APPENDIX A

Programming Code

```
!/usr/bin/python
# standard includes

from pox.lib.addresses import IPAddr
#from pox.lib.packet import *
from pox.core import core
from pox.lib.util import dpidToStr
import pox.openflow.libopenflow_01 as of
import time

# include as part of the betta branch
from pox.openflow.of_json import *

log = core.getLogger()

# handler for timer function that sends the requests to all the
# switches connected to the controller

def _timer_func ():
  for connection in core.openflow._connections.values():
    connection.send(of.ofp_stats_request(body=of.ofp_flow_stats_request()))
    connection.send(of.ofp_stats_request(body=of.ofp_port_stats_request()))
  log.debug("Sent %i flow/port stats request(s)", len(core.openflow._connections))



def _handle_flowstats_received (event):


  stats = flow_stats_to_list(event.stats)
  log.debug("FlowStatsReceived from %s: %s",
  dpidToStr(event.connection.dpid), stats)

  def drop (duration = None):
    """
    Drops this packet and optionally installs a flow to continue
    dropping similar ones for a while
    """

    if duration is not None:
      if not isinstance(duration, tuple):
        duration = (duration,duration)
      msg = of.ofp_flow_mod()
      msg.match = of.ofp_match.from_packet(packet)
      msg.idle_timeout = duration[0]
      msg.hard_timeout = duration[1]
```

```
      msg.buffer_id = event.ofp.buffer_id
      self.connection.send(msg)
   # elif event.ofp.buffer_id is not None:
    #  msg = of.ofp_packet_out()
     # msg.buffer_id = event.ofp.buffer_id
     # msg.in_port = event.port
     # self.connection.send(msg)
  # Get number of bytes/packets in flows for web traffic only
  ipList = []
#  poList = []
  p_n = ""
  count = 0
  n_flows = 0
  T1 = 0
  T2 = 0

  for f in event.stats:
    p_n = str(f.match.nw_proto)
    count += f.packet_count
    n_flows += 1
    ipList.append(str(f.match.nw_dst))
  #  poList.append(str(f.match.tp_dst))
    T1 = time.time()
    T2 = time.time()
    TE = T2-T1

    if count >= 1500:
      print(count)
      for i in range(1, len(ipList)):
        if ipList [i] == ipList[i-1]:
         if TE <= 30:
#            print("Attack is detected")
            log.warning("Attack is detected from %s on %s through %s.  Drop it."
             % (f.match.nw_src, f.match.nw_dst, f.match.nw_proto))
            print(ipList)
            print("time elapsed between packets:", TE)
            drop()
            return

#       print(ipList)
#      print(len(ipList))
#         print("time elapsed between packets:", TE)

    elif n_flows <= 10:
#        for j in range(1, len(poList)):
#         if poList [j] == poList[j-1]:
```

Attack Detection

Attack Mitigation

```
            print("a scanning attack is detected")
            print(n_flows)
            #print(poList)

            for connection in core.openflow.connections:
               connection.send(of.ofp_flow_mod(command=of.OFPFC_DELETE_STRICT,
 action=of.ofp_action_output(port=3),priority=32,
 match=of.ofp_match(dl_type=0x800,nw_dst="10.0.0.3")))
#           print(n_flows)
#           print(poList)
 #dl_type=0x800
# handler to display port statistics received in JSON format
#def _handle_portstats_received (event):
 # stats = flow_stats_to_list(event.stats)
  #log.debug("PortStatsReceived from %s: %s",
   # dpidToStr(event.connection.dpid), stats)

# main functiont to launch the module
def launch ():
  from pox.lib.recoco import Timer

  # attach handsers to listners

  core.openflow.addListenerByName("FlowStatsReceived", _handle_flowstats_received)
#  core.openflow.addListenerByName("PortStatsReceived",
 #   _handle_portstats_received)

  # timer set to execute every five seconds
  Timer(5, _timer_func, recurring=True)

#  core.registerNew(Detect_Mitigate_Func)
```