Advanced Packet Editor Manual

Intro

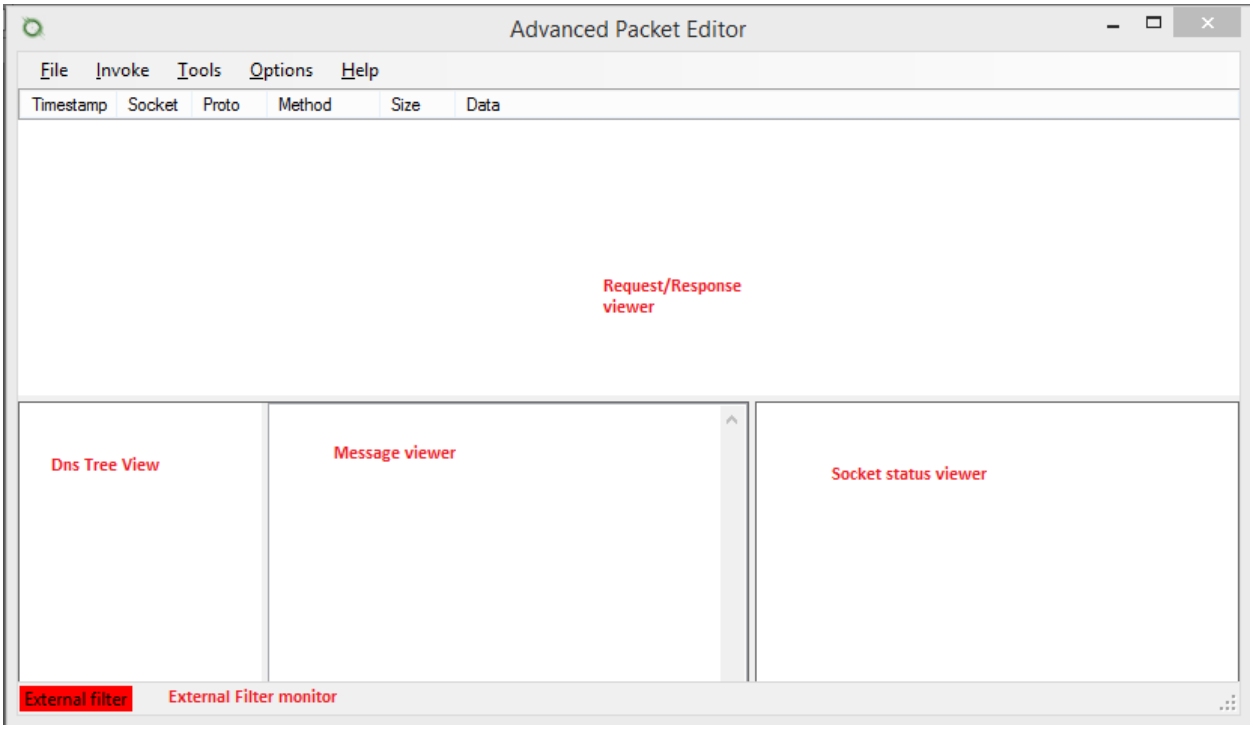
Advanced Packet Editor (APE) is an open source packet editor for viewing and editing TCP and HTTP communication.

APE can attach itself to a process to get intercept communication between a server and a client

APE uses filters, both external (using python scripts) and internal for automatically substituting text on the fly or simply injecting/editing request and responses.

APE is easy to use and configure!

Main Screen



Request / Response viewer

Timestamp	Socket	Proto	Method	Size	Data
15:57:47	01EC	IP	send()	199	
15:57:47	01EC	IP	WSARecv()	7	
15:57:47	020C	IP	send()	8	
15:57:47	020C	IP	WSARecv()	10	
15:57:48	020C	IP	send()	179	
15:57:48	020C	IP	WSARecv()	30	

TimeStamp – The timestamp of the message.

Socket – The socket address.

Proto – Protocol used for the Message.

Method – The method/function that was used (e.g. send(), WSASend(), recv(), etc).

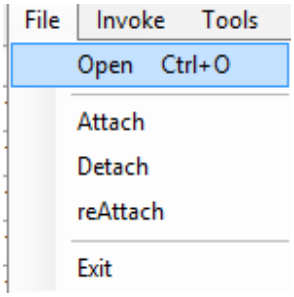
Size – Size of the message

Data - The data that was sent/received.

File Menu

Executing an application and attaching to its process

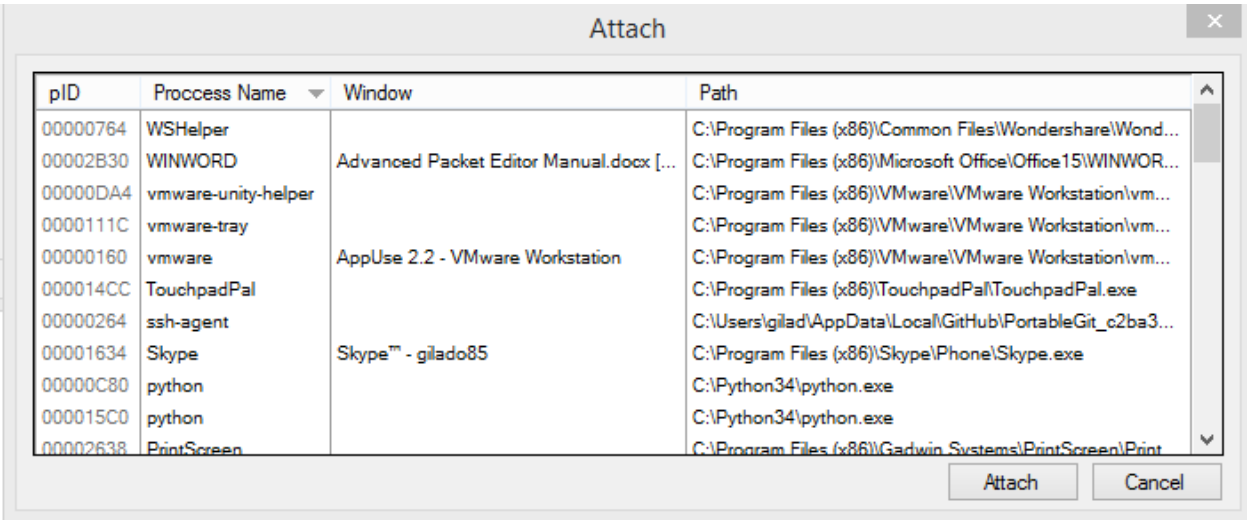
1. Click on File -> Open



2. From the popup file selection, select the file which you wish to execute
3. Advanced packet editor will automatically attach the new process

Attaching a process

1. Click on File -> Attach



2. Select the process that you wish to attach Advanced packet editor to and click 'Attach'

Detaching a process

When Advanced packet editor is attached to a process Click on File -> Detach to detach APE from the current process.

Reattaching a previous process

If your process gets detached (if it closed or was manually detached) you can click on File -> reattach to reattach the APE to the previous process or a new process with the same name.

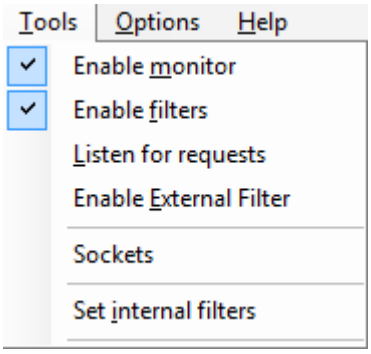
Invoke menu

Freeze

Click on Invoke -> Freeze to freeze the current attached process.

Clicking on Invoke -> Freeze will unfreeze the process.

Tool menu



Enable/Disable monitor

Enable or disable logging of incoming and outgoing packets.

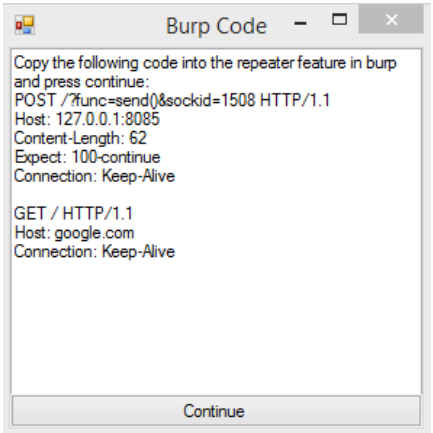
Enable/Disable filters

Enable or disable filtering of packets. If filtering was disabled, results will show "as they are".

Listen to request

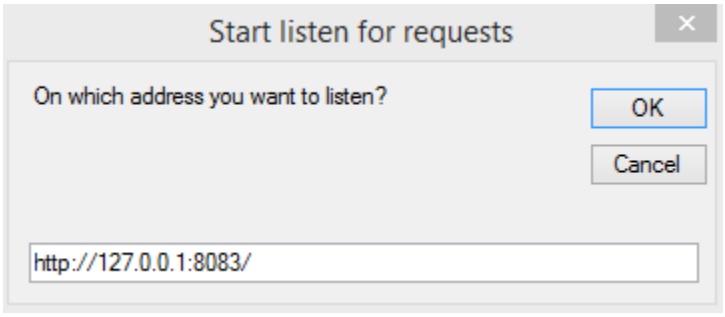
Enable or disable integration with external tool (such as Burp Suite proxy), the goal is to help you send your own data to the server on the exist sockets.

1. A popup will appear with request format that needs to be added to Burp Suit for proxy/repeater integration:



Click continue when the request was copied to the repeater. Set the func and sockid in the URL to tell the APE how and where to inject your data. Currently func support only "send" option.
The second http request is actually the data to send the server. Of course, it may be any binary data, it is just a sample.

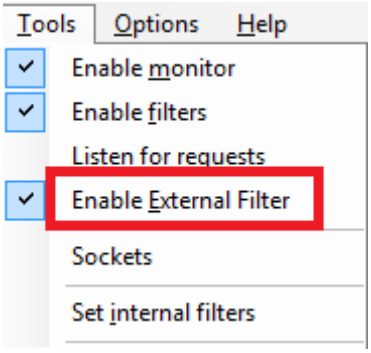
2. A prompt will popup asking for the address and port to listen for. This address you'll need to configure in Burp Suite when you send the data to the APE.



Enabling/Disabling External Filters

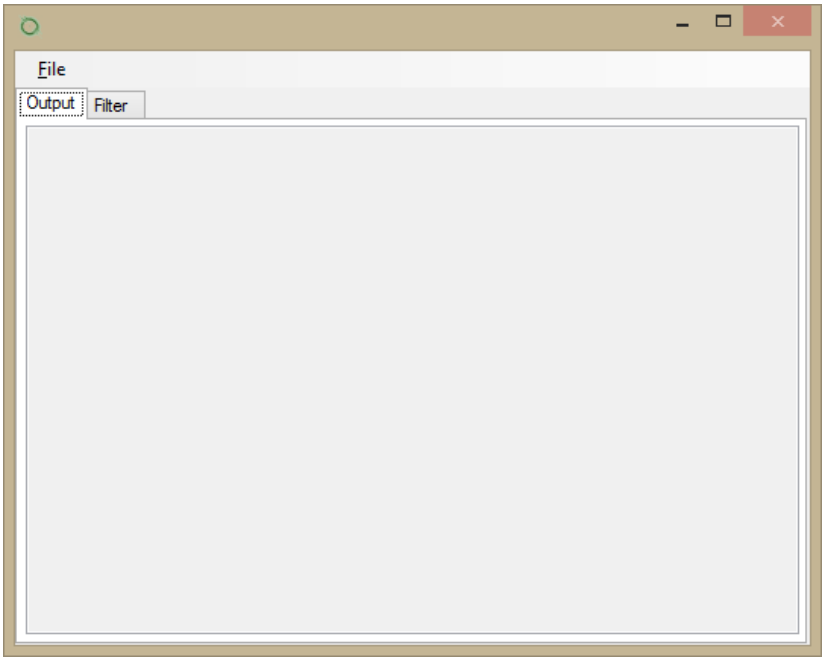
Advanced packet editor allows customization of filters from an external source written in Python.

1. Click on Enable filter

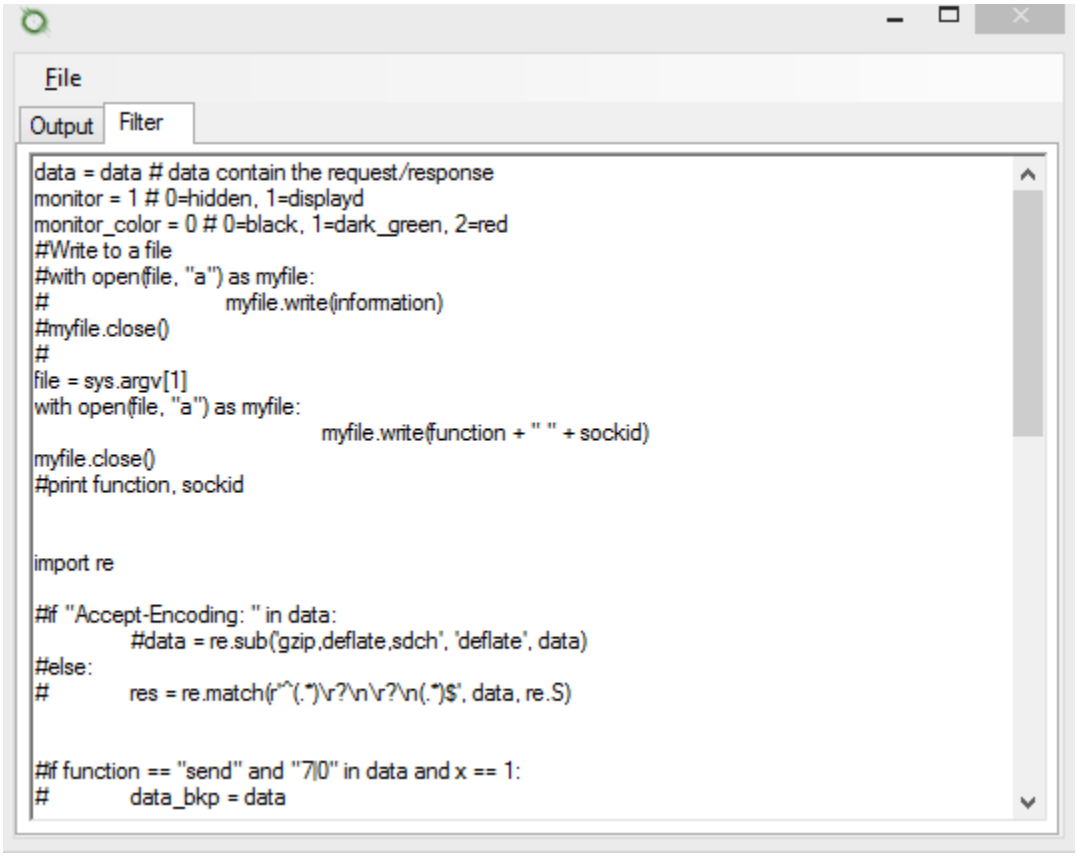


2. Choose a port in the popup prompt (it is recommended to not change it)
3. The External filter label will change to green

4. A external python filter window viewer will open



5. The 'Output' tab will display the log from the external python filter
6. The 'Filter' tab lets you choose the filter options

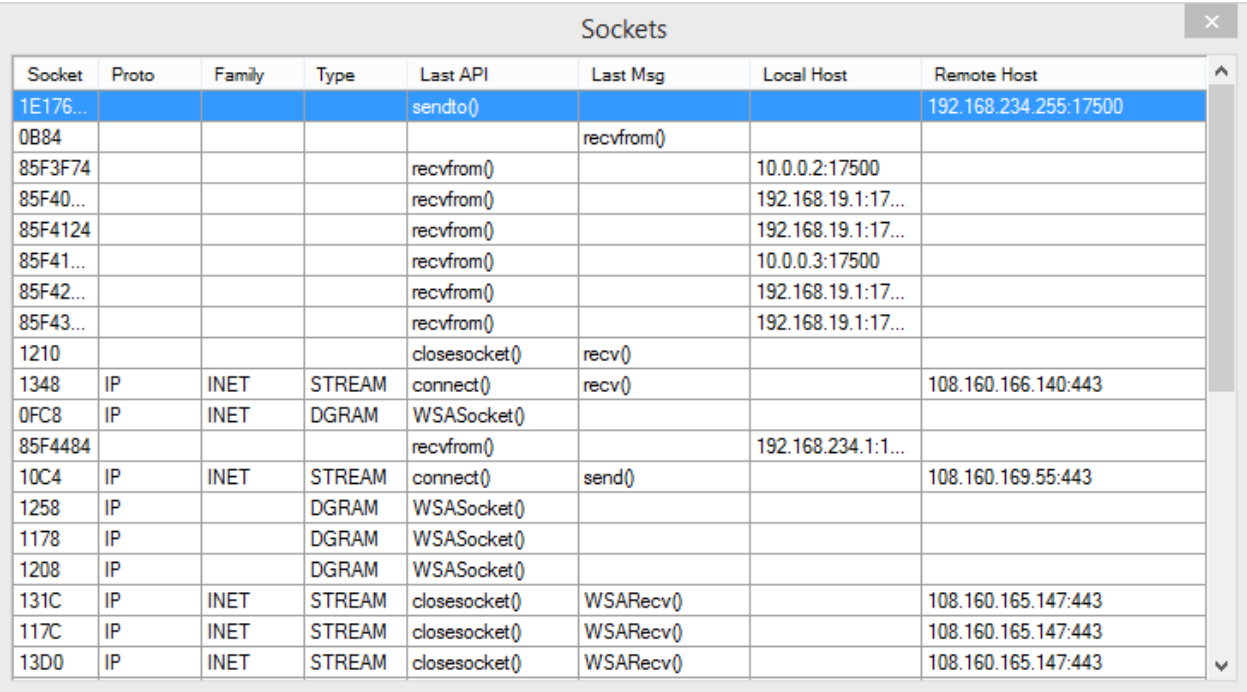


By changing the variable data you can manipulate the data. By changing the variables monitor and monitor_color you can manage the appearance in the filter-log.
It is recommended to leave the first three lines.

Remember to save the changes using the File menu. Check in the Output tab if it works OK and no syntax error.

Sockets

Clicking on Sockets will show a more detailed view of the currently opened sockets:

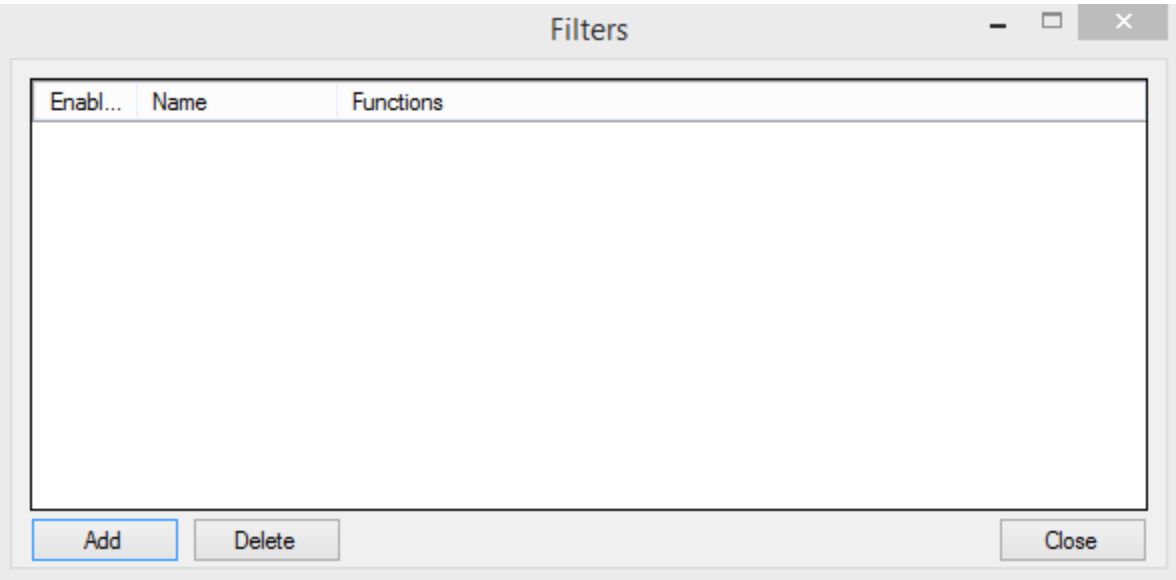


Socket	Proto	Family	Type	Last API	Last Msg	Local Host	Remote Host
1E176...				sendto()			192.168.234.255:17500
0B84					recvfrom()		
85F3F74				recvfrom()		10.0.0.2:17500	
85F40...				recvfrom()		192.168.19.1:17...	
85F4124				recvfrom()		192.168.19.1:17...	
85F41...				recvfrom()		10.0.0.3:17500	
85F42...				recvfrom()		192.168.19.1:17...	
85F43...				recvfrom()		192.168.19.1:17...	
1210				closesocket()	recv()		
1348	IP	INET	STREAM	connect()	recv()		108.160.166.140:443
0FC8	IP	INET	DGRAM	WSASocket()			
85F4484				recvfrom()		192.168.234.1:1...	
10C4	IP	INET	STREAM	connect()	send()		108.160.169.55:443
1258	IP		DGRAM	WSASocket()			
1178	IP		DGRAM	WSASocket()			
1208	IP		DGRAM	WSASocket()			
131C	IP	INET	STREAM	closesocket()	WSARecv()		108.160.165.147:443
117C	IP	INET	STREAM	closesocket()	WSARecv()		108.160.165.147:443
13D0	IP	INET	STREAM	closesocket()	WSARecv()		108.160.165.147:443

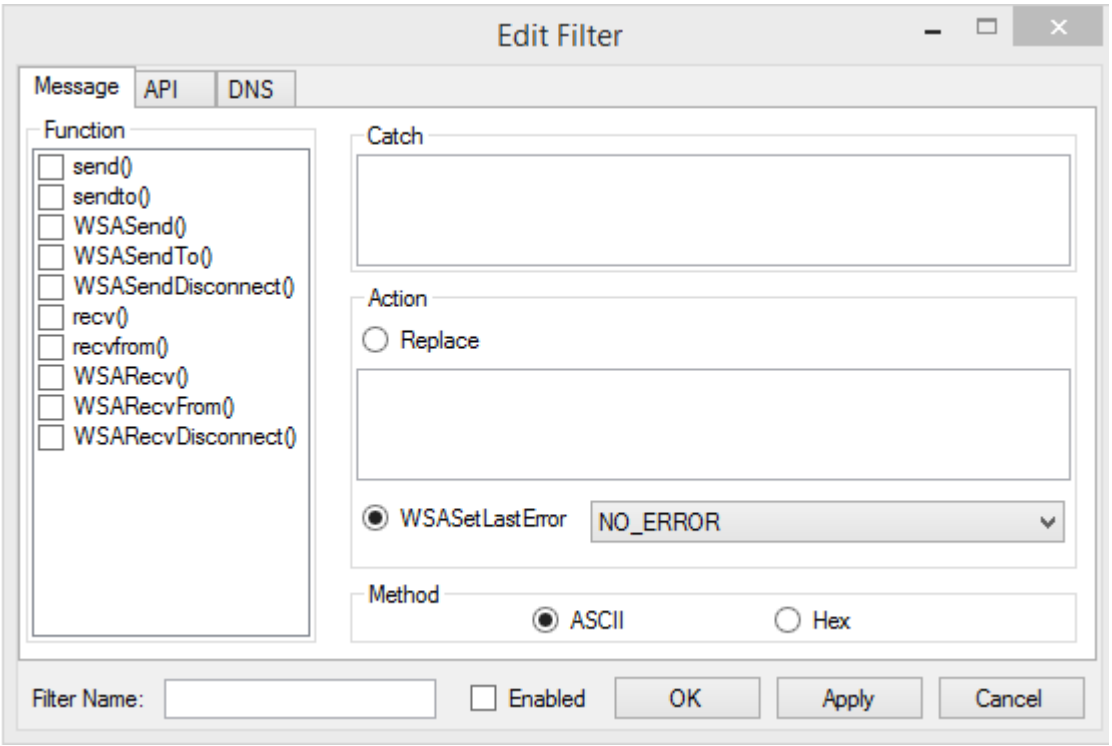
Setting internal filters

This window is used for simple manipulation and/or manipulations that does not supported in the Python filter module.

Internal filter rule menu

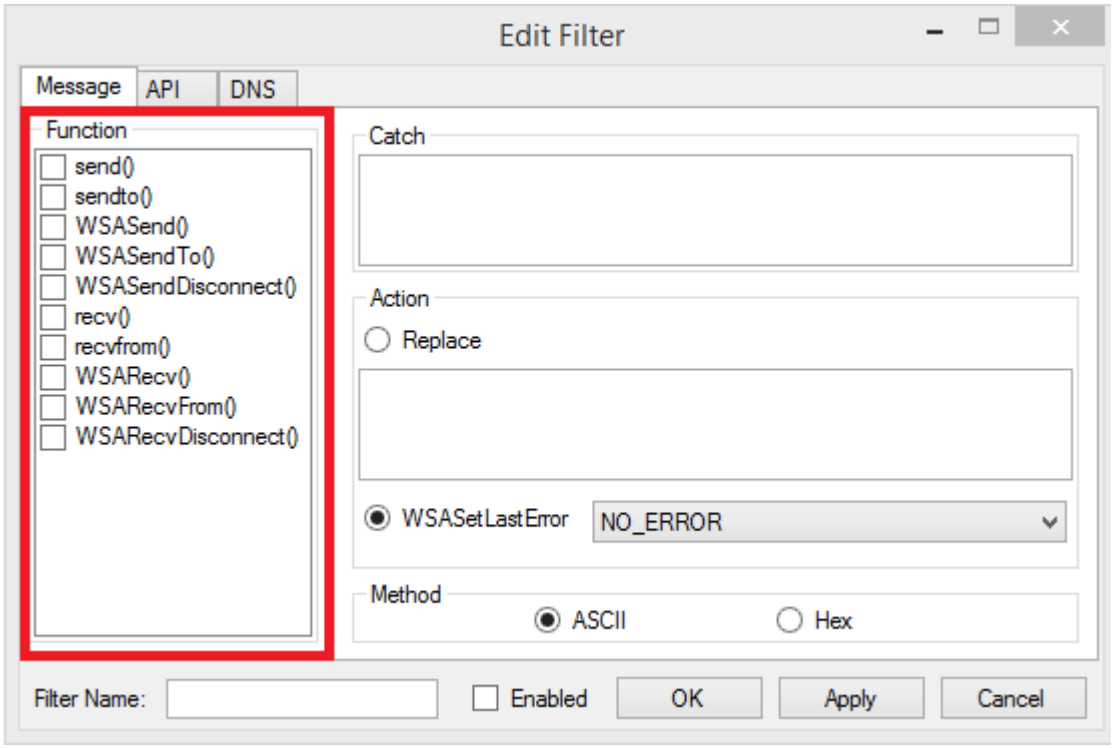


Clicking 'Add' will open a rule creating dialog



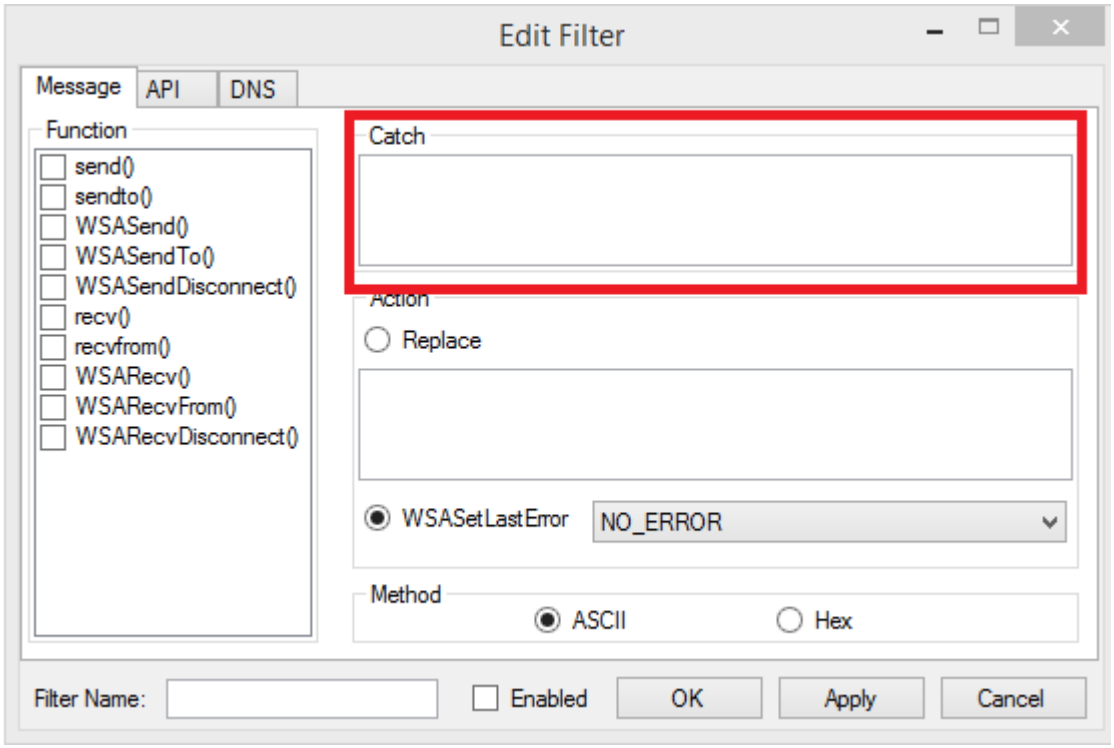
Message/API/DNS tab

Function



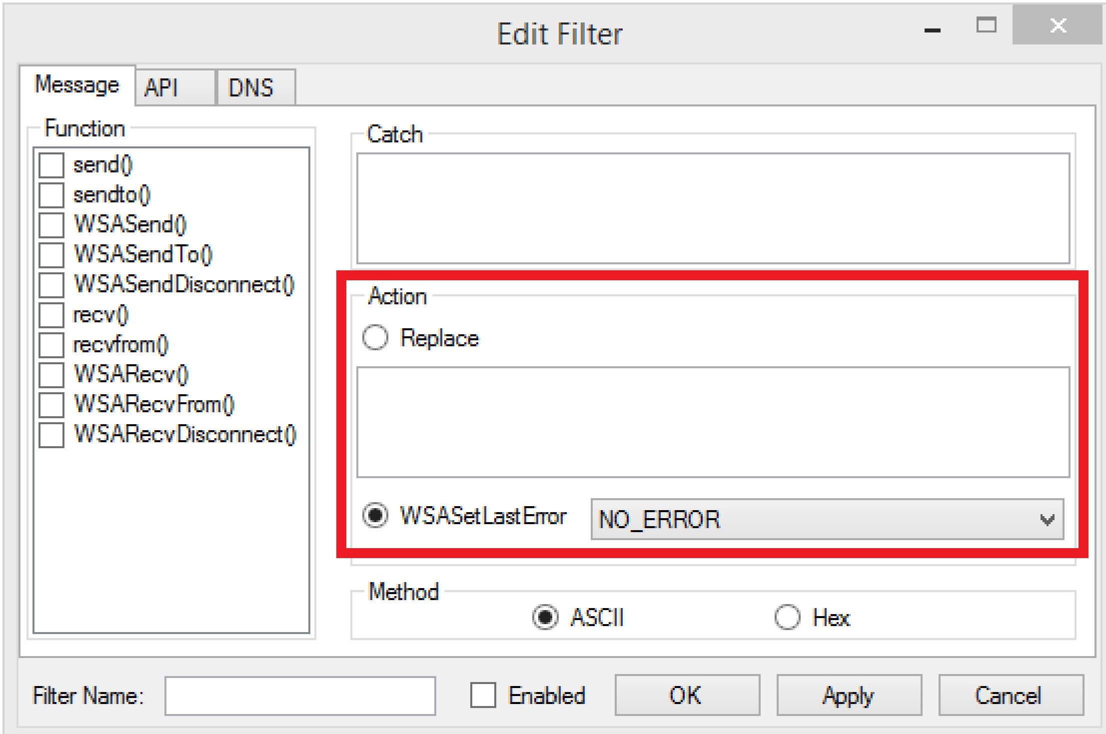
Selecting the function for which the rule should apply

Catch



Enter a string to look for the specified function.

Action



Choose whether to replace the caught string or to set the WSASetLast Error

Set the filter name, and tag enable to enable the rule

Captured packets

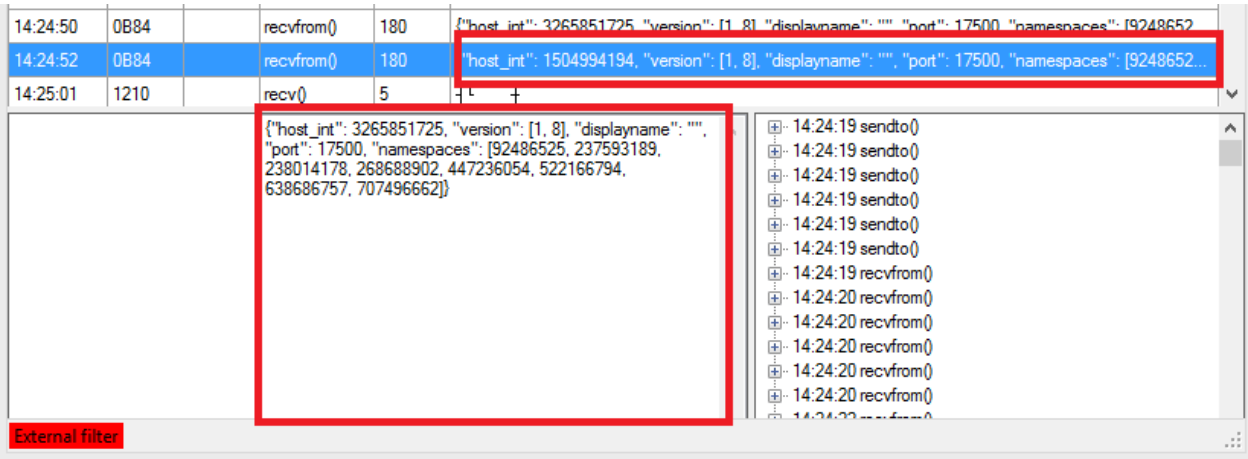
Once attached to a process, Advanced packet editor will start listening to incoming and outgoing packets.

The Request/Response Viewer will present the information in a grid.

For example:

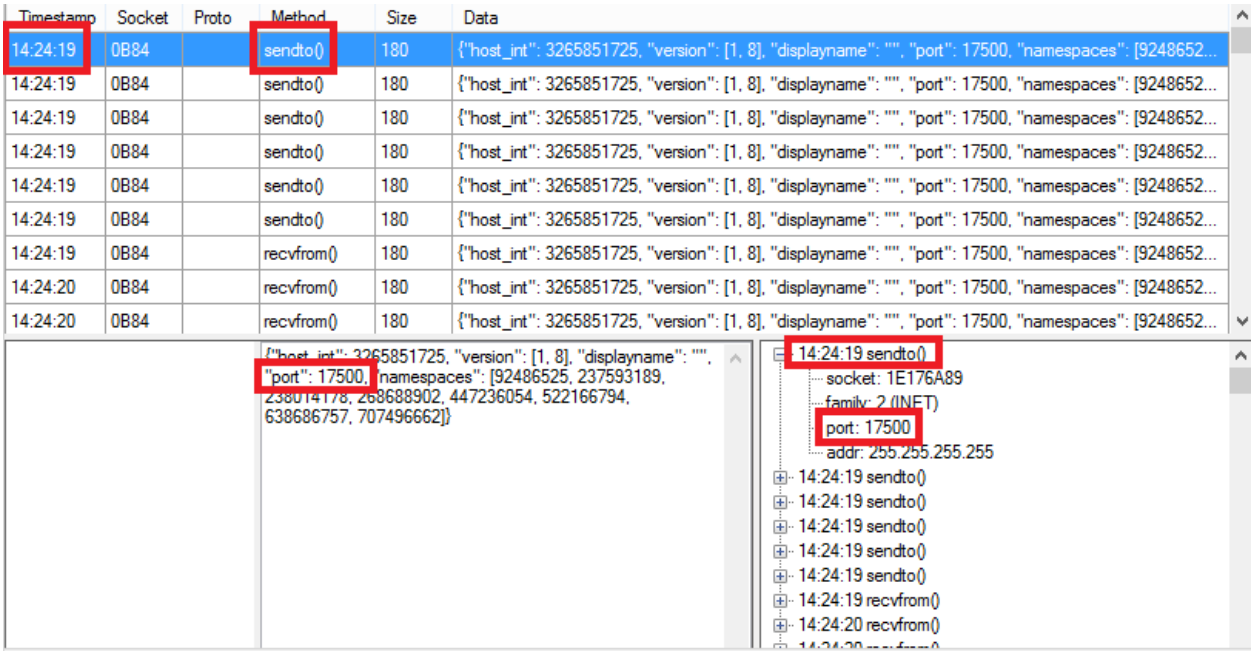
Timestamp	Socket	Proto	Method	Size	Data
14:24:19	0B84		sendto()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...
14:24:19	0B84		sendto()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...
14:24:19	0B84		sendto()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...
14:24:19	0B84		sendto()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...
14:24:19	0B84		sendto()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...
14:24:19	0B84		sendto()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...
14:24:19	0B84		recvfrom()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...
14:24:20	0B84		recvfrom()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...
14:24:20	0B84		recvfrom()	180	{"host_int": 3265851725, "version": [1, 8], "displayname": "", "port": 17500, "namespaces": [9248652...

Selecting a record will fill the Message viewer with the data:

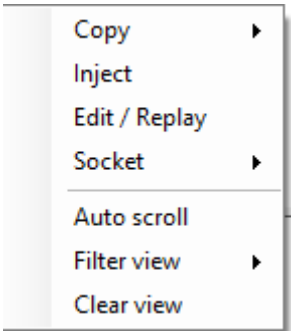


Socket status

While information passes between a client and a server, you can view the status of the socket of each request and response in the Socket status viewer:



Message editing



Copy

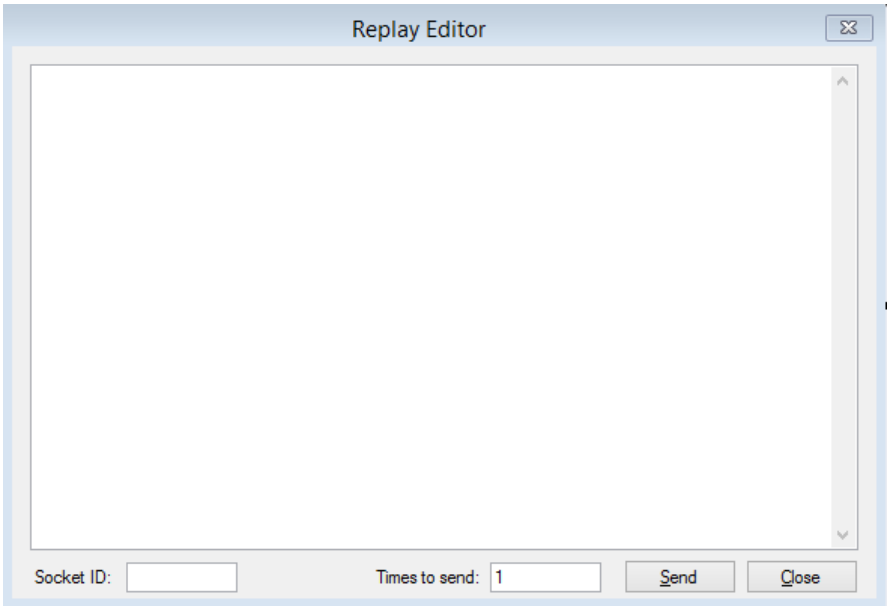
Click copy to copy the selected data either in Hex or in ASCII format.

Inject

Inject data into the selected socket

Edit / Replay

Right-clicking on a row in the Message viewer and choosing Edit / Replay will open a replay dialog:



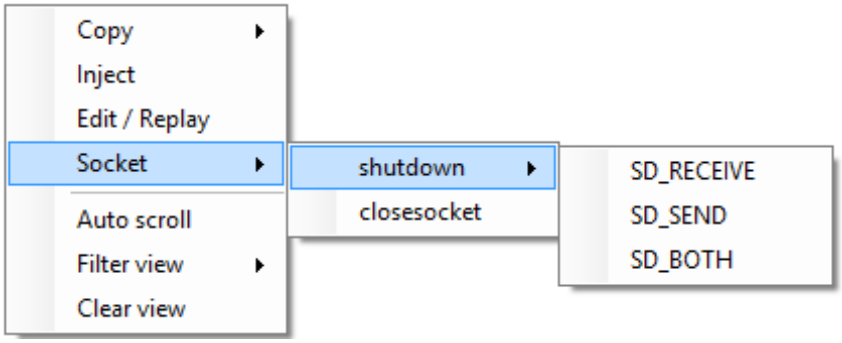
The socket ID should be filled and can be changed.

Change the content either in HEX or in ASCII.

Select the number of times to replay the message. It is very useful for race condition attacks.

Click Send to start sending the message or close to close out.

Socket



Shutdown or close the selected socket.

Auto Scroll

Enable / Disable auto scrolling.

Filter View

Choose which messages to display, Request and/or responses.

Clear View

Clear the message viewer.

Enjoy!