Simple Sideband User Instructions

Peter White Version 1.2 17th July 2023

Introduction

Simple Sideband is an iRule library which helps the user to perform sideband operations in a simple way. A sideband operation is where an iRule creates a network connection to an external service such as a web server.

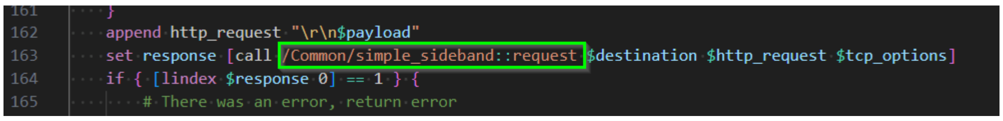
Simple sideband is installed as an iRule library and called from a separate iRule which is installed on the virtual server. There are six functions – *http_req*, *tcp_req*, *udp_req*, *dns_query*, *dns_response* and *request*. The user can control options such as HTTP method, timeouts, returned data etc.

The library also contains a helper iRule which runs on the helper virtual server and sends traffic to the destination specified in a HTTP header.

Installation

Rule naming

The iRule should be installed as /Common/simple_sideband. Note that it *can* be installed as a different name, but this name is used within the iRule so that should be updated. For example line 163 uses the /Common/simple_sideband::request so this would need to be updated:



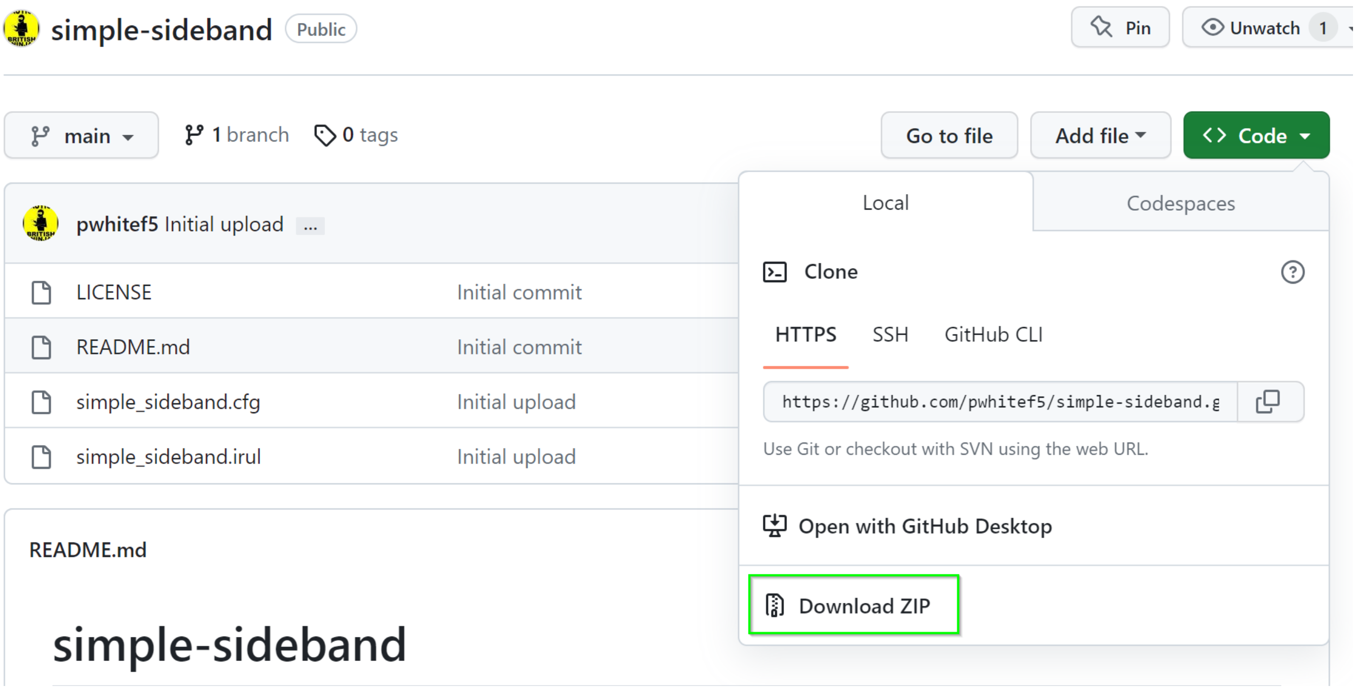
```
161     }
162     append http_request "\r\n$payload"
163     set response [call /Common/simple_sideband::request $destination $http_request $tcp_options]
164     if { [lindex $response 0] == 1 } {
165         # There was an error, return error
```

Download the latest version

- Navigate to my Github page at https://github.com/pwhitef5/simple-sideband and download the repository as a ZIP file:

Simple Sideband User Instructions

Peter White   Version 1.2   17th July 2023



pwhitef5 Initial upload	
LICENSE	Initial commit
README.md	Initial commit
simple_sideband.cfg	Initial upload
simple_sideband.irul	Initial upload

https://github.com/pwhitef5/simple-sideband.g

Use Git or checkout with SVN using the web URL.

README.md

simple-sideband

- Extract the zip archive

Installation on platform

There are two ways to install the iRule – via the GUI or via tmsh

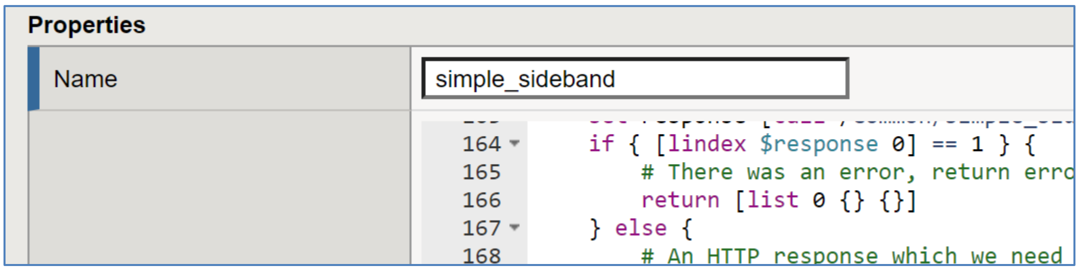
Loading iRule via GUI

This is very simple. First, login to the GUI as an appropriate user eg admin and ensure you are in the Common partition

- Navigate to Local Traffic > iRules and hit the + button (

), or the Create on the right hand side (

)
- Set Name to be *simple_sideband* and paste in the contents of simple_sideband.irul



```
    if { [lindex $response 0] == 1 } {
        # There was an error, return erro
        return [list 0 {} {}]
    } else {
        # An HTTP response which we need
```

- Hit Finished

Loading iRule via tmsh

- Copy the simple_sideband.cfg file to the /var/tmp directory of the target device using an appropriate method eg SCP

Simple Sideband User Instructions

Peter White   Version 1.2   17th July 2023

- Run the command *tmsh load sys config merge file /var/tmp/simple_sideband.cfg*

```
tmsh load sys config merge file /var/tmp/simple_sideband.cfg
Loading configuration...
  /var/tmp/simple_sideband.cfg
```

- Save the config with the command *tmsh save sys config*

```
tmsh save sys config
Saving running configuration...
  /config/bigip.conf
  /config/bigip_base.conf
  /config/bigip_user.conf
  /config/partitions/dmzinternet/bigip.conf
Saving Ethernet map ...done
Saving PCI map ...
 - verifying checksum .../var/run/f5pcimap: OK
done
 - saving ...done
```

Simple Sideband User Instructions

Peter White   Version 1.2   17th July 2023

Usage

Simple sideband is a library which provides functionality to other iRules. There are a number of functions – the basic of which is *request*.

Request

This takes three variables: destination, payload and options

- Destination is either an IP address and port eg “1.2.3.4:80” or the full name of a helper virtual server eg “/Common/https_helper”
- Payload is the payload to be sent – either ASCII data such as HTTP or binary data encoded with binary format
- Options allows you to set certain options as a TCL list. When you set this in dependent functions such as http_req, this will flow down to request.
    - protocol – String. Set the transport protocol ie TCP or UDP. Default is TCP
    - debug – Boolean. Create debugging logs. Defaults to 0
    - connect_timeout. Float. Set connect timeout in secs. Default 1 sec
    - timeout – Float. Set total timeout in seconds. Default 5 secs
    - idle – Float. Set the idle timeout. Default 3 secs
    - recv_bytes – Integer. Set amount of data to receive. Default 0
    - retries – Integer. Set the number of retries. Default 1

Example:

```
set response [call /Common/simple_sideband::request 10.67.182.10:80 "GET
/\r\n" {connect_timeout 3 recv_bytes 3} ]
```

The response from this function is a TCL list such as { <error> <data> }

Error is a Boolean to indicate whether the procedure hit an error ie if it is 0, there was no error. In the case that there is no error, data will be either the returned data or, if no data is returned, the number of bytes sent. If there is an error, data will show the error message. Therefore you should check the return from the procedure before using data

Example:

```
set response [call /Common/simple_sideband::request 10.67.182.10:80 "GET
/\r\n" {connect_timeout 3 recv_bytes 3} ]
if { [lindex $response 0] == 1 } {
      log local0.err "Error [lindex $response 1] "
      return
} else {
      Set data [lindex $response 1]
}
```

Simple Sideband User Instructions

Peter White Version 1.2 17th July 2023

Note that if you want to receive the response, you have to set the *recv_bytes* option to a positive number, otherwise it will just send the request and return the number of octets sent.

tcp_req and udp_req

These are basically a pass-through to the request function. Set destination, payload and options.

http_req

This is used for sending and receiving HTTP-based data. For HTTPS, use a helper virtual server as shown in examples.

Variables are destination, url and options. As with request, destination is an ip address:port or a helper virtual server name. url defines the URL to be requested eg /index.html. Options includes the options from request but also has:

- method – the HTTP method to be used eg POST. Default is GET
- version – the HTTP version. Default is v1.1
- payload – in the case of a POST this is the payload to be sent
- headers – a TCL list of extra headers to be sent eg { Content-Type application/json }

The return is an array such as <status code> <headers> <body>.

The HTTP status code (or 0 for timeout) eg 200. Headers is a TCL list of response headers eg { Content-Length 123}. Body is the response body. This will often be ASCII text such as HTML, but could be binary data such as a JPEG.

dns_query

This is a procedure to demonstrate creating a binary packet ie a DNS query packet, to be sent to a DNS resolving name server. Variable is an FQDN such as www.example.com, return is a valid DNS query packet ready to be sent by udp_req.

dns_response

The partner to dns_query, this is used to decode a DNS response packet. Variable is a response packet such as from udp_req, return is a TCL list such as: <header> <query> <answers> <auth> <additional>

Simple Sideband User Instructions

Peter White   Version 1.2   17th July 2023

Helper iRule

The idea of this helper iRule is for it to be installed on your helper virtual server and it will intercept requests to the helper virtual server and send the traffic to a destination based on a HTTP headers. This means that you can use a single helper virtual server and send traffic to any required destination. Port 443 is the default, though you can specify this if required eg 10.67.182.10:8443.

You can also specify the SNAT address to use for the request

```
when HTTP_REQUEST {
    # Create HTTP request
    set options [list headers [list "X-SS-Destination" "10.67.182.10" "X-SS-Snat"
"10.67.182.3"]]
    set response [call /Common/simple_sideband::http_req "/Common/https_helper"
"/mgmt/tm/ltm/virtual" $options ]
    if { [lindex $response 0] == 200 } {
        HTTP::respond 200 content $response
    } else {
        HTTP::respond 500 content $response
    }
}
```

Simple Sideband User Instructions

Peter White Version 1.2 17th July 2023

Examples

UDP

Sending iRule:

```
when HTTP_REQUEST {
    # Create DNS request
    set q [call /Common/simple_sideband::dns_query "www.example.com" ]
    binary scan $q H* qhex
    log local0.debug "query:$qhex"
    # Send request to DNS server
    set response [call /Common/simple_sideband::udp_req 10.67.182.10:53 $q
{recv_bytes 1} ]
    if { [lindex $response 0] == 0 } {
        # Successful response - decode the DNS esponse
        set data [lindex $response 1]
        binary scan $data H* data_hex
        log local0.debug "Success!: $data_hex"
        set r [call /Common/simple_sideband::dns_response $data ]
        HTTP::respond 200 content $r
    } else {
        HTTP::respond 500 content {failure}
    }
}
```

Answer output:

```
# curl 10.67.182.40
{18766 32776 1 1 0 0} {www.example.com 1 1} {{1 1 12 1.2.3.4}} {} {}
```

Logs:

```
Jul 14 08:54:26 simple-sideband-bigip1.pwhite debug tmm[30491]: Rule
/Common/sideband_test <HTTP_REQUEST>:
query:494e00080001000000000000037777770765786616d706c6503636f6d0000010001
Jul 14 08:54:27 simple-sideband-bigip1.pwhite debug tmm[30491]: Rule
/Common/sideband_test <HTTP_REQUEST>: Success!:
494e80080001000100000000037777770765786616d706c6503636f6d0000010001c00c0001000
10000000c000401020304
```

Additional output:

```
curl 10.67.182.40
{38749 32776 1 2 0 1} {www.example.com 1 1} {{1 1 12 1.2.3.4} {1 1 12
1.2.3.5}} {} {{2 1 12 {1 51 1 52 1 53 1 54 0}}}
```

Logs:

```
Jul 14 09:05:10 simple-sideband-bigip1.pwhite debug tmm[30491]: Rule
/Common/sideband_test <HTTP_REQUEST>:
query:975d00080001000000000000037777770765786616d706c6503636f6d0000010001
Jul 14 09:05:10 simple-sideband-bigip1.pwhite debug tmm[30491]: Rule
/Common/sideband_test <HTTP_REQUEST>: Success!:
975d80080001000200000001037777770765786616d706c6503636f6d0000010001c00c0001000
10000000c000401020304c00c000100010000000c000401020305c00c000200010000000c0009
013301340135013600
```

Simple Sideband User Instructions

Peter White Version 1.2 17th July 2023

Simple Sideband User Instructions

Peter White Version 1.2 17th July 2023

TCP

Sending iRule:

```
when HTTP_REQUEST {
    # Create HTTP request
    set response [call /Common/simple_sideband::tcp_req 10.67.182.10:80 "GET
/\r\n" {recv_bytes 3} ]
    if { [lindex $response 0] == 0 } {
        HTTP::respond 200 content $response
    } else {
        HTTP::respond 500 content $response
    }
}
```

Command output:

```
# curl 10.67.182.40
0 {HTTP/1.0 200 OK
Server: BigIP
Connection: close
Content-Length: 20

hello world! port:80}
```

HTTP

Sending iRule:

```
when HTTP_REQUEST {
    # Create HTTP request
    set response [call /Common/simple_sideband::http_req 10.67.182.10:80 "/"
{} ]
    if { [lindex $response 0] == 200 } {
        HTTP::respond 200 content $response
    } else {
        HTTP::respond 500 content $response
    }
}
```

Command output:

```
# curl 10.67.182.40
200 {Server BigIP Connection Keep-Alive Content-Length 20} {hello world!
port:80}
```

Simple Sideband User Instructions

Peter White   Version 1.2   17th July 2023

HTTPS

Helper VS:

```
ltm virtual https_helper {
    creation-time 2023-07-14:09:23:06
    destination 0.0.0.0:search-agent
    ip-protocol tcp
    last-modified-time 2023-07-14:09:23:06
    mask 255.255.255.255
    pool https_pool
    profiles {
        http { }
        serverssl {
            context serverside
        }
        tcp { }
    }
    serverssl-use-sni disabled
    source 0.0.0.0/0
    source-address-translation {
        type automap
    }
    translate-address enabled
    translate-port enabled
    vlans-enabled
    vs-index 3
}
```

Sending iRule:

```
when HTTP_REQUEST {
    # Create HTTP request
    set response [call /Common/simple_sideband::http_req
"/Common/https_helper" "/" {} ]
    if { [lindex $response 0] == 200 } {
        HTTP::respond 200 content $response
    } else {
        HTTP::respond 500 content $response
    }
}
```

Command output:

```
# curl 10.67.182.40
200 {Server BigIP Connection Keep-Alive Content-Length 21} {hello world!
port:443}
```

Simple Sideband User Instructions

Peter White Version 1.2 17th July 2023

License

Support

There is no support inherent in this code – you use it at your own risk. However, if you find bugs then you are welcome to contact me and I will try to resolve them on a best efforts basis. If you would like features, feel free to contact me and I may implement them, or refer you to F5 Professional Services.