```
when RULE_INIT {
           set static::airgap_ssl_bypass_categories {
                      /Common/Games
           }

}

when CLIENT_ACCEPTED {
    set hsl [HSL::open -proto UDP -pool syslog_server_poo]
           # DEBUG On/Off : 1/0
           set DEBUG 1
           # disable client/serverside ssl profile by default
           HTTP::disable
           SSL::disable clientside
           SSL::disable serverside

           # Check bypass or intercept
           # 1. Bypass-DIP for destination IP
           # 2. Bypass-SIP for source IP
           # 3. Intercept-DIP for destionation IP
           #  - trigger TCP::collect to extract SNI
           # 4. Intercept-SIP for source IP
           #  - trigger TCP::collectto extract SNI
           # 5. Bypass-Host for SNI (Only triggered after Intercept-SIP/DIP match)
           #   - SNI is matched from Bypass-Host datagroup -> bypass
           #   - SNI is matched from Intercept-Host datagroup -> intercetp
           #   - Otherwise intercept traffic

           #if { $DEBUG } { log local0. "[IP::client_addr] -> [IP::local_addr]" }
                      if { [class match -name -- [IP::local_addr] equals Bypass-DIP
                      ] ne "" } {
                      if { $DEBUG } { HSL::send $hsl "class match result : [class
                      match -name -- [IP::local_addr] equals --Bypass-DIP ]" }
                      #if { $DEBUG } { log local0. "Run Destination IP bypass:
                      Destination IP address is registered on bypass list:
                      [IP::local_addr]" }

           #} elseif { [class match -name -- [IP::client_addr] equals Bypass-SIP ]
           ne "" } {
                      #if { $DEBUG } { log local0. "class match result : [class
                      match -name -- [IP::client_addr] equals Bypass-SIP ]" }
                      #if { $DEBUG } { log local0. "Run Source IP bypass: Source IP
                      address is registered on bypass list: [IP::client_addr]" }
           #}
           #if { [class match -name -- [IP::local_addr] equals Intercept-DIP ] ne ""
           } {
                      #if { $DEBUG } { log local0. "Destionation IP address is
                      registred on Intercept-DIP list" }
                      #if { $DEBUG } { log local0. "class match result : [class
                      match -name -- [IP::local_addr] equals Intercept-DIP ]" }

                      # run TCP collect to check SNI for bypass before Intercept-DIP
                      SSL traffic
                      # log local0. "run client collect command"
                      TCP::collect
                      set monitor_id [\
                      after 500 {
                                 TCP::release
                                 #if { $DEBUG } { log local0.
                                 "[IP::client_addr]:[TCP::client_port]-[IP::local_addr
                                 ]:[TCP::local_port] -No Delayed Binding"}
                      }\
                      ]
           } elseif { [class match -name -- [IP::client_addr] equals Intercept-SIP ]
           ne "" } {

                      #if { $DEBUG } { log local0. "Source IP address is registred
                      on Intercept-SIP list" }
                      #if { $DEBUG } { log local0. "class match result : [class
                      match -name -- [IP::client_addr] equals Intercept-SIP ]" }
                      # run TCP collect to check SNI for bypass before intercept SSL
                      traffic
                      # log local0. "run client collect command"
```

```
57                          TCP::collect
58                          set monitor_id [\
59                          after 500 {
60                                      TCP::release
61                                      #if { $DEBUG } { log local0.
                                        "[IP::client_addr]:[TCP::client_port]-[IP::local_addr
                                        ]:[TCP::local_port] -No Delayed Binding"}
62                          }\
63                          ]
64              }
65  }
66
67  when CLIENT_DATA {
68              after cancel $monitor_id
69              binary scan [TCP::payload] cSS tls_xacttype tls_version tls_recordlen
70              if { ( $tls_xacttype == 23 ) or ( $tls_xacttype == 20 ) or ($tls_xacttype
                == 22)} {
71                          set record_offset 43
72                          binary scan [TCP::payload] @${record_offset}c tls_sessidlen
73                          set record_offset [expr {$record_offset + 1 + $tls_sessidlen}]
74                          binary scan [TCP::payload] @${record_offset}S tls_ciphlen
75                          set record_offset [expr {$record_offset + 2 + $tls_ciphlen}]
76                          binary scan [TCP::payload] @${record_offset}c tls_complen
77                          set record_offset [expr {$record_offset + 1 + $tls_complen}]
78                          if { ([TCP::payload length] > $record_offset) } {
79                                      binary scan [TCP::payload] @${record_offset}S
                                        tls_extenlen
80                                      set record_offset [expr {$record_offset + 2}]
81                                      binary scan [TCP::payload] @${record_offset}a*
                                        tls_extensions
82                                      for { set x 0 } { $x < $tls_extenlen } { incr x 4 } {
83                                                  set start [expr {$x}]
84                                                  binary scan $tls_extensions @${start}SS
                                                    etype elen
85                                                  if { ($etype == "00") } {
86                                                              set grabstart [expr {$start
                                                                + 9}]
87                                                              set grabend [expr {$elen - 5}]
88                                                              binary scan $tls_extensions
                                                                @${grabstart}A${grabend}
                                                                tls_servername
89                                                              set start [expr {$start +
                                                                $elen}]
90                                                  } else {
91                                                              # Bypass all other TLS
                                                                extensions.
92                                                              set start [expr {$start +
                                                                $elen}]
93                                                  }
94                                                  set x $start
95                                      }
96                          }
97                          if { ([info exists tls_servername]) } {
98                                                  #if { $DEBUG } { log local0.
                                                    "[IP::client_addr]:[TCP::client_port] -
                                                    [IP::local_addr]:[TCP::local_port]
                                                    $tls_servername"}
99                                      if { [class match $tls_servername contains
                                        Intercept-Host] } {
100                                                 #if { $DEBUG } { log local0. "class
                                                    match result : [class match -name --
                                                    $tls_servername equals Intercept-Host
                                                    ]" }
101                                                 #if { $DEBUG } { log local0. "RUN
                                                    Intercept-Host: SNI is matched with
                                                    Intercept-Host: $tls_servername" }
102                                                 #virtual VS_443_Proxy-3
103                                                 SSL::enable clientside
104                                             SSL::enable serverside
105                                         HTTP::enable
106                                          } elseif { [class match $tls_servername contains
                                            Bypass-Host] } {
107                                                 if { $DEBUG } { HSL::send $hsl
```

```
                                                "[IP::client_addr]:[TCP::client_port]
                                                -> [IP::local_addr]:[TCP::local_port]
                                                $tls_servername --Bypass-Host"}
108                                             #if { $DEBUG } { log local0. "Run
                                                Bypass-Host: SNI is matched with
                                                Bypass-Host" }
109
110                                     } else {
111                                             set this_uri http://$tls_servername/
112                                             set reply [getfield [CATEGORY::lookup
                                                $this_uri] " " 1]
113                                             set decision [lsearch -exact
                                                $static::airgap_ssl_bypass_categories
                                                $reply]
114                                             if {[lsearch -exact
                                                $static::airgap_ssl_bypass_categories
                                                $reply] >= 0}{
115                                                     set ssl_bypass_mitm 1
116                                                     if { $DEBUG } { HSL::send
                                                        $hsl
                                                        "[IP::client_addr]:[TCP::clien
                                                        t_port] ->
                                                        [IP::local_addr]:[TCP::local_p
                                                        ort] $tls_servername
                                                        --Bypass_Category"}
117                                             } else {
118                                                     set ssl_bypass_mitm 0
119                                             }
120                                             if { [info exists ssl_bypass_mitm] } {
121                                                     if { $ssl_bypass_mitm } {
122                                                             #log local0.
                                                                "$static::airgap
                                                                _ssl_bypass_cate
                                                                gories $reply"
123                                                             #log local0.
                                                                "$reply"
124                                                     } else {
125                                                             SSL::enable
                                                                clientside
126                                                             SSL::enable
                                                                serverside
127                                                             HTTP::enable
128                                                     }
129                                             }
130                                     }
131                             } else {
132                                     #if { $DEBUG } { log local0. "No SNI exist -> Run
                                        Intercept by Intercept-SIP or Intercpet-DIP" }
133                                     SSL::enable clientside
134                                     SSL::enable serverside
135                                     HTTP::enable
136                             }
137                     }
138                     TCP::release
139     }
140
141     when CLIENTSSL_HANDSHAKE {
142         if { $DEBUG } { HSL::send $hsl "[IP::client_addr]:[TCP::client_port] ->
            [IP::local_addr]:[TCP::local_port] $tls_servername --Decryption"}
143             LB::detach
144             SSL::disable serverside
145             virtual VS_443_Proxy-2
146     }
147
```