Progetto Gestione di Reti 2020/2021

Tranchida Marco 559737

# 1. Introduzione

Il seguente progetto è incentrato sull'implementazione di un *check* per *ntopng*, un software web-based per l'analisi del traffico di rete e flow collection.

Il check implementato ha il compito di controllare se il numero di pacchetti, inviati e ricevuti, da un host supera una certa soglia (*threshold*) e in tal caso notificare alla *gui* tramite un meccanismo di *alert*.

# 2. Configurazione del check & alert

Per poter utilizzare il check all'interno della gui di *ntopng* è necessario:

Spostare i seguenti file all'interno della cartella di *ntopng*

- *PktThreshold.h → ntopng/include/host_checks*
- *PktThreshold.cpp → ntopng/src/host_checks*
- *PktThresholdAlert.h → ntopng/include/host_alert*
- *PktThresholdAlert.cpp → ntopng/src/host_alert*
- *pkt_threshold.lua → ntopng/scripts/lua/modules/check_definitions/host/*
- *host_alert_pkt_threshold.lua → ntopng/scripts/lua/modules/alert_definitions/*

Modificare i seguenti file all'interno della cartella di *ntopng*

- *ntopng/include/ntop_typedefs.h*

```
typedef enum {
  host_alert_normal                    = 0,
  host_alert_smtp_server_contacts      = 1,
  host_alert_dns_server_contacts       = 2,
  host_alert_ntp_server_contacts       = 3,
  host_alert_flow_flood                = 4,
  host_alert_syn_scan                  = 5,
  host_alert_syn_flood                 = 6,
  host_alert_domain_names_contacts     = 7,
  host_alert_p2p_traffic               = 8,
  host_alert_dns_traffic               = 9,
  host_alert_flows_anomaly             = 10,
  host_alert_score_anomaly             = 11,
  host_alert_remote_connection         = 12,
  host_alert_host_log                  = 13,
  host_alert_dangerous_host            = 14,
  host_alert_ntp_traffic               = 15,
  host_alert_countries_contacts        = 16,
  host_alert_score_threshold           = 17,
  host_alert_icmp_flood                = 18,
  host_alert_pkt_threshold             = 19,

  MAX_DEFINED_HOST_ALERT_TYPE, /* Leave it as last member */
  MAX_HOST_ALERT_TYPE = 32 /* Constrained by HostAlertBitmap */
} HostAlertTypeEnum;
```

```
typedef enum {
  host_check_http_replies_requests_ratio = 0,
  host_check_dns_replies_requests_ratio,
  host_check_syn_flood,
  host_check_syn_scan,
  host_check_flow_flood,
  host_check_ntp_server_contacts,
  host_check_smtp_server_contacts,
  host_check_countries_contacts,
  host_check_dns_server_contacts,
  host_check_score_host,
  host_check_p2p_traffic,
  host_check_dns_traffic,
  host_check_flow_anomaly,
  host_check_score_anomaly,
  host_check_remote_connection,
  host_check_dangerous_host,
  host_check_ntp_traffic,
  host_check_domain_names_contacts,
  host_check_score_threshold,
  host_check_icmp_flood,
  host_check_pkt_threshold,

  NUM_DEFINED_HOST_CHECKS, /* Leave it as last member */
} HostCheckID;
```

- *ntopng/include/HostCheckStatus*

```
class HostChecksStatus { /* Container to keep per-c
consecutive calls) */
 private:
  time_t last_call_min, /* The last time minute che
    last_call_5min;     /* The last time 5minute ch
  u_int64_t ntp_bytes;  /* Holds the NTP bytes and
across consecutive check calls */
  u_int64_t p2p_bytes;  /* Holds the P2P bytes and
across consecutive check calls */
  u_int64_t dns_bytes;  /* Holds the DNS bytes and
across consecutive check calls */
  u_int64_t my_pkt_counter;
```

```
public:
 HostChecksStatus() {
   last_call_min = last_call_5min = 0;
   /* Set members to their maximum values to discard the first delta */
   ntp_bytes = p2p_bytes = dns_bytes = my_pkt_counter = (u_int64_t)-1;
 }
 virtual ~HostChecksStatus() {};

  inline bool isTimeToRunMinChecks(time_t now)  const { return last_call_min  +  60 <= now; }
  inline bool isTimeToRun5MinChecks(time_t now) const { return last_call_5min + 300 <= now; }

  inline void setMinLastCallTime(time_t now)  { last_call_min  = now; }
  inline void set5MinLastCallTime(time_t now) { last_call_5min = now; }

  /* Checks status API */
  inline u_int64_t cb_status_delta_ntp_bytes(u_int64_t new_value) { return
Utils::uintDiff(&ntp_bytes, new_value); };
  inline u_int64_t cb_status_delta_p2p_bytes(u_int64_t new_value) { return
Utils::uintDiff(&p2p_bytes, new_value); };
  inline u_int64_t cb_status_delta_dns_bytes(u_int64_t new_value) { return
Utils::uintDiff(&dns_bytes, new_value); };
  inline u_int64_t cb_status_delta_my_pkt_counter(u_int64_t new_value) { return
Utils::uintDiff(&my_pkt_counter, new_value); };
};
```

- *ntopng/src/HostChecksLoader.cpp*

```
void HostChecksLoader::registerChecks() {
  /* TODO: implement dynamic loading */
  HostCheck *fcb;

  if((fcb = new CountriesContacts()))          registerCheck(fcb);
  if((fcb = new FlowFlood()))                  registerCheck(fcb);
  if((fcb = new SYNScan()))                    registerCheck(fcb);
  if((fcb = new SYNFlood()))                   registerCheck(fcb);
  if((fcb = new DNSServerContacts()))          registerCheck(fcb);
  if((fcb = new SMTPServerContacts()))         registerCheck(fcb);
  if((fcb = new NTPServerContacts()))          registerCheck(fcb);
  if((fcb = new NTPTraffic()))                 registerCheck(fcb);
  if((fcb = new P2PTraffic()))                 registerCheck(fcb);
  if((fcb = new DNSTraffic()))                 registerCheck(fcb);
  if((fcb = new RemoteConnection()))           registerCheck(fcb);
  if((fcb = new DangerousHost()))              registerCheck(fcb);
  if((fcb = new DomainNamesContacts()))        registerCheck(fcb);
  if((fcb = new ScoreThreshold()))             registerCheck(fcb);
  if((fcb = new ICMPFlood()))                  registerCheck(fcb);
  if((fcb = new PktThreshold()))          registerCheck(fcb);
```

- *ntopng/include/host_checks_includes.h*

```
#include "host_checks/PktThreshold.h"
```

- *ntopng/scripts/locales/en.lua* aggiungere le seguenti stringhe

  ["host_alert_pkt_threshold"] = "Pkt Threshold"

  ["host_alert_pkt_threshold"] = "Packet threshold exceeded by %{entity} [%{value} %{op} %{threshold}]"

# 3. Descrizione dell'implementazione

Il check *Pkt_Threshold* effettua un controllo sugli host della rete contando i pacchetti inviati e ricevuti ogni minuto con la chiamata *PeriodicUpdate*. Se il numero pacchetti superano la soglia stabilita (pkts/s) tramite *gui,* scatta un allarme che verrà riportato nella sezione *Allarmi → Host* di *ntopng.*

| | | | | | |
|---|---|---|---|---|---|
| 17:29:49 | 100 | 02:01 | Pkt Threshold | marco-omen-by-hp-laptop → | Packet threshold exceeded by marco-omen-by-hp-laptop [4950 > 3000] |
| 17:40:52 | 100 | 05:24 | Pkt Threshold | marco-omen-by-hp-laptop → | Packet threshold exceeded by marco-omen-by-hp-laptop [3834 > 3000] |

Il *threshold* configurato è in *pkts/s* ma dato che il *PeriodicUpdate* avviene ogni minuto viene effettuata una media per stimare grossolanamente il valore in *pkts/s* dei pacchetti conteggiati al tempo d'esecuzione del check.