

## D.A.Q. SNORT'S MODULE INTEGRATION DUE THE EXPORTATION OF THE SECURITY ANALYSIS RESULTS.

*Francesco Crecchi*  
*451619*  
*a.a. 2012/2013*

### **Intro:**

Since version 2.9 of Snort (IDS/IPS opensource) has been decided to separate the packets acquisition module from the data security analysis in a stand-alone module named *daq*.

This module has been projected to analyze the network traffic, one packet at time, and to generate a *verdict* (positive/negative) based to the network traffic rules defined by the network administrator when the IDS/IPS was configured.

In particular the conformity match of the packet than the IDS/IPS rules is performed by the **pfring\_daq\_acquire()** function: here is created a verdict and ,basing on it, is decided what to do; if the verdict is *blacklist* the packet is dropped and a new filter rule is added to Snort to drop packets of the same flow.

### **The idea:**

The project is based on the assumption that in a real world context, where in complex networks more prober are used to capture network's traffic, is reasonable to have somewhat like **common consciousness**, i.e. seems to be obsolete the “isolated IDS/IPS” idea who decides and regulates the traffic basing only on it's own rules.

We felt the necessity of exporting the data generated by Snort with the objective to make a **smart-cloud** that others IDS/IPS/common hosts could interrogate to integrate their own idea of network's rules and traffic to make a better and more precise network's analysis.

### **Redis:**

Redis is a *NoSQL* database; *<key,value>* entry type.

It's reasonable to ask: “Why not use a traditional DB to export the packet's informations?”.

The question is really simple: *they doesn't scale!*

Redis instead works quite exclusively in ram: this feature allows a single server Redis that runs on a “commodity-hardware” Linux host to obtain the interesting performance of 100 000 query/sec.

Seems to be clear which of the two types of DB is the most indicated to track thousand packets per second!

### **The project:**

The project is based on a extension of the Snort's *daq* (data acquisition) module to export the analysis results in a Redis DB located directly on the host or to a remote host specified by the user.

In particular we are going to make two *sets* into the Redis DB named *Attackers* and *Targets*.

When the packet analysis match to blacklist means that we break some Snort's rules; basing on the assumption that (at least in 99% of the cases) the client (the attacker) is who starts the communication, we push the client IP address into the Attackers set and the server IP address into the Targets set, incrementing, every time that the key is referred, the value of the entry; this feature allows to make something like an “malware-detections hosts ranking” and “attacks-targets host ranking” of the net. Thinking on the collected data storage we decided to use the Redis *EXPIRETIME* feature *caching*

packets informations and invalidating them after a *TTL* if they are not referred newly.

### **The implementation:**

First has been introduced the possibility to starts the *daq module* with the ***redis*** option that allows the user to specify the *RedisServerIP* followed by the *RedisServerPort*, to let the user chose where to put the Redis Server: on the local host or in a remote host.

Second it's been modified the *daq module*, in particular the ***pfring\_daq\_acquire()*** function, who is delegated to dialog with the Snort module to make a verdict based on the analysis of the current packet. When the verdict is generated, it's checked if is blacklisted (*DAQ\_VERDICT\_BLACKLIST*) and the packet is parsed to extract the source IP address and the destination IP address.

As told before, making the assumption that the attacker's IP is the client's IP in 99% of the cases we proceed with the insert of the two IP informations above into two specified set, Attackers and Targets, calling the function ***pfring\_daq\_redis\_insert()*** that insert/update the key in the set incrementing the value associated and updating the relative expire time (*TTL*).

### **Conclusions:**

This quite simple code extension permits to add noteworthy functionality to the *daq module*, like the introduction of a common consciousness stored into the cloud that should be really important for:

- IDS/IPS who can build its' own rules basing on a *actual and real global vision* of the net.
- Host specialized in making network's stats that, extracting infos from the Redis DB, could easily generate an *ip ranking* of the most infected hosts on the net and who are often attacks object.
- Network Admins whom can *monitor the health status of the net*, sanitize the infected hosts and build stronger defense to protect the hosts that are often attacker's object.
- Evaluate, in case of many false-positive, *IDS/IPS rules too restrictive*.