```
--
-- required packages for this script
--
local bin = require "bin"
local nmap = require "nmap"
local shortport = require "shortport"
local stdnse = require "stdnse"
local string = require "string"
local table = require "table"

--Usage:
--Identify MELSEC-Q Series PLC CPUINFO
--nmap -script melsecq-discover.nse -sT -p 5007 <host>

--Output Example:
--PORT     STATE SERVICE                REASON
--5007/tcp open  Mitsubishi/Melsoft TCP syn-ack
--| melsecq-discover:
--|_  CPUINFO: Q03UDECPU


description = [[
discovery Mitsubishi Electric Q Series PLC
        GET CPUINFO
]]

author = "ICS Security Workspace(plcscan.org)"
license = "Same as Nmap--See http://nmap.org/book/man-legal.html"
categories = {"discovery","intrusive"}

function set_nmap(host, port)
        port.state = "open"
        port.version.name = "Mitsubishi/Melsoft TCP"
        port.version.product = "Mitsubishi Q PLC"
        nmap.set_port_version(host, port)
        nmap.set_port_state(host, port, "open")

 end

function send_receive(socket, query)
        local sendstatus, senderr = socket:send(query)
        if(sendstatus == false) then
    return "Error Sending getcpuinfopack"
        end
        local rcvstatus,response = socket:receive()
        if(rcvstatus == false) then
        return "Error Reading getcpuinfopack"
        end
        return response
        end

portrule = shortport.port_or_service(5007, "Melsoft/TCP", "tcp")
action = function(host,port)
        local getcpuinfopack =
bin.pack("H","5700000000111107000ffff030000fe03000014001c080a080000000000000004" .. "0101" ..
"010000000001")
        local response
        local output = stdnse.output_table()
        local sock = nmap.new_socket()
        local constatus,conerr = sock:connect(host,port)
        if not constatus then
    stdnse.print_debug(1,
      'Error establishing connection for %s - %s', host,conerr
      )
```

```
    return nil
        end
        response  = send_receive(sock, getcpuinfopack)
        local mel, pack_head = bin.unpack("C", response, 1)
--      local mel, space_id = bin.unpack("C", response, 55)
        local offset = 0
        if ( pack_head == 0xd7) then
--              if ( space_id == 0x20) then
                local mel
                local mel, cpuinfo = bin.unpack("z", response, 42 + offset)
                output["CPUINFO"] = string.sub(cpuinfo, 1, 16)
                set_nmap(host, port)
                sock:close()
                return output
--      end
        else
        sock:close()
    return nil

        end


end
```