PROXY

**Overview**

The Proxy scenario is divided into two types:

•         Proxy the socket which is achieved in Win7+ by use of FWPM\_LAYER\_ALE\_BIND\_REDIRECT.

•         Proxy the connection which is further divided into two methods:

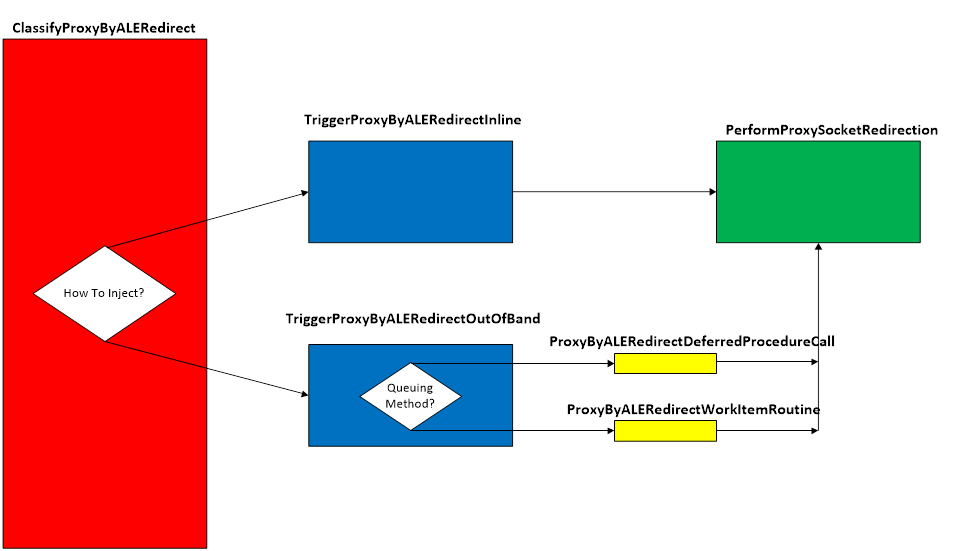
o   Proxy the connection using injection which is supported on Vista+.  It should be noted though that for Win7+ the redirect method should be used.

o   Proxy the connection using FWPM\_LAYER\_ALE\_CONNECT\_REDIRECT.  This method is recommended for all proxying of connections post Vista.

All filters added sit in WFPSampler's sublayer (which is weighted just below IPsec's sublayer), unless otherwise specified using the -sl <SUBLAYER> command line option.  All filters are associated with WFPSampler's provider.

**Proxy the Socket**

The following diagram shows how the code flows for this callout:

  
**Figure A. Code flow for Proxy (socket) by ALE Redirect Scenario**

When traffic matches a filter at the specified layer, **ClassifyProxyByALERedirect()** is invoked by the Filtering Engine.  This function will create the REDIRECT\_DATA which consists of the classifyHandle, the redirectHandle, and the writableLayerData.  The appropriate triggerFn is called.

If the operation method is synchronous (inline), **TriggerProxyByALERedirectInline()** is invoked.  This function creates the CLASSIFY\_DATA, which consists of the data that was passed into the classifyFn.  **PerformProxySocketRedirection**() is then invoked.

If the injection method is asynchronous (out of band), **TriggerProxyByALERedirectOutOfBand()** is invoked.  This function creates the CLASSIFY\_DATA which consists of copies and references of the data that was passed into the classifyFn.  Based on the queuing method, the appropriate queueFn is invoked.

Regardless of which queueFn is used, each will call **PerformProxySocketRedirection()**.

**PerformProxySocketRedirection** () will modify the writable layer data with the values passed to the command line.  The action is set to FWP\_ACTION\_PERMIT and the layer data is applied.  From this point on, the socket will be bound to the new local address and /or new local port that was specified on the command line.

**Applicable Layers**

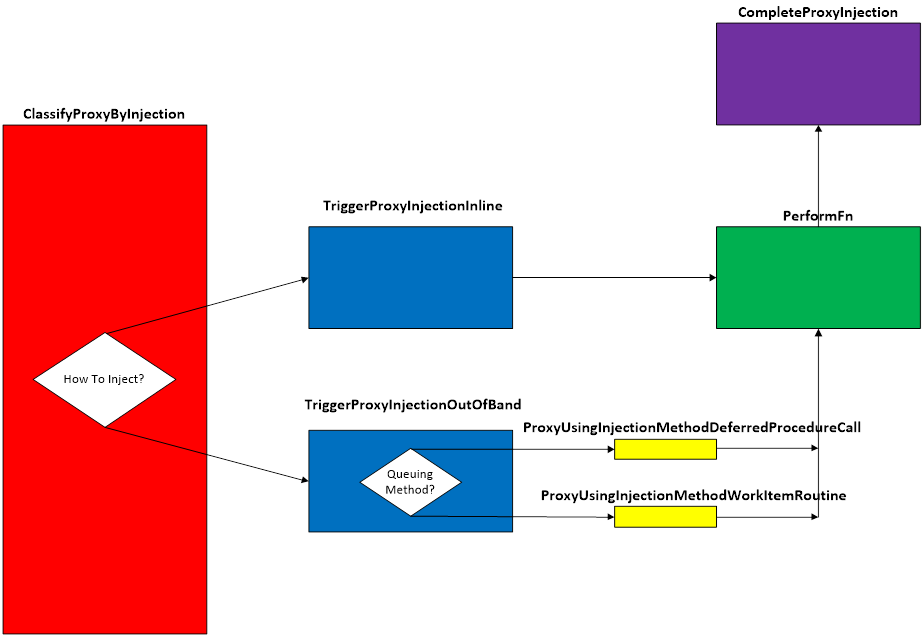
❖  FWPM\_LAYER\_ALE\_BIND\_REDIRECT\_V4              (Win7+)

❖  FWPM\_LAYER\_ALE\_BIND\_REDIRECT\_V6              (Win7+)

**Proxy the Connection**

**By Injection**

The following diagram shows how the code flows for this callout:

  
**Figure B. Code flow for Proxy By Injection Scenario**

When traffic matches a filter at the specified layer, **ClassifyProxyByInjection()** is invoked by the Filtering Engine.  This function validates that we can perform the injection by looking at the pClassifyOut rights.  It will then create the INJECTION\_DATA which consists of the injectionHandle and the injectionState.  If the injectionState indicates that we haven’t injected this packet before, then the injection method is determined (default is asynchronous), and the appropriate triggerFn is called.   At this point, the original packet will be blocked.

If the injection method is synchronous (inline), **TriggerProxyInjectionInline()** is invoked.  This function creates the CLASSIFY\_DATA, which consists of the data that was passed into the classifyFn.  Depending on which layer the injection is happening, the appropriate performFn is called.

If the injection method is asynchronous (out of band), **TriggerProxyInjectionOutOfBand()** is invoked.  This function creates the CLASSIFY\_DATA which consists of copies and references of the data that was passed into the classifyFn.  Based on the queuing method, the appropriate queueFn is invoked.

Regardless of which queueFn is used, each will call the appropriate performFn based on the layer the injection is happening.

Each of the performFns are tailored to inject for their respective layers.  Each will get the required data for its specific injectionFn.  Depending on the layer, the offsets are adjusted on the original so the whole packet is available.  Once the offsets are adjusted, the original is cloned, and the offsets of the original are returned to the original place.  The clone is modified with the new address and ports that are specified, and the injection function is called.

Upon successful injection, **CompleteProxyInjection()** will be called by the TCP/IP stack.  This function will show the status of the injected packet.  Additionally, any memory that was allocated from the functions above, will be freed and any references released.

**Applicable Layers**

❖  FWPM\_LAYER\_INBOUND\_TRANSPORT\_V4         (Vista+)

❖  FWPM\_LAYER\_INBOUND\_TRANSPORT \_V6        (Vista+)

❖  FWPM\_LAYER\_OUTBOUND\_TRANSPORT\_V4     (Vista+)

❖  FWPM\_LAYER\_OUTBOUND\_TRANSPORT \_V6    (Vista+)

❖  FWPM\_LAYER\_DATAGRAM\_DATA\_V4                  (Vista+)

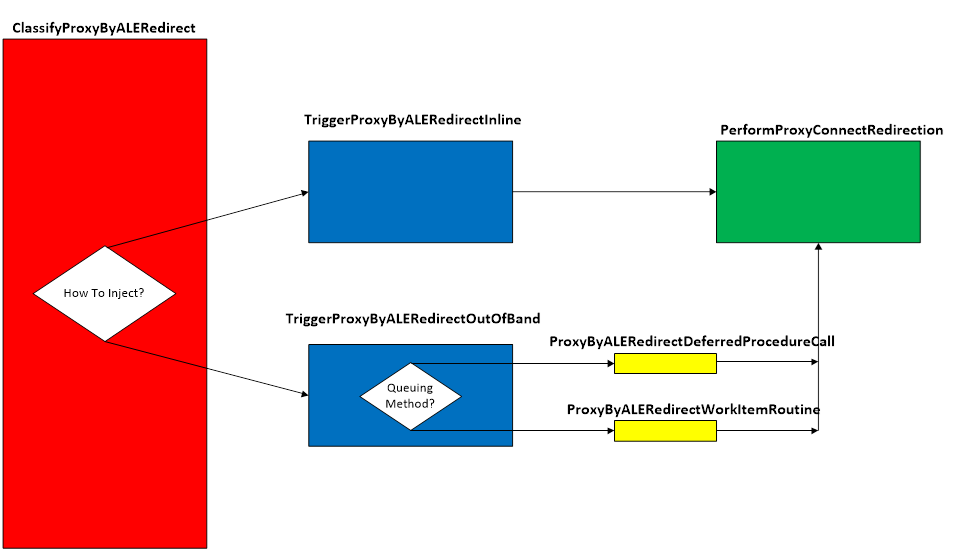
❖  FWPM\_LAYER\_DATAGRAM\_DATA\_V6                  (Vista+)

❖  FWPM\_LAYER\_STREAM\_PACKET\_V4                     (Win7+)

❖  FWPM\_LAYER\_STREAM\_PACKET\_V6                     (Win7+)

**By ALE Redirect**

The following diagram shows how the code flows for this callout:

  
**Figure C. Code flow for Proxy (connection) by ALE Redirect Scenario**

When traffic matches a filter at the specified layer, **ClassifyProxyByALERedirect()** is invoked by the Filtering Engine.  This function will create the REDIRECT\_DATA which consists of the classifyHandle, the redirectHandle, and the writableLayerData.  The appropriate triggerFn is called.

If the operation method is synchronous (inline), **TriggerProxyByALERedirectInline()** is invoked.  This function creates the CLASSIFY\_DATA, which consists of the data that was passed into the classifyFn.  **PerformProxyConnectRedirection**() is then invoked.

If the injection method is asynchronous (out of band), **TriggerProxyByALERedirectOutOfBand()** is invoked.  This function creates the CLASSIFY\_DATA which consists of copies and references of the data that was passed into the classifyFn.  Based on the queuing method, the appropriate queueFn is invoked.

Regardless of which queueFn is used, each will call **PerformProxyConnectRedirection()**.

**PerformProxyConnectRedirection** () will modify the writable layer data with the values passed to the command line.  The action is set to FWP\_ACTION\_PERMIT and the layer data is applied.  From this point on, the connection will be adjusted to use the new remote address and /or new remote port that was specified on the command line.

**Applicable Layers**

❖  FWPM\_LAYER\_ALE\_CONNECT\_REDIRECT\_V4     (Win7+)

❖  FWPM\_LAYER\_ALE\_CONNECT\_REDIRECT\_V6     (Win7+)

**Command Line Usage**

|  |  |  |
| --- | --- | --- |
| **Option** | **Argument** | **Meaning** |
| -s | PROXY | Implement the PROXY scenario |
| -l | Applicable Layer | Layer at which this filter will apply |
| -pla | IP Address | New local IP address for the socket (or connection if proxying the connection by injection). |
| -plp | Port | New local port for the socket (or connection if proxying the connection by injection). |
| -pra | IP Address | New destination IP address for the connection. |
| -prp | Port | New destination port for the connection. |
| -plspid | Process ID of local Proxy Service | Proxy to the specified local proxy service.  This is only valid for proxying the connection by ALE redirection. |
| -prs |  | Proxy to a remote proxy service.  This is valid only for proxying the connection by ALE redirection. |
| -sl | Applicable subLayer | SubLayer to associate with the filter.  [default is WFPSAMPLER\_SUBLAYER]. |
| -v |  | Make the objects associated with this scenario’s instance dynamic |
| -b |  | Make the objects associated with this scenario’s instance available during boot-time |
| -in |  | Perform the injection synchronously (inline) |
| -tdpc |  | Use threaded DPCs for asynchronous (out of band) queuing method |
| -wi |  | Use work items for asynchronous (out of band) queuing method |
| -r |  | Remove objects associated with this scenario instance |
| -? |  | Display help |

“**WFPSampler.Exe -s PROXY -?**“ provides help output

“**WFPSampler.Exe -s PROXY -l FWPM\_LAYER\_ALE\_BIND\_REDIRECT\_V4 -aaid C:\Traffic.exe -ipla 1.0.0.1 -pla 1.0.0.2 -plp 0x4444 -v”** adds a dynamic filter (**-v**) at FWPM\_LAYER\_ALE\_BIND\_REDIRECT\_V4 (**-l**) which references the appropriate callout.  This filter will have 2 conditions; FWPM\_CONDITION\_ALE\_APP\_ID (**-aaid**) equals C:\Traffic, and FWPM\_CONDITION\_IP\_LOCAL\_ADDRESS (**-ipla**) equals 1.0.0.1.  When classified, it will modify the socket to be bound to local address (**-pla**) 1.0.0.2 and local port (**-plp**) 0x4444.  This change endures for the lifetime of the socket.

**WFPSampler.Exe -s PROXY -l FWPM\_LAYER\_ALE\_BIND\_REDIRECT\_V4 -aaid C:\Traffic.exe -ipla 1.0.0.1 -pla 1.0.0.2 -plp 0x4444 -v -r**“  removes (**-r**) the dynamic filter (**-v**) at FWPM\_LAYER\_ALE\_BIND\_REDIRECT\_V4 (**-l**) which references the appropriate callout.

“**WFPSampler.Exe -s PROXY -l FWPM\_LAYER\_OUTBOUND\_TRANSPORT\_V4 -pra 127.0.0.1 -prp 0x4444**“ adds a persistent filter at FWPM\_LAYER\_OUTBOUND\_TRANSPORT\_V4 (**-l**) which references the appropriate callout.  This filter will have no conditions, meaning it will act on all traffic seen at this layer.  It will modify the destination address (**-pra**)to the software loopback (**127.0.0.1**) and the destination port (**-prp**) to 0x4444.  For full end-to-end proxying to occur, a second filter will need to be added in the inverse direction to proxy back to the original endpoint.

“**WFPSampler.Exe -s PROXY -l FWPM\_LAYER\_OUTBOUND\_TRANSPORT\_V4 -pra 127.0.0.1 -prp 0x4444 -r**“ removes (**-r**) the persistent filter at FWPM\_LAYER\_OUTBOUND\_TRANSPORT\_V4 (**-l**) which references the appropriate callout.

“**WFPSampler.Exe -s PROXY -l FWPM\_LAYER\_ALE\_CONNECT\_REDIRECT\_V4 -aaid C:\Traffic.exe -ipla 1.0.0.1 -pra 127.0.0.1 -prp 0x4444 –plspid 501 -v”** adds a dynamic filter (**-v**) at FWPM\_LAYER\_ALE\_BIND\_REDIRECT\_V4 (**-l**) which references the appropriate callout.  This filter will have 2 conditions; FWPM\_CONDITION\_ALE\_APP\_ID (**-aaid**) equals C:\Traffic, and FWPM\_CONDITION\_IP\_LOCAL\_ADDRESS (**-ipla**) equals 1.0.0.1.  When classified, it will modify the connection’s TCB to use remote address (**-pra**) 127.0.0.1 and remote port (**-prp**) 0x4444.  The process that is listening for this modified traffic is identified by processe ID (**-plspid**) 501. This change endures for the lifetime of the connection.

“**WFPSampler.Exe -s PROXY -l FWPM\_LAYER\_ALE\_CONNECT\_REDIRECT\_V4 -aaid C:\Traffic.exe -ipla 1.0.0.1 -pra 127.0.0.1 -prp 0x4444 –plspid 501 -v -r**“  removes (**-r**) the dynamic filter (**-v**) at FWPM\_LAYER\_ALE\_CONNECT\_REDIRECT\_V4 (**-l**) which references the appropriate callout.

“**WFPSampler.Exe -s PROXY -l FWPM\_LAYER\_ALE\_CONNECT\_REDIRECT\_V4 -aaid C:\Traffic.exe -ipla 1.0.0.1 -pra 1.0.0.254 -prp 0x4444 –prs-v”** adds a dynamic filter (**-v**) at FWPM\_LAYER\_ALE\_BIND\_REDIRECT\_V4 (**-l**) which references the appropriate callout.  This filter will have 2 conditions; FWPM\_CONDITION\_ALE\_APP\_ID (**-aaid**) equals C:\Traffic, and FWPM\_CONDITION\_IP\_LOCAL\_ADDRESS (**-ipla**) equals 1.0.0.1.  When classified, it will modify the connection’s TCB to use remote address (**-pra**) 1.0.0.254 and remote port (**-prp**) 0x4444.  The process that is listening for this modified traffic is on a remote server (**-prs**). This change endures for the lifetime of the connection.

For a list of conditions applicable to each layer, refer to Filtering Conditions Available at Each Filtering Layer.

For a list of command line parameters for configuring each condition, refer to Conditions for Command Line.