Access-control with IPv4 forwarding in *NetASM*

```
.decls
  # Constants
  PORT_COUNT = 16
  ETH_IP_TYPE = 0x0800
  IP_TCP_PRTCL = 0x06
  IP_UDP_PRTCL = 0x11

  # Tables
  # Access-control list
  acl_table_size = 2048
  acl_match_table =
    ([(ipv4_src, 32, Binary)
     ,(ipv4_dst, 32, Binary)
     ,(tcp_src, 16, Binary)
     ,(tcp_dst, 16, Binary)
     ,(udp_src, 16, Binary)
     ,(udp_dst, 16, Binary)]
     ,acl_table_size
     ,TableType.CAM)

  acl_params_table =
    ([(acl_action, 2)]
    ,acl_table_size
    ,TableType.RAM)

  # IPv4
  ipv4_table_size = 4096
  ipv4_match_table =
    ([(ipv4_dst, 32, Binary)]
     ,ipv4_table_size
     ,TableType.CAM)

  ipv4_params_table =
    ([(ipv4_action, 1)
     ,(ipv4_eth_src, 32)
     ,(ipv4_eth_dst, 32)
     ,(ipv4_outport_bitmap, PORT_COUNT)]
    ,ipv4_table_size
    ,TableType.RAM)

.code
  ##################
  ## Parse packet ##
  ##################

  # Add Ethernet header fields
  ADD eth_dst, 48
  ADD eth_src, 48
  ADD eth_type, 16

  # Add IP header fields
  ADD has_ip, 1
  ADD ipv4_ver, 4
  ADD ipv4_ihl, 4
  ADD ipv4_dscp, 6
  ADD ipv4_ecn, 2
  ADD ipv4_tlen, 16
  ADD ipv4_id, 16
  ADD ipv4_flgs, 3
  ADD ipv4_fo, 13
  ADD ipv4_ttl, 8
  ADD ipv4_prtcl, 8
  ADD ipv4_chksm, 16
  ADD ipv4_src, 32
  ADD ipv4_dst, 32

  # Add TCP header fields
  ADD has_tcp, 1
  ADD tcp_src, 16
  ADD tcp_dst, 16

  # Add UDP header fiedls
  ADD has_udp, 1
  ADD udp_src, 16
  ADD udp_dst, 16

  # Load fields with default values
  LD eth_dst, 0
  LD eth_src, 0
  LD eth_type, 0

  LD has_ip, 0
  LD ipv4_ver, 0
  LD ipv4_ihl, 0
  LD ipv4_dscp, 0
  LD ipv4_ecn, 0
  LD ipv4_tlen, 0
  LD ipv4_id, 0
  LD ipv4_flgs, 0
  LD ipv4_fo, 0
  LD ipv4_ttl, 0
  LD ipv4_prtcl, 0
  LD ipv4_chksm, 0
  LD ipv4_src, 0
  LD ipv4_dst, 0

  LD has_tcp, 0
  LD tcp_src, 0
  LD tcp_dst, 0

  LD has_udp, 0
  LD udp_src, 0
  LD udp_dst, 0

  # Parse Etherent
  LD eth_dst, (0, 48)
  LD eth_src, (48, 48)
  LD eth_type, (96, 16)

  # Check if the Ethernet payload is IP
  BR eth_type, Eq, ETH_IP_TYPE, 'LBL_PARSE_0'

  # Case: not an IP packet
  CTR "PARSER:NOT_AN_IP_PACKET"
  JMP "LBL_HLT"
```

```
# Parse IP
LBL 'LBL_PARSE_0'

# Note: we assume that ip has no optional fields
LD has_ip, 1
LD ipv4_ver, (112, 4)
LD ipv4_ihl, (116, 4)
LD ipv4_dscp, (120, 6)
LD ipv4_ecn, (126, 2)
LD ipv4_tlen, (128, 16)
LD ipv4_id, (144, 16)
LD ipv4_flgs, (160, 3)
LD ipv4_fo, (163, 13)
LD ipv4_ttl, (176, 8)
LD ipv4_prtcl, (184, 8)
LD ipv4_chksm, (192, 16)
LD ipv4_src, (208, 32)
LD ipv4_dst, (240, 32)

# Add and load ipv4_ihl_bits field
ADD ipv4_ihl_bits, 16
OP ipv4_ihl_bits, ipv4_ihl, Mul, 32

# Check if the IP payload is TCP
BR ipv4_prtcl, Neq, IP_TCP_PRTCL, 'LBL_PARSE_1'

# Case: IP payload is TCP
# Add tcp_offset field
ADD tcp_offset, 16

# Load TCP header fields from the packet
OP tcp_offset, 160, Add, ipv4_ihl_bits
LD tcp_src (tcp_offset, 16)
OP tcp_offset, tcp_offset, Add, 16
LD tcp_dst (tcp_offset, 16)
LD has_tcp, 1

JMP 'LBL_ACL'

# Case: IP payload is not TCP
LBL 'LBL_PARSE_1'

# Check if the IP payload is UDP
BR ipv4_prtcl, Neq, IP_UDP_PRTCL, 'LBL_PARSE_2'

# Case: IP payload is UDP
# Add udp_offset field
ADD udp_offset, 16

# Load UDP header fields from the packet
OP udp_offset, 160, Add, ipv4_ihl_bits
LD udp_src (udp_offset, 16)
OP udp_offset, udp_offset, Add, 16
LD udp_dst (udp_offset, 16)
LD has_udp, 1

JMP 'LBL_ACL'

# Case: unhandled IP payload
LBL 'LBL_PARSE_2'

CTR 'PARSER:UNHANDLED_IP_PAYLOAD'
JMP 'LBL_HLT'

########################
## Simple 5-tuple ACL ##
########################
LBL 'LBL_ACL'

ADD acl_index, log2(acl_table_size)

LKt acl_index, acl_match_table, [ipv4_src, ipv4_dst, tcp_src, tcp_dst, udp_src, udp_dst]
BR acl_index, Neq, -1, 'LBL_ACL_0'

# Case: entry not present in the ACL match table
CTR 'ACL:MATCH_TABLE_MISS'
JMP 'LBL_HLT'

# Case: found an entry in the ACL match table
LBL 'LBL_ACL_0'

ADD acl_action, 2

# Load action parameters from the ACL params table
LDt [acl_action], acl_params_table, acl_index

# Check if ACL action is drop packet
BR acl_action, Gt, 0, 'LBL_ACL_1'

# Case: if drop packet
DRP 'ACL:DROP_PACKET'
JMP 'LBL_HLT'

# Case: if not drop packet
LBL 'LBL_ACL_1'

# Check if ACL action is intercept packet
BR acl_action, Gt, 1, 'LBL_ACL_2'

# Case: if intercept packet
CTR 'ACL:INTERCEPT_PACKET'
JMP 'LBL_HLT'

# Case: if not intercept packet
LBL 'LBL_ACL_2'

# Check if ACL action is allow packet
BR acl_action, Neq, 2, 'LBL_ACL_3'

# Case: if allow packet
JMP 'LBL_IPv4'

# Case: if not allow packet
LBL 'LBL_ACL_3',

CTR 'ACL:PACKET_NOT_ALLOWED'
JMP 'LBL_HLT'

####################
## IPv4 forwarding ##
####################
LBL 'LBL_IPv4'

ADD ipv4_index, log2(ipv4_table_size)

LKt ipv4_index, ipv4_match_table, [ipv4_dst]
BR ipv4_index, Neq, -1, 'LBL_IPv4_0'

# Case: entry not present in the IPv4 match table
CTR 'IPv4:MATCH_TABLE_MISS'
JMP 'LBL_HLT'
```

```
# Case: found an entry in the IP match table
LBL 'LBL_IPv4_0'

ADD ipv4_action, 1
ADD ipv4_eth_src, 32
ADD ipv4_eth_dst, 32

# Load parameters from the IP params table
LDt [ipv4_action, ipv4_eth_src, ipv4_eth_dst,
     ipv4_outport_bitmap], ipv4_params_table
    , ipv4_index

# Check if ipv4_action is send-to-controller
    for local subnet addresses needing ARP
BR ipv4_action, Gt, 0, 'LBL_IPv4_1'

# Case: if send to controller
CTR 'IPv4:SEND_TO_CONTROLLER'
JMP 'LBL_HLT'

# Case: if not send-to-controller
LBL 'LBL_IPv4_1'

# Check if ipv4_action is unicast/multicast
BR ipv4_action, Neq, 1, 'LBL_IPv4_2'

# Case: if unicast/multicast
OP ipv4_ttl, ipv4_ttl, Sub, 1
LD eth_src, ipv4_eth_src
LD eth_dst, ipv4_eth_dst

JMP 'LBL_WRITE_BACK'

# Case: if not multicast/unicast
LBL 'LBL_IPv4_2'

CTR 'IPv4:PACKET_NOT_UNI/MULTICAST'
JMP 'LBL_HLT'

#########################
## Write-back to packet ##
#########################
LBL 'LBL_WRITE_BACK'

# Check if packet is ip -- this check is
    redundant as we only let ip packets in.
BR has_ip, Neq, 1, 'LBL_HLT'

# Compute new ipv4 checksum
CRC ipv4_chksm, [ipv4_ver, ipv4_ihl,
    ipv4_dscp, ipv4_ecn
                ,ipv4_tlen, ipv4_id,
                    ipv4_flgs, ipv4_fo
                ,ipv4_ttl, ipv4_prtcl,
                    ipv4_src, ipv4_dst]

ST (0, 48), eth_src
ST (48, 48), eth_dst
ST (176, 8), ipv4_ttl
ST (192, 16), ipv4_chksm

##########
## Halt ##
##########
LBL 'LBL_HLT'
HLT
```

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