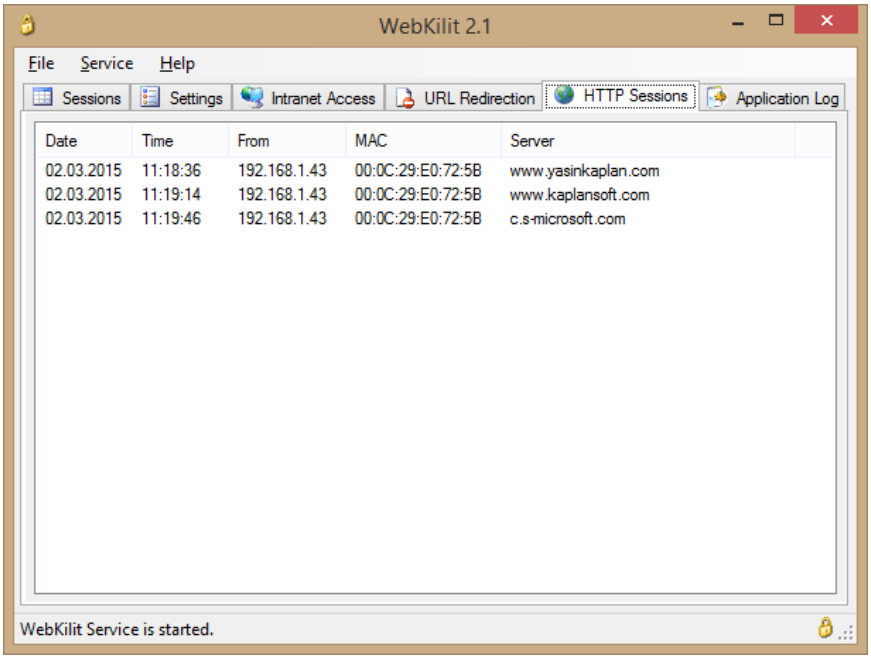

Figure - 6. HTTP Sessions

WebKilit automatically adds detected HTTP sessions to HTTP Sessions list and clear entry when users disconnects from the HTTP server.

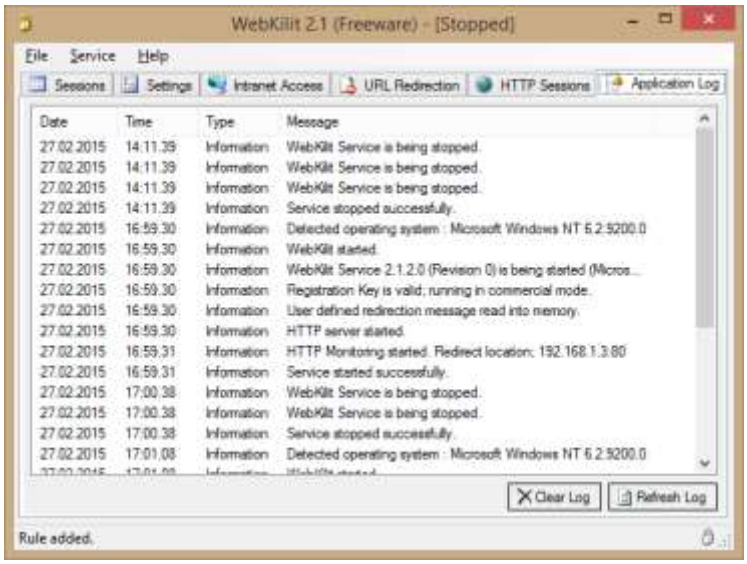

Figure - 7. Application Log Tab

Application Log

You can monitor system events in Application Log tab. You can manually refresh log entries and clear log entries. Click Enable Auto Refresh option to refresh log list every seconds.

User Defined Redirection Interface

You can specify a custom redirection interface for blocked URLs. You can create custom redirected.html file and place into root directory. WebKilit comes with a default redirection interface;

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <title>WebKilit HTTP Server - Access Blocked</title>
    <meta http-equiv="refresh" content="60" />
    <style type="text/css" media="all">
        body
        {
            font: 0.8em arial, helvetica, sans-serif;
        }

        .info
        {
            -moz-border-radius: 8px;
            -webkit-border-radius: 8px;
            background-image: url('info.gif');
            background-repeat: no-repeat;
            background-position: 20px 15px;
            padding: 20px 20px 20px 85px;
            margin: 0px 15px 20px 15px;
            background-color: #FF0000;
            border: 2px solid #3F7F47;
            line-height: normal;
            width: 340px;
            height: 32px;
        }
        .style1
        {
            font-size: xx-large;
        }
    </style>
</head>
<body>
    <div class="info" style="color: #FFFFFF">
        <b class="style1">Access Blocked</b>
    </div>
</body>
</html>
```

Figure - 8. Default redirection form

User Defined Login Interface

You can use your own defined login, info and error pages. WebKilit uses built in html resources for this pages. WebKilit looks for alternative login.html, info.html and error.html files. If any of them found, WebKilit uses user defined html file. Login.html must have following form and form objects;

```
<form name="LoginForm" method="post" action="tslogin" id="WebKilitLoginForm">
<input name="Username" type="Text" id="Username">
<input name="Password" type="Password" id="Password">
<input type="submit" name="Login" value="Login" id="Login">
```

You can display user's connection date and remained credit by adding `%ipaddress%`, `%connected%`, `%infomessage%` and `%remained%` variables to info.html. WebKilit will replace real values of these variables prior to send html response. You can display WebKilit generated error message in error.html file using `%error%` variable. Here is a sample login.html;

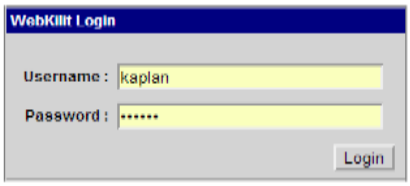

Figure - 9. Default login form

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>WebKilit Login</title>
</head>
<body style="font-size: 12px; font-family: Arial">
    <form name="LoginForm" method="post" action="tslogin" id="WebKilitLoginForm">
       <table id="FormTable" cellspacing="0" cellpadding="3" rules="all" border="3"
            style="border-color: #F0F0F0; width: 265px; border-style: solid;">
            <tr style="color: White; background-color: Navy;">
                <td colspan="2" style="border-color: #F0F0F0">
                    <b>WebKilit Login %message%</b></td>
            </tr>
            <tr style="background-color: #F0F0F0; font-size: Smaller; height: 6px;">
                <td colspan="2" style="border-color: #F0F0F0; border-style: solid;">
                    <font size="1px"> </font></td>
            </tr>
            <tr style="background-color: #F0F0F0;">
                <td align="right" style="border-color: #F0F0F0; border-style: solid;">
                    <b>Username : </b>
                </td>
                <td style="border-color: #F0F0F0; border-style: solid;">
                    <input name="Username" type="Text" id="Username" style="width: 150px;" /></td>
            </tr>
            <tr style="background-color: #F0F0F0;">
                <td align="right" style="border-color: #F0F0F0; border-style: solid;">
                    <b>Password : </b>
                </td>
                <td style="border-color: #F0F0F0; border-style: solid;">
                    <input name="Password" type="Password" id="Password" style="width: 150px;" /></td>
            </tr>
            <tr style="background-color: #F0F0F0;">
                <td style="border-color: #F0F0F0; border-style: solid;">
                     </td>
                <td align="right" style="border-color: #F0F0F0; border-style: solid;">
                    <input type="submit" name="Login" value="Login" id="Login" style="background-color:
#F0F0F0;" /> </td>
                </tr>
        </table>
    </form>
    <br />
</body>
```

```
</html>
```

Figure - 10. Info display after login

Starting WebKilit

Click “Service” menu and select “Start” to run WebKilit after making necessary configuration and saving configuration. If service starts successfully you will see “WebKilit is started” message at bottom left message section of WebKilit. Optionally you can start/stop WebKilit using the button on Settings tab. When you make any change(s) in configuration, WebKilit will be restarted when you save the settings.

If WebKilit cannot start please examine Application Log tab as well as WebKilit log file under <Application Directory>\Logs if you were enabled logging in Settings tab.

Monitoring Filters[2]

WebKilit filters are based on a declarative predicate syntax. A filter is an ASCII string containing a filtering *expression*. The expression selects which packets will be captured. If no expression is given, all packets on the net will be accepted by the kernel-level filtering engine. Otherwise, only packets for which *expression* is ‘true’ will be accepted.

The *expression* consists of one or more *primitives*. Primitives usually consist of an *id* (name or number) preceded by one or more qualifiers. There are three different kinds of qualifier:

type
: qualifiers say what kind of thing the id name or number refers to. Possible types are **host**, **net** and **port**. E.g., `host foo', `net 128.3', `port 20'. If there is no type qualifier, **host** is assumed.

dir
: qualifiers specify a particular transfer direction to and/or from *id*. Possible directions are **src**, **dst**, **src or dst** and **src and dst**. E.g., `src foo', `dst net 128.3', `src or dst port ftp-data'. If there is no dir qualifier, **src or dst** is assumed. For `null' link layers (i.e. point to point protocols such as slip) the **inbound** and **outbound** qualifiers can be used to specify a desired direction.

proto
: qualifiers restrict the match to a particular protocol. Possible protos are: **ether**, **fddi**, **tr**, **ip**, **ip6**, **arp**, **rarp**, **decnet**, **tcp** and **udp**. E.g., `ether src foo', `arp net

[2] This section has been drawn from the tcpdump man page. The original version can be found at http://www.tcpdump.org/manpages/pcap-filter.7.html.

128.3', `tcp port 21'. If there is no proto qualifier, all protocols consistent with the type are assumed. E.g., `src foo' means `(ip or arp or rarp) src foo' (except the latter is not legal syntax), `net bar' means `(ip or arp or rarp) net bar' and `port 53' means `(tcp or udp) port 53'.

[`fddi' is actually an alias for `ether'; the parser treats them identically as meaning ``the data link level used on the specified network interface.'' FDDI headers contain Ethernet-like source and destination addresses, and often contain Ethernet-like packet types, so you can filter on these FDDI fields just as with the analogous Ethernet fields. FDDI headers also contain other fields, but you cannot name them explicitly in a filter expression.

Similarly, `tr' is an alias for `ether'; the previous paragraph's statements about FDDI headers also apply to Token Ring headers.]

In addition to the above, there are some special `primitive' keywords that don't follow the pattern: **gateway**, **broadcast**, **less**, **greater** and arithmetic expressions. All of these are described below.

More complex filter expressions are built up by using the words **and**, **or** and **not** to combine primitives. E.g., `host foo and not port ftp and not port ftp-data'. To save typing, identical qualifier lists can be omitted. E.g., `tcp dst port ftp or ftp-data or domain' is exactly the same as `tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain'.

Allowable primitives are:

dst host *host*
: True if the IPv4/v6 destination field of the packet is *host*, which may be either an address or a name.

src host *host*
: True if the IPv4/v6 source field of the packet is *host*.

host *host*
: True if either the IPv4/v6 source or destination of the packet is *host*. Any of the above host expressions can be prepended with the keywords, **ip**, **arp**, **rarp**, or **ip6** as in:
ip host *host*
which is equivalent to:
ether proto *\ip* **and host** *host*
If *host* is a name with multiple IP addresses, each address will be checked for a match.

ether dst *ehost*
: True if the ethernet destination address is *ehost*. *Ehost* may be either a name from /etc/ethers or a number (see *ethers*(3N) for numeric format).

ether src *ehost*
: True if the ethernet source address is *ehost*.

ether host *ehost*
: True if either the ethernet source or destination address is *ehost*.

gateway *host*
: True if the packet used *host* as a gateway. I.e., the ethernet source or destination address was *host* but neither the IP source nor the IP destination was *host*. *Host* must be a name and must be found both by the machine's host-name-to-IP-address resolution mechanisms (host name file, DNS, NIS, etc.) and by the machine's host-name-to-Ethernet-address resolution mechanism (/etc/ethers, etc.). (An equivalent expression is
ether host *ehost* **and not host** *host*

which can be used with either names or numbers for *host / ehost*.) This syntax does not work in IPv6-enabled configuration at this moment.

dst net *net*

True if the IPv4/v6 destination address of the packet has a network number of *net*. *Net* may be either a name from /etc/networks or a network number (see *networks(4)* for details).

src net *net*

True if the IPv4/v6 source address of the packet has a network number of *net*.

net *net*

True if either the IPv4/v6 source or destination address of the packet has a network number of *net*.

net *net* **mask** *netmask*

True if the IP address matches *net* with the specific *netmask*. May be qualified with **src** or **dst**. Note that this syntax is not valid for IPv6 *net*.

net *net*/*len*

True if the IPv4/v6 address matches *net* with a netmask *len* bits wide. May be qualified with **src** or **dst**.

dst port *port*

True if the packet is ip/tcp, ip/udp, ip6/tcp or ip6/udp and has a destination port value of *port*. The *port* can be a number or a name used in /etc/services (see *tcp*(4P) and *udp*(4P)). If a name is used, both the port number and protocol are checked. If a number or ambiguous name is used, only the port number is checked (e.g., **dst port 513** will print both tcp/login traffic and udp/who traffic, and **port domain** will print both tcp/domain and udp/domain traffic).

src port *port*

True if the packet has a source port value of *port*.

port *port*

True if either the source or destination port of the packet is *port*. Any of the above port expressions can be prepended with the keywords, **tcp** or **udp**, as in:

```
tcp src port port
```

which matches only tcp packets whose source port is *port*.

less *length*

True if the packet has a length less than or equal to *length*. This is equivalent to:

```
len <= length.
```

greater *length*

True if the packet has a length greater than or equal to *length*. This is equivalent to:

```
len >= length.
```

ip proto *protocol*

True if the packet is an IP packet (see *ip*(4P)) of protocol type *protocol*. *Protocol* can be a number or one of the names *icmp*, *icmp6*, *igmp*, *igrp*, *pim*, *ah*, *esp*, *vrrp*, *udp*, or *tcp*. Note that the identifiers *tcp*, *udp*, and *icmp* are also keywords and must be escaped via backslash (\), which is \\ in the C-shell. Note that this primitive does not chase the protocol header chain.

ip6 proto *protocol*

True if the packet is an IPv6 packet of protocol type *protocol*. Note that this primitive does not chase the protocol header chain.

ip6 protochain *protocol*

True if the packet is IPv6 packet, and contains protocol header with type *protocol* in its protocol header chain. For example,

```
ip6 protochain 6
```

matches any IPv6 packet with TCP protocol header in the protocol header chain. The packet may contain, for example, authentication header, routing header, or hop-by-hop option header, between IPv6 header and TCP header. The BPF code emitted by this primitive is

complex and cannot be optimized by BPF optimizer code in *tcpdump*, so this can be somewhat slow.

ip protochain *protocol*

Equivalent to **ip6 protochain** *protocol*, but this is for IPv4.

ether broadcast

True if the packet is an ethernet broadcast packet. The *ether* keyword is optional.

ip broadcast

True if the packet is an IP broadcast packet. It checks for both the all-zeroes and all-ones broadcast conventions, and looks up the local subnet mask.

ether multicast

True if the packet is an ethernet multicast packet. The *ether* keyword is optional. This is shorthand for **`ether[0] & 1 != 0**'.

ip multicast

True if the packet is an IP multicast packet.

ip6 multicast

True if the packet is an IPv6 multicast packet.

ether proto *protocol*

True if the packet is of ether type *protocol*. *Protocol* can be a number or one of the names *ip*, *ip6*, *arp*, *rarp*, *atalk*, *aarp*, *decnet*, *sca*, *lat*, *mopdl*, *moprc*, *iso*, *stp*, *ipx*, or *netbeui*. Note these identifiers are also keywords and must be escaped via backslash (\).

[In the case of FDDI (e.g., **`fddi protocol arp**') and Token Ring (e.g., **`tr protocol arp**'), for most of those protocols, the protocol identification comes from the 802.2 Logical Link Control (LLC) header, which is usually layered on top of the FDDI or Token Ring header.

When filtering for most protocol identifiers on FDDI or Token Ring, *tcpdump* checks only the protocol ID field of an LLC header in so-called SNAP format with an Organizational Unit Identifier (OUI) of 0x000000, for encapsulated Ethernet; it doesn't check whether the packet is in SNAP format with an OUI of 0x000000.

The exceptions are *iso*, for which it checks the DSAP (Destination Service Access Point) and SSAP (Source Service Access Point) fields of the LLC header, *stp* and *netbeui*, where it checks the DSAP of the LLC header, and*atalk*, where it checks for a SNAP-format packet with an OUI of 0x080007 and the Appletalk etype.

In the case of Ethernet, *tcpdump* checks the Ethernet type field for most of those protocols; the exceptions are *iso*, *sap*, and *netbeui*, for which it checks for an 802.3 frame and then checks the LLC header as it does for FDDI and Token Ring, *atalk*, where it checks both for the Appletalk etype in an Ethernet frame and for a SNAP-format packet as it does for FDDI and Token Ring, *aarp*, where it checks for the Appletalk ARP etype in either an Ethernet frame or an 802.2 SNAP frame with an OUI of 0x000000, and *ipx*, where it checks for the IPX etype in an Ethernet frame, the IPX DSAP in the LLC header, the 802.3 with no LLC header encapsulation of IPX, and the IPX etype in a SNAP frame.]

decnet src *host*

True if the DECNET source address is *host*, which may be an address of the form ``10.123'', or a DECNET host name. [DECNET host name support is only available on Ultrix systems that are configured to run DECNET.]

decnet dst *host*

True if the DECNET destination address is *host*.

decnet host *host*

True if either the DECNET source or destination address is *host*.

ip, **ip6**, **arp**, **rarp**, **atalk**, **aarp**, **decnet**, **iso**, **stp**, **ipx**, *netbeui*

Abbreviations for:

```
ether proto p
```

where *p* is one of the above protocols.

lat, **moprc**, **mopdl**

Abbreviations for:
`ether proto p`
where *p* is one of the above protocols. Note that *tcpdump* does not currently know how to parse these protocols.

vlan *[vlan_id]*
True if the packet is an IEEE 802.1Q VLAN packet. If *[vlan_id]* is specified, only true is the packet has the specified *vlan_id*. Note that the first **vlan** keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a VLAN packet.

tcp, **udp**, **icmp**
Abbreviations for:
`ip proto p or ip6 proto p`
where *p* is one of the above protocols.

iso proto *protocol*
True if the packet is an OSI packet of protocol type *protocol*. *Protocol* can be a number or one of the names *clnp*, *esis*, or *isis*.

clnp, **esis**, **isis**
Abbreviations for:
`iso proto p`
where *p* is one of the above protocols. Note that *tcpdump* does an incomplete job of parsing these protocols.

expr relop expr
True if the relation holds, where *relop* is one of >, <, >=, <=, =, !=, and *expr* is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators [+, -, *, /, &, |], a length operator, and special packet data accessors. To access data inside the packet, use the following syntax:
`proto [ expr : size ]`
Proto is one of **ether, fddi, tr, ip, arp, rarp, tcp, udp, icmp** or **ip6**, and indicates the protocol layer for the index operation. Note that *tcp, udp* and other upper-layer protocol types only apply to IPv4, not IPv6 (this will be fixed in the future). The byte offset, relative to the indicated protocol layer, is given by *expr*. *Size* is optional and indicates the number of bytes in the field of interest; it can be either one, two, or four, and defaults to one. The length operator, indicated by the keyword **len**, gives the length of the packet.

For example, `**ether[0] & 1 != 0**' catches all multicast traffic. The expression `**ip[0] & 0xf != 5**' catches all IP packets with options. The expression `**ip[6:2] & 0x1fff = 0**' catches only unfragmented datagrams and frag zero of fragmented datagrams. This check is implicitly applied to the **tcp** and **udp** index operations. For instance, **tcp[0]** always means the first byte of the TCP *header*, and never means the first byte of an intervening fragment.

Some offsets and field values may be expressed as names rather than as numeric values. The following protocol header field offsets are available: **icmptype** (ICMP type field), **icmpcode** (ICMP code field), and **tcpflags**(TCP flags field).

The following ICMP type field values are available: **icmp-echoreply**, **icmp-unreach**, **icmp-sourcequench**, **icmp-redirect**, **icmp-echo**, **icmp-routeradvert**, **icmp-routersolicit**, **icmp-timxceed**, **icmp-paramprob**, **icmp-tstamp**, **icmp-tstampreply**, **icmp-ireq**, **icmp-ireqreply**, **icmp-maskreq**, **icmp-maskreply**.

The following TCP flags field values are available: **tcp-fin**, **tcp-syn**, **tcp-rst**, **tcp-push**, **tcp-push**, **tcp-ack**, **tcp-urg**.

Primitives may be combined using:

A parenthesized group of primitives and operators (parentheses are special to the Shell and must be escaped).
Negation (`**!**' or `**not**').
Concatenation (`**&&**' or `**and**').
Alternation (`||' or `**or**').

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit **and** tokens, not juxtaposition, are now required for concatenation.

If an identifier is given without a keyword, the most recent keyword is assumed. For example,

```
not host vs and ace
```

is short for

```
not host vs and host ace
```

which should not be confused with

```
not ( host vs or ace )
```

Expression arguments can be passed to *tcpdump* as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

WebKilit Service Messages in WebKilit Logs

WebKilit Service x.y.0.0 (Revision 0) is being started (<Windows version>).

This message notifies that WebKilit is being started.

Registration Key is valid; running in commercial mode.

You see this message when a valid Registration.key file exists under WebKilit application directory.

User defined login form read into memory

You see this message when a valid user defined login form html file exists under Root Directory.