

Access-control with IPv4 forwarding in *NetASM*

```
1 .decls
2   # Constants
3   PORT_COUNT = 16
4   ETH_IP_TYPE = 0x0800
5   IP_TCP_PRTCL = 0x06
6   IP_UDP_PRTCL = 0x11
7
8   # Tables
9   # Access-control list
10  acl_table_size = 2048
11  acl_match_table =
12    ([ (ipv4_src, 32, Binary)
13      , (ipv4_dst, 32, Binary)
14      , (tcp_src, 16, Binary)
15      , (tcp_dst, 16, Binary)
16      , (udp_src, 16, Binary)
17      , (udp_dst, 16, Binary) ]
18    , acl_table_size
19    , TableType.CAM)
20
21  acl_params_table =
22    ([ (acl_action, 2) ]
23    , acl_table_size
24    , TableType.RAM)
25
26  # IPv4
27  ipv4_table_size = 4096
28  ipv4_match_table =
29    ([ (ipv4_dst, 32, Binary) ]
30    , ipv4_table_size
31    , TableType.CAM)
32
33  ipv4_params_table =
34    ([ (ipv4_action, 1)
35      , (ipv4_eth_src, 32)
36      , (ipv4_eth_dst, 32)
37      , (ipv4_outport_bitmap, PORT_COUNT) ]
38    , ipv4_table_size
39    , TableType.RAM)
40
41 .code
42 #####
43 ## Parse packet ##
44 #####
45
46 # Add Ethernet header fields
47 ADD eth_dst, 48
48 ADD eth_src, 48
49 ADD eth_type, 16
50
51 # Add IP header fields
52 ADD has_ip, 1
53 ADD ipv4_ver, 4
54 ADD ipv4_ihl, 4
55 ADD ipv4_dscp, 6
56 ADD ipv4_ecn, 2
57 ADD ipv4_tlen, 16
58 ADD ipv4_id, 16
59 ADD ipv4_flgs, 3
60 ADD ipv4_fo, 13
61 ADD ipv4_ttl, 8
62 ADD ipv4_prtcl, 8
63 ADD ipv4_chksm, 16
64 ADD ipv4_src, 32
65 ADD ipv4_dst, 32
66
67 # Add TCP header fields
68 ADD has_tcp, 1
69 ADD tcp_src, 16
70 ADD tcp_dst, 16
71
72 # Add UDP header fields
73 ADD has_udp, 1
74 ADD udp_src, 16
75 ADD udp_dst, 16
76
77 # Load fields with default values
78 LD eth_dst, 0
79 LD eth_src, 0
80 LD eth_type, 0
81
82 LD has_ip, 0
83 LD ipv4_ver, 0
84 LD ipv4_ihl, 0
85 LD ipv4_dscp, 0
86 LD ipv4_ecn, 0
87 LD ipv4_tlen, 0
88 LD ipv4_id, 0
89 LD ipv4_flgs, 0
90 LD ipv4_fo, 0
91 LD ipv4_ttl, 0
92 LD ipv4_prtcl, 0
93 LD ipv4_chksm, 0
94 LD ipv4_src, 0
95 LD ipv4_dst, 0
96
97 LD has_tcp, 0
98 LD tcp_src, 0
99 LD tcp_dst, 0
100
101 LD has_udp, 0
102 LD udp_src, 0
103 LD udp_dst, 0
104
105 # Parse Etherent
106 LD eth_dst, (0, 48)
107 LD eth_src, (48, 48)
108 LD eth_type, (96, 16)
109
110 # Check if the Ethernet payload is IP
111 BR eth_type, Eq, ETH_IP_TYPE, 'LBL_PARSE_0'
112
113 # Case: not an IP packet
114 CTR "PARSER:NOT_AN_IP_PACKET"
115 JMP "LBL_HLT"
116
```

```

117 # Parse IP
118 LBL 'LBL_PARSE_0'
119
120 # Note: we assume that ip has no optional
      fields
121 LD has_ip, 1
122 LD ipv4_ver, (112, 4)
123 LD ipv4_ihl, (116, 4)
124 LD ipv4_dscp, (120, 6)
125 LD ipv4_ecn, (126, 2)
126 LD ipv4_tlen, (128, 16)
127 LD ipv4_id, (144, 16)
128 LD ipv4_flgs, (160, 3)
129 LD ipv4_fo, (163, 13)
130 LD ipv4_ttl, (176, 8)
131 LD ipv4_prtcl, (184, 8)
132 LD ipv4_chksm, (192, 16)
133 LD ipv4_src, (208, 32)
134 LD ipv4_dst, (240, 32)
135
136 # Add and load ipv4_ihl_bits field
137 ADD ipv4_ihl_bits, 16
138 OP ipv4_ihl_bits, ipv4_ihl, Mul, 32
139
140 # Check if the IP payload is TCP
141 BR ipv4_prtcl, Neq, IP_TCP_PRTCL, '
      LBL_PARSE_1'
142
143 # Case: IP payload is TCP
144 # Add tcp_offset field
145 ADD tcp_offset, 16
146
147 # Load TCP header fields from the packet
148 OP tcp_offset, 160, Add, ipv4_ihl_bits
149 LD tcp_src (tcp_offset, 16)
150 OP tcp_offset, tcp_offset, Add, 16
151 LD tcp_dst (tcp_offset, 16)
152 LD has_tcp, 1
153
154 JMP 'LBL_ACL'
155
156 # Case: IP payload is not TCP
157 LBL 'LBL_PARSE_1'
158
159 # Check if the IP payload is UDP
160 BR ipv4_prtcl, Neq, IP_UDP_PRTCL, '
      LBL_PARSE_2'
161
162 # Case: IP payload is UDP
163 # Add udp_offset field
164 ADD udp_offset, 16
165
166 # Load UDP header fields from the packet
167 OP udp_offset, 160, Add, ipv4_ihl_bits
168 LD udp_src (udp_offset, 16)
169 OP udp_offset, udp_offset, Add, 16
170 LD udp_dst (udp_offset, 16)
171 LD has_udp, 1
172
173 JMP 'LBL_ACL'
174
175 # Case: unhandled IP payload
176 LBL 'LBL_PARSE_2'
177
178 CTR 'PARSER:UNHANDLED_IP_PAYLOAD'
179 JMP 'LBL_HLT'
180
181 #####
182 ## Simple 5-tuple ACL ##
183 #####

```

```

184 LBL 'LBL_ACL'
185
186 ADD acl_index, log2(acl_table_size)
187
188 LKt acl_index, acl_match_table, [ipv4_src,
      ipv4_dst, tcp_src, tcp_dst, udp_src,
      udp_dst]
189 BR acl_index, Neq, -1, 'LBL_ACL_0'
190
191 # Case: entry not present in the ACL match
      table
192 CTR 'ACL:MATCH_TABLE_MISS'
193 JMP 'LBL_HLT'
194
195 # Case: found an entry in the ACL match table
196 LBL 'LBL_ACL_0'
197
198 ADD acl_action, 2
199
200 # Load action parameters from the ACL params
      table
201 LDt [acl_action], acl_params_table, acl_index
202
203 # Check if ACL action is drop packet
204 BR acl_action, Gt, 0, 'LBL_ACL_1'
205
206 # Case: if drop packet
207 DRP 'ACL:DROP_PACKET'
208 JMP 'LBL_HLT'
209
210 # Case: if not drop packet
211 LBL 'LBL_ACL_1'
212
213 # Check if ACL action is intercept packet
214 BR acl_action, Gt, 1, 'LBL_ACL_2'
215
216 # Case: if intercept packet
217 CTR 'ACL:INTERCEPT_PACKET'
218 JMP 'LBL_HLT'
219
220 # Case: if not intercept packet
221 LBL 'LBL_ACL_2'
222
223 # Check if ACL action is allow packet
224 BR acl_action, Neq, 2, 'LBL_ACL_3'
225
226 # Case: if allow packet
227 JMP 'LBL_IPv4'
228
229 # Case: if not allow packet
230 LBL 'LBL_ACL_3',
231
232 CTR 'ACL:PACKET_NOT_ALLOWED'
233 JMP 'LBL_HLT'
234
235 #####
236 ## IPv4 forwarding ##
237 #####
238 LBL 'LBL_IPv4'
239
240 ADD ipv4_index, log2(ipv4_table_size)
241
242 LKt ipv4_index, ipv4_match_table, [ipv4_dst]
243 BR ipv4_index, Neq, -1, 'LBL_IPv4_0'
244
245 # Case: entry not present in the IPv4 match
      table
246 CTR 'IPv4:MATCH_TABLE_MISS'
247 JMP 'LBL_HLT'
248

```

<pre> 249 # Case: found an entry in the IP match table 250 LBL 'LBL_IPv4_0' 251 252 ADD ipv4_action, 1 253 ADD ipv4_eth_src, 32 254 ADD ipv4_eth_dst, 32 255 256 # Load parameters from the IP params table 257 LDt [ipv4_action, ipv4_eth_src, ipv4_eth_dst, ipv4_outport_bitmap], ipv4_params_table , ipv4_index 258 259 # Check if ipv4_action is send-to-controller for local subnet addresses needing ARP 260 BR ipv4_action, Gt, 0, 'LBL_IPv4_1' 261 262 # Case: if send to controller 263 CTR 'IPv4:SEND_TO_CONTROLLER' 264 JMP 'LBL_HLT' 265 266 # Case: if not send-to-controller 267 LBL 'LBL_IPv4_1' 268 269 # Check if ipv4_action is unicast/multicast 270 BR ipv4_action, Neq, 1, 'LBL_IPv4_2' 271 272 # Case: if unicast/multicast 273 OP ipv4_ttl, ipv4_ttl, Sub, 1 274 LD eth_src, ipv4_eth_src 275 LD eth_dst, ipv4_eth_dst 276 277 JMP 'LBL_WRITE_BACK' 278 </pre>	<pre> 279 # Case: if not multicast/unicast 280 LBL 'LBL_IPv4_2' 281 282 CTR 'IPv4:PACKET_NOT_UNI/MULTICAST' 283 JMP 'LBL_HLT' 284 285 ##### 286 ## Write-back to packet ## 287 ##### 288 LBL 'LBL_WRITE_BACK' 289 290 # Check if packet is ip -- this check is redundant as we only let ip packets in. 291 BR has_ip, Neq, 1, 'LBL_HLT' 292 293 # Compute new ipv4 checksum 294 CRC ipv4_chksm, [ipv4_ver, ipv4_ihl, ipv4_dscp, ipv4_ecn 295 ,ipv4_tlen, ipv4_id, ipv4_flgs, ipv4_fo 296 ,ipv4_ttl, ipv4_prctl, ipv4_src, ipv4_dst] 297 298 ST (0, 48), eth_src 299 ST (48, 48), eth_dst 300 ST (176, 8), ipv4_ttl 301 ST (192, 16), ipv4_chksm 302 303 ##### 304 ## Halt ## 305 ##### 306 LBL 'LBL_HLT' 307 HLT </pre>
---	---

Listing 1: A program for Access-control and IPv4 forwarding in NetASM.