Network Security Monitoring

Coleman Kane
Coleman.Kane@ge.com

September 24, 2014

Passive monitoring analyzes traffic with the intention of being non-disruptive to the transmission of the traffic. Typically used in environments where high-network-availability is critical. This is typical in many coporate environments where the network supports a considerably large user base. As such, there are numerous solutions out there which provide passive network and host monitoring.

Levels of Network Monitoring

There are various approaches to network monitoring which range from basic "flow data" to full packet capture, and even beyond.

Flow data Logs per-packet endpoint information, optionally including packet sizes. Common for aggregate reporting to determine activity anomolies between your network and specific external or internal hosts.

Transaction data Logs deeper connection-level information, which may span multiple packets within a connection. Must have pre-defined templates for protocol formatting. Common for logging HTTP header/request information, SMTP command data, etc.

Alert data Typically the result of finely-tuned signatures matching against packet content, and similar in nature to transaction data. This information, rather than being for logging purposes is intended to indicate discrete events which might be attacks.

Levels of Network Monitoring 2

Packet capture Full capture of packet-level contents (sometimes called **pcap**). This is roughly identical to what the OS sees on the wire

Reassembly Beyond packet-level capture, this functionality involves transport/application recognition and can reassemble IP packets into TCP stream data, or even parse application-level communications to rebuild a conversation, file transfer, or even email.

Per-packet statistics, sometimes including connection/packet-counting. The goal of this is to determine how much traffic (either counts or bytes) is **flowing** into or out of a particular host. Distributed across the network and combined with centralized aggregation of data from many sensors, you can build a picture of your network to help you identify "high traffic" entities which might indicate a problem (such as active data theft).

Flow Data Graphic

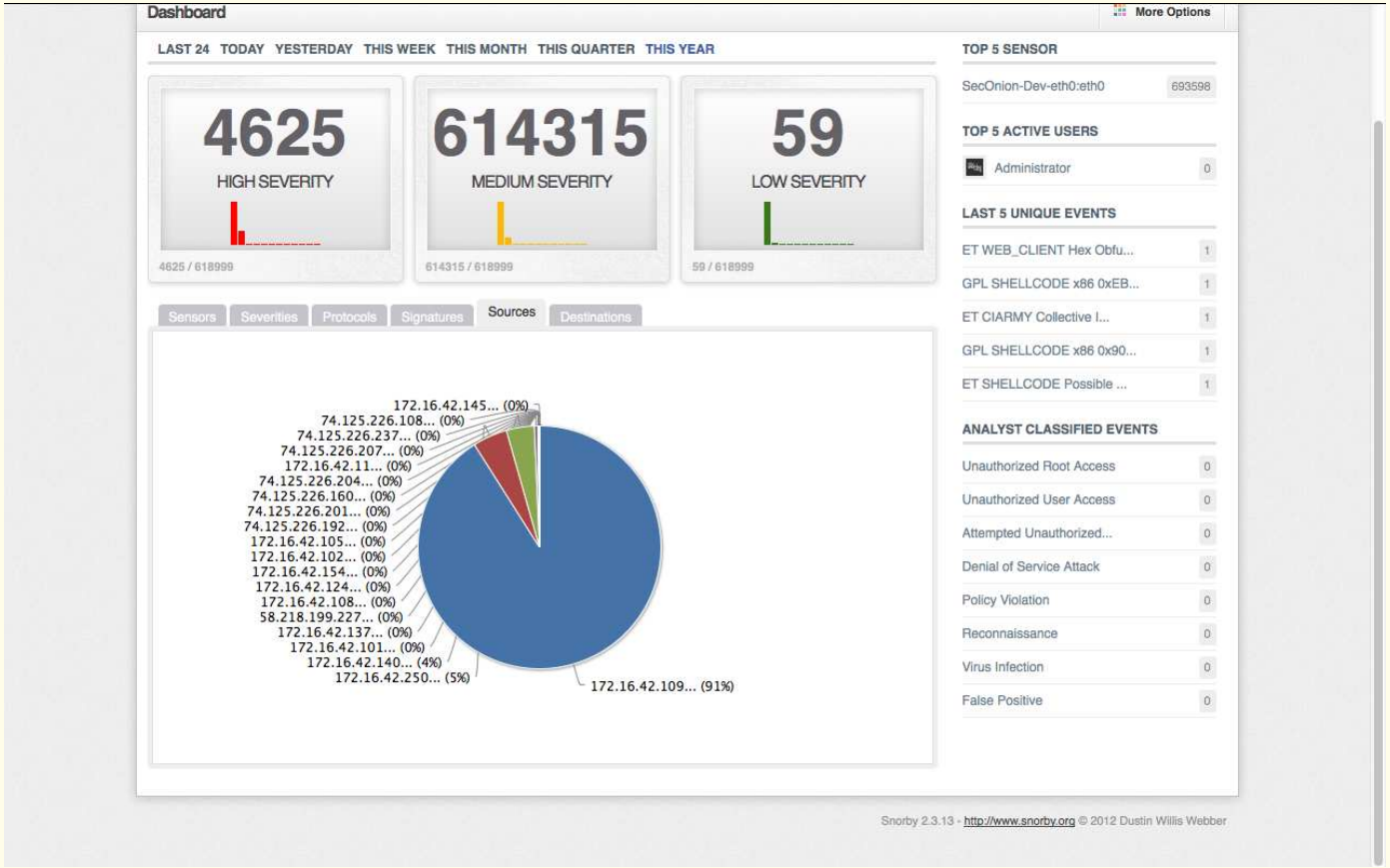

Figure 1: Flow Data in Snorby[2]

Example tools which capture/analyze flow data

cxtracker `http://www.gamelinux.org/?cat=21`
`https://github.com/gamelinux/cxtracker/`

SANCP `http://www.metre.net/sancp.html`

Cisco NetFlow Collector
`http://www.cisco.com/c/en/us/products/cloud-systems-management/netflow-collection-engine/index.html`

Arbor PeakFlow
`http://www.arbornetworks.com/products/peakflow/sp`

netstat With `-s`/`-statistics` option on Linux (`-m` on BSD & MacOS X) can report host-local flow information per interface. Endpoints could log this info at regular intervals to centralized system to build endpoint flow database.

RFC 3176: sFlow standard

Standard UDP-reporting mechanism for multiple devices to communicate "net flow" information amongst one another

- `http://www.sflow.org/`
- `http://www.faqs.org/rfcs/rfc3176.html`
- Products supporting the export and import of sFlow data: `http://www.sflow.org/products/index.php`

Transaction Data

Sometimes lumped into **flow data** category. However, we will make the distinction that transaction data requires sensor-level parsing that is traditionally not included in the base OS. For instance, transaction-level monitoring might involve parsing the **Referer** header from HTTP traffic, usernames from FTP or Telnet traffic, or any other metadata which must be parsed via a prepared recipe, but gets reported while associated traffic may be discarded after parsing is complete.

Example tools which handle/generate transaction data

httpry `http://dumpsterventures.com/jason/httpry/`

bro `https://www.bro.org/sphinx/httpmonitor/index.html`
`https://www.bro.org/sphinx/mimestats/index.html`

Colasoft Capsa `http://www.colasoft.com/capsa-free/`

Network Proxy Most proxies can do this for you for HTTP & FTP, sometimes more protocol support too

Transaction Data Graphic

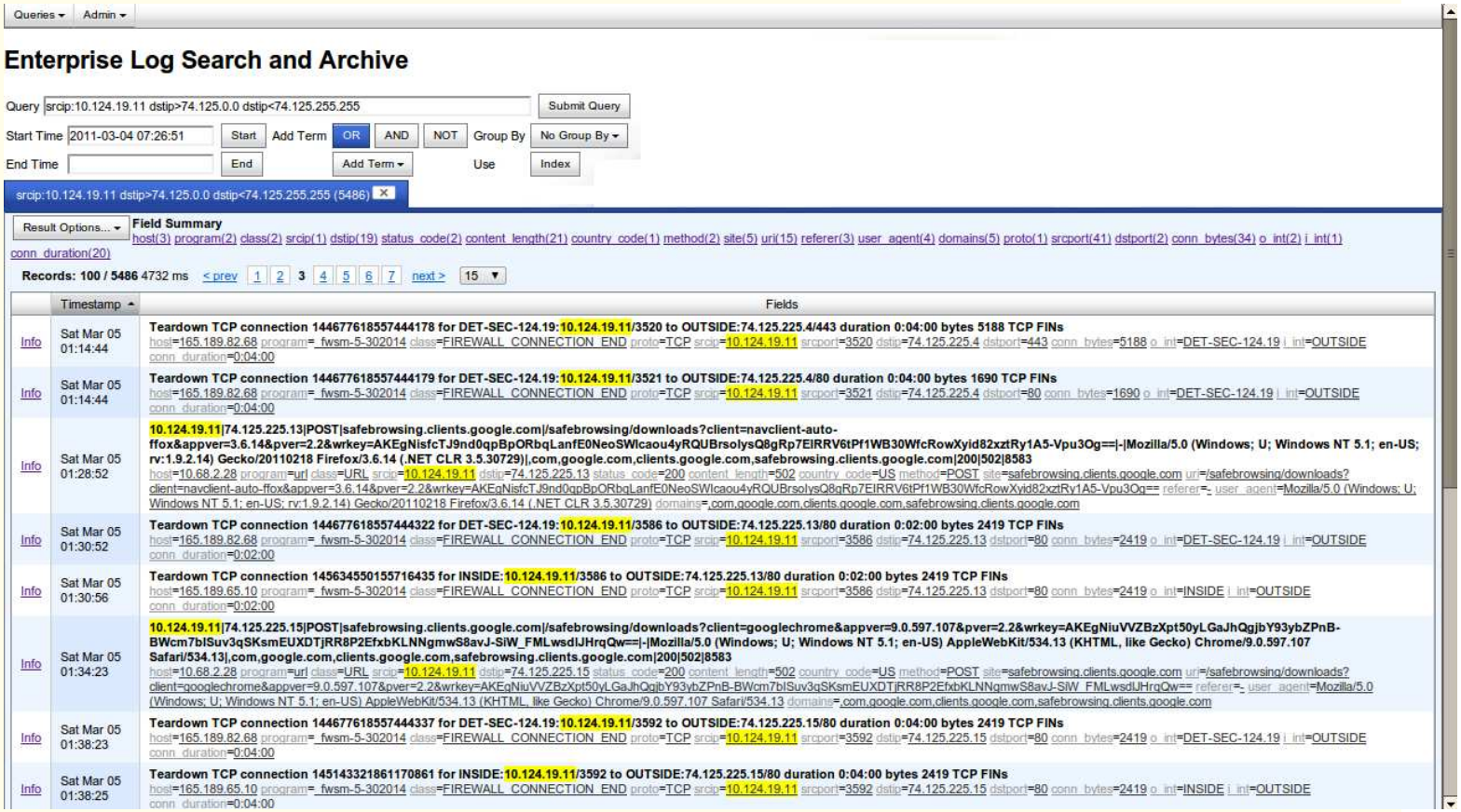

Figure 2: ELSA ScreenShot including HTTP transaction data[1]

Alert Data

Alert data is the monitoring data which is intended to be used to drive investigation & work. While Flow & Transactional data is intended to log a small amount of the total data from all network traffic (80/20 rule), alert data is intended to identify to analysts where/when occurred the specific network traffic which is highly suspicious, and necessitates being investigated.

- Identifies where to look in your transaction logs & packet capture archives quickly
- "Best guess" alert as to what attack might be happening

Snort One of the oldest signature-based packet analysis tools `https://www.snort.org/`

Suricata Newer signature-based packet analysis tool, similar to Snort `http://suricata-ids.org/`

Bro Newer versions have signature analysis capability on top of the transactional parsing capabilities `https://www.bro.org/`

SourceFire Now owned by Cisco Systems `http://www.sourcefire.com/`, based upon Snort

HP TippingPoint Firewal & IDS `http://www.tippingpoint.com`

Alert Management Tools

These tools can be used for managing response to alerts

RT Best Practical Request Tracker, has a module for incident response called "RTIR"

Sguil Very popular alert management/handling console `http://nsmwiki.org`

Commercial A lot of commercial IDS systems provide a custom console for managing alerts

ArcSight "SEIM" tool which can help manage alerts across multiple IDS/IPS tools

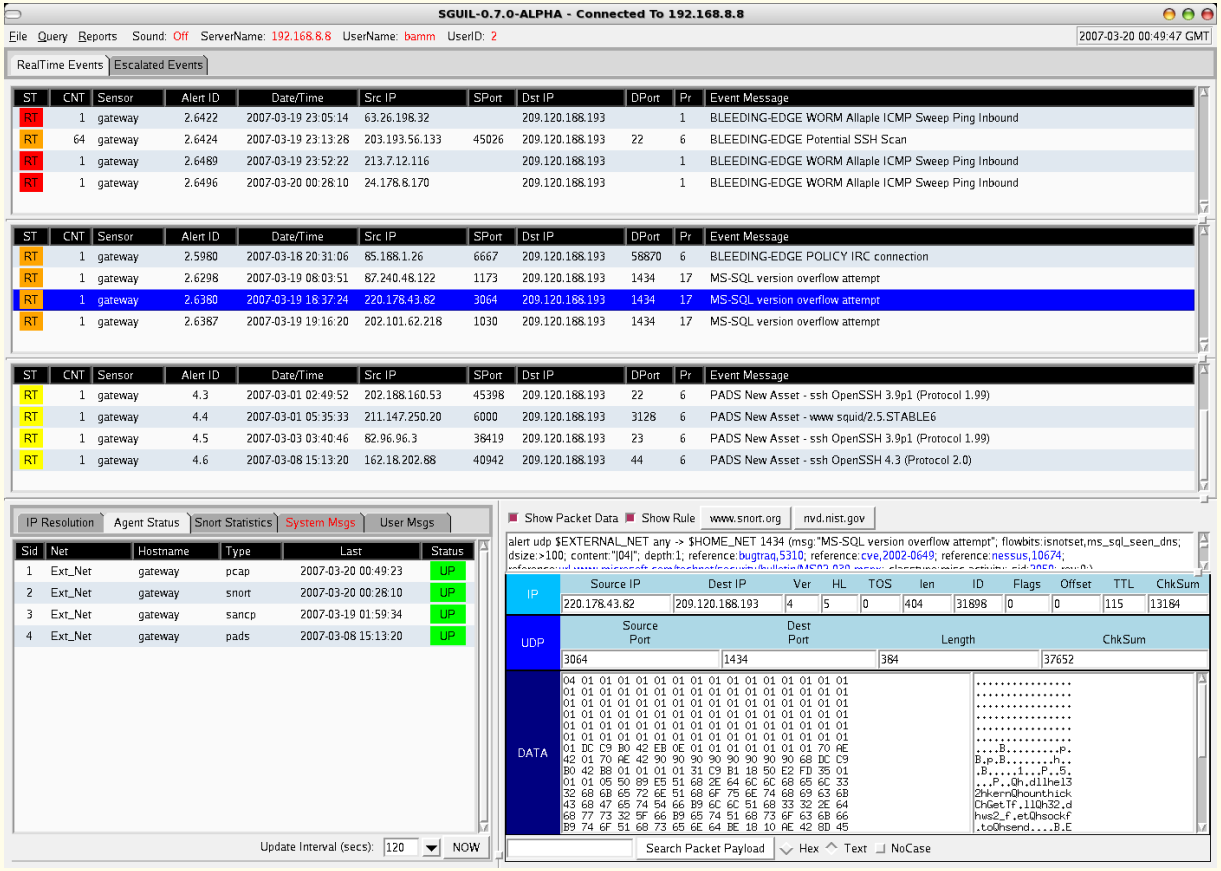

Figure 3: Sguil with a bunch of Snort alerts

Packet capture is the mechanism by which full or partial network traffic is archived for analysis. Some sensors implement continuous recording of traffic to a storage device, in the event that it needs to be retrieved in response to alerting in the future. This is commonly referred to as "**full packet capture**", and storage is typically managed in a FIFO.

Packet Capture Tools

You should already be familiar with:

- WireShark, tshark
- Tcpdump

Also:

- NetSniff-NG `http://netsniff-ng.org/`
- IPCopper `http://www.ipcopper.com/`

Packet Capture Purpose

Reasons for implementing packet capture:

- You might only learn of an attack after it happens
- Many alerts built for later-stages of intrusion, but you want to learn entire attack
- User traffic baselining
- Reference "good" traffic sampling

Stream Capture

Step above packet capture, and typically assists in stream analysis and forensics. Packet capture will have all data objects broken up into individual packets, and must be reassembled using network tools. Additionally, forensics on this data set need to be performed by a networking expert. Storing reassembled files, machine-to-machine conversations and other stream objects can improve forensic analysis and enable more participants to help analyze. Some attacks can only be identified at this level, too.

Bro NSM supports two capture methods:

- File capture/carving
- Stream capture carving
- Configurable filtering for both

In many cases, network monitoring doesn't come down to a single implementation from above, but rather a sensor software stack built out of multiple components to fill your network gaps.

Security Onion Linux

- Turn-key implementation of Ubuntu Linux to provide a complete software stack using the best open-source projects for monitoring
- `http://blog.securityonion.net/p/securityonion.html`
- Named "security onion" to reflect the many layers it is built from
- Easy, scripted install, and designed to run in a VirtualBox environment

References

[1] Fighting apt with open-source software, part 1: Logging. http://ossectools.blogspot.com/2011/03/fighting-apt-with-open-source-software.html, March 2011.

[2] Scott Runnels. Managing overactive signatures. https://code.google.com/p/security-onion/wiki/ManagingAlerts, January 2014.