Analisi delle performance ed espressività di regole per Intrusion Detection System generate in linguaggio LUA per scenari industriali

Edoardo Marinelli (matricola n. 0000970527)

Obiettivo della tesi

- Creare **regole Suricata** per ambiti industriali
- Implementare metodi per ampliarne l’**espressività**
- Migliorare la qualità dei **file di log** tradizionali
- Creare un ambiente di testing **master-slave Modbus**

- Protocollo di comunicazione sviluppato originariamente per i PLC negli anni '70
- Due varianti: **Modbus Seriale** (basato su comunicazione seriale come RS-485) e **Modbus TCP** (basato su Ethernet e Internet)
- Struttura Master-Slave: dove lo slave (client) invia dati al master (server).
- Implementato grazie a **PyModbus**

- Intrusion Detection System
- Basato su delle regole

```
alert modbus any any -> any any (msg:"Too much
CPU usage/ high temperature/high humidity.
Possible attack!!!"; luajit:prova1.lua;
threshold: type threshold, track by_src,
count 2, seconds 180; sid:103; rev:1;)
```

- Permette l’integrazione di script Lua

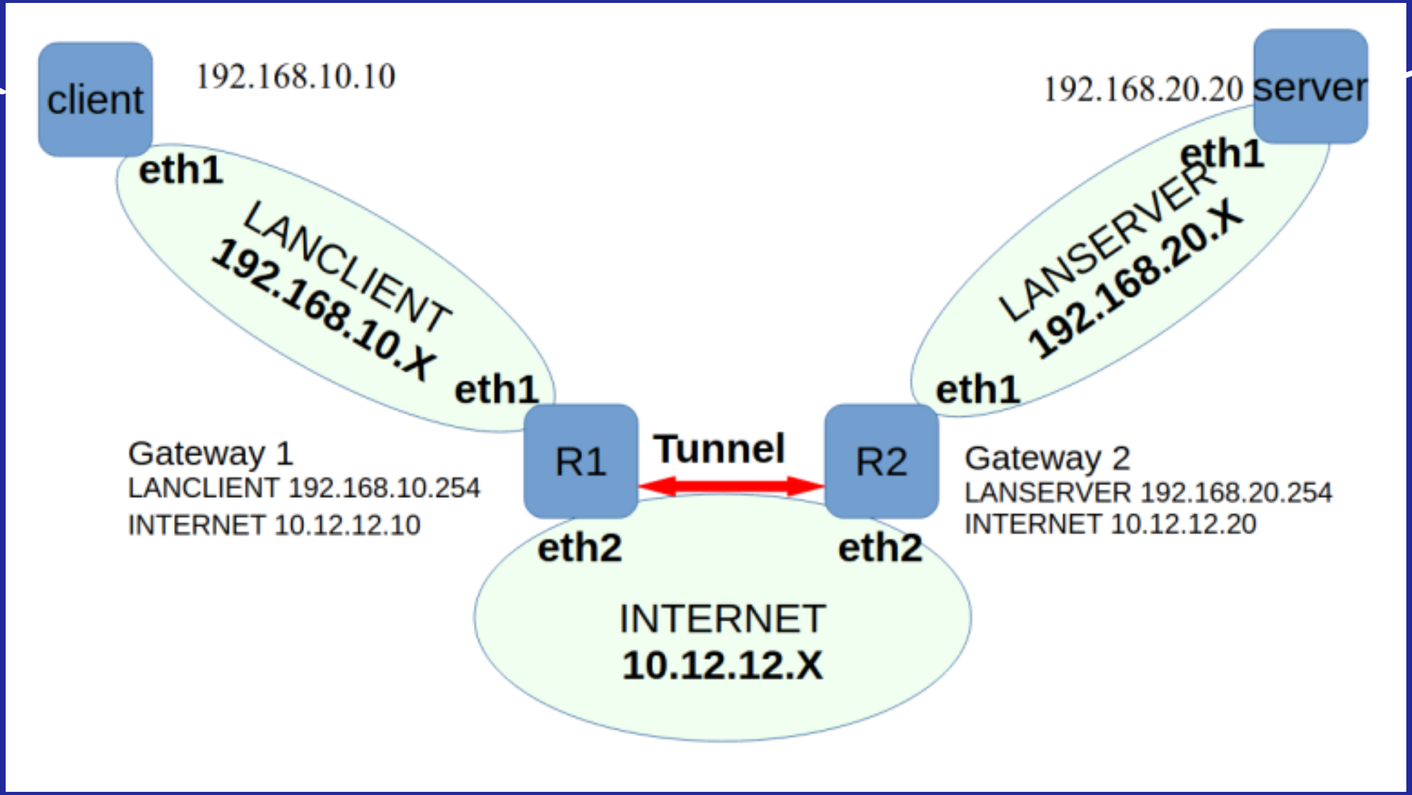
- Pienamente supportato da Suricata
- **Ispezione del payload** e **generazione file di log personalizzati**

```lua
function init (args)
    local needs = {}
    needs["payload"] = tostring(true)
    return needs
end

function match(args)
    local a = args["payload"]

    local reversed_payload = a:reverse()
    local reversed_temperature = string.match(reversed_payload, "(%d+) :erutarepmeT")

    local pressure = nil
    if reversed_temperature then
        temperature = reversed_temperature:reverse()
    end

    local reversed_cpu = string.match(reversed_payload, "(%d+) :egasU UPC")

    local cpu = nil
    if reversed_cpu then
        cpu = reversed_cpu:reverse()
    end

    local reversed_humidity = string.match(reversed_payload, "(%d+) :ytidimuH")

    local humidity = nil
    if reversed_humidity then
        humidity = reversed_humidity:reverse()
    end
    cpu1 = tonumber(cpu)
    temp1 = tonumber(temperature)
    hum1 = tonumber(humidity)

    if cpu1 > 60 and temp1 > 50 and hum1 > 40 then
        return 1
    end
    return 0
end

return 0
```

```lua
function init (args)
    local needs = {}
    needs["type"] = "packet"
    needs["filter"] = "alerts"
    return needs
end

function setup (args)
    filename = SCLogPath() .. "/" .. "ceftest.cef"
    file = assert(io.open(filename, "a"))
    SCLogInfo("ModbusLuaTest Log Filename " .. filename)
    packet = 1
end

function log(args)
    timestring = SCPacketTimeString()
    ip_version, src_ip, dst_ip, protocol, src_port, dst_port = SCPacketTuple()
    msg = SCRuleMsg()
    p = SCPacketPayload()
    sid, rev, gid = SCRuleIds()
    class, prio = SCRuleClass()
    local handle = io.popen("suricata -V | awk '{print $5}'")
    local result = handle:read("*a")
    str = result:gsub("\n", "")
    handle:close()
    if protocol == 6 then
        proto = "TCP"
    end
    if dst_port ~= 502 then
    return
    end
     if p == "" then
        return
     end
    file:write ("CEF:0|OISF|Suricata|" .. str .. "|" .. rev .. ":" .. sid .. ":" .. gid ..
        "|Modbus Communication|" .. prio .. "| rt=" .. timestring .. " act=alert" ..
        " proto=" .. proto .. " src=" .. src_ip .. " spt=" .. src_port ..
        " dst=" .. dst_ip .. " dpt= " .. dst_port  .. " msg=" .. msg .. "\n\n")
    file:flush()

    packet = packet + 1
end

function deinit (args)
    SCLogInfo ("Modbus transactions logged: " .. packet);
    file:close(file)
end
```

Infrastruttura di testing

Client
PyModbus



Server
PyModbus

S.O. Kali Linux insieme a macchina virtuale Vagrant

Suricata Keywords vs Lua Scripting

Keyword “content”: analisi grossolana del payload

Script Lua: possibilità di trattare valori dopo stringhe

```
Humidity: 94%
CPU Usage: 32%
Noise level: 8dB
Pressure: 28hPa
Temperature: 48C
Noise level: 58dB
```

fast.log vs luatest.log

```
vagrant@client:/vagrant/test$ sudo tail -f
/var/log/suricata/fast.log
10/10/2023-15:18:03.831135  [**] [1:103:1]
Too much CPU usage/high temperature/high hu
midity. Possible attack!!! [**] [Classifica
tion: (null)] [Priority: 3] {TCP} 192.168.1
0.10:39538 -> 192.168.20.20:502
```

```
ALERT DETAILS
Too much CPU usage/high temperature/high humidity. Possible attack!!!
sid: 103, rev: 1, gid: 1, Priority: 3
N:4 | Timestamp: 08/31/2023-17:11:57.713252 | IPv4 | Protocol: TCP
| Source/Destination: 192.168.10.10:54334 -> 192.168.20.20:502
| CRITICAL PAYLOAD
CPU Usage: 77%
Temperature: 64°
Humidity: 53%

TOTAL PAYLOAD:

O{:t
CPU Usage: 77%
Pressure: 1hPa
Humidity: 71%
Temperature: 64C
gljoaqngs
Humidity: 1%
Humidity: 53%
bagicjhulcvjbkt
```

- Completa personalizzazione
- Migliore espressività dei dati
- Possibilità di espansione in altri ambiti

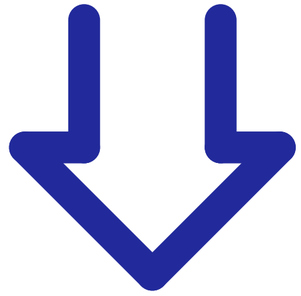
Supporto al formato CEF

```
vagrant@client:~$ sudo tail -f /var/log/suricata/ceftest.cef
CEF:0|OISF|Suricata|6.0.1|1:103:1|Modbus Communication|3| rt=09/01/2023-10:24:54.644893 act=alert proto=TCP src=192.
168.10.10 spt=60414 dst=192.168.20.20 dpt= 502 msg=Too much CPU usage/high temperature/high humidity. Possible attac
k!!!
```

- Formato di **logging standardizzato**
- Supportato dalla maggior parte dei sistemi **SIEM**
- Facilita lo scambio di informazioni tra diversi strumenti e piattaforme di sicurezza

Obiettivi raggiunti

- Ampliamento dell’espressività delle regole Suricata
- Creazione file di log finalizzati all’analisi specifica del payload
- Aggiunta supporto a file di log in formato CEF



- Espansione verso altri formati di log (LEEF, ELF, ECS, ecc...)

Grazie dell’attenzione!