


Progetto Gestione di Reti 2020/2021  
Tranchida Marco 559737

# 1. Introduzione

Il seguente progetto è incentrato sull'implementazione di un *check* per *ntopng* come richiesto in <https://github.com/ntop/ntopng/issues/5906>.

## Packet size checks #5906

 Open lucaderi opened this issue on 20 Sep · 1 comment



lucaderi commented on 20 Sep

Member



Similar to other checks, implement a new check for spotting hosts that misbehave. Possible thresholds could be

0-128 byte > 30000 pkt/s

129-256 byte > 10000 pkt/s

or just have a single threshold (10000 pkt/s)

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-consecutive calls */
private:
    time_t last_call_min, /* The last time minute check */
    last_call_5min; /* The last time 5minute check */
    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 NPTraffic())) 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*.

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.

### 4. Check in funzione

Per dimostrare che il check implementato rispetti la specifica ho effettuato un test di verifica con un *threshold* di 4000 pkts/s.

Il test consiste nel controllare il numero di pacchetti dall'inizio fino al sorgere dell'alert e verificare che il numero conteggiato sia fedele al numero riportato nell'alert stesso.







Per fare ciò ho utilizzato *wireshark* per ottenere dei dati più precisi rispetto a alle informazioni offerte da *ntopng* per quanto riguarda numero preciso di pacchetti e temporizzazione.

Per far sì di poter superare la soglia di pacchetti ho messo in riproduzione uno stream in 4k e se il check funziona a dovere dovrei ricevere due alert, uno dall'host che sta ricevendo (il mio PC) e l'altro dall'host che trasmette (server).

Il test ha preso luogo alle 15:40:10 (orario di ricezione del primo pacchetto) e si è concluso alle 15:41:10 (orario di comparsa dell'alert).

No.	Time	Source	Orario	Destination
23	5.018484...	r1.sn-8vq54voxpu-2hve.googlevideo.com	15:40:10,716619533	marco-OMEN-by-HP-Laptop.local
308647	64.30190...	r1.sn-8vq54voxpu-2hve.googlevideo.com	15:41:10,000042184	marco-OMEN-by-HP-Laptop.local

Data/Ora	Score	Durata	Allarme	Host	Descrizione	Azioni
15:41:10	100	00:24	Pkt Threshold	marco-omen-by-hp-laptop →	Packet threshold exceeded by marco-omen-by-hp-laptop [5143 > 4000]	  
15:41:13	100	00:21	Pkt Threshold	r1--sn-8vq54voxpu-2hve.... →	Packet threshold exceeded by 91.80.48.12 [5263 > 4000]	  

Il check prende il numero di pacchetti al minuto e il *threshold*, moltiplica quest'ultimo per 60, e confronta in numero di pacchetti totali (rcvd+sent) con il *threshold*. Se il *threshold* viene superato restituisce all'alert il numero di pacchetti totale diviso 60 (approssimazione della media in pkts/s) e possiamo vedere il valore risultante direttamente nell'alert.

In questo caso il numero di pacchetti totale è all'incirca 308647 → /60 → 5144 che si avvicina abbastanza al valore presente nell'alert.