This repository Search Pull requests Issues Gist

kamir / **Snaffer**
forked from nehme/Sniffer

Unwatch 1 Star 0 Fork 1

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

Python Sniffer — Edit

46 commits **1** branch **0** releases **2** contributors

Branch: **master** **New pull request** **Create new file** **Upload files** **Find file** **Clone or download**

This branch is 1 commit behind nehme:master. Pull request Compare

kamir committed on **GitHub Enterprise** Update README.md Latest commit `b4a23d3` an hour ago

doc	Add files via upload	an hour ago
pcapy-0.10.8	Added a bootstrap procedure to install pcapy and other dependencies a…	4 hours ago
schema	first commit of snaffer.py for TCP dumps in Avro format	9 hours ago
Log.log	Added a bootstrap procedure to install pcapy and other dependencies a…	4 hours ago
README.md	Update README.md	an hour ago
README.md~	Added a simple upload-procedure to README.	8 hours ago
bootstrap.sh	Fixed bootstrap	3 hours ago
bootstrap.sh~	Fixed bootstrap	3 hours ago
bootstrapHDFS.sh	Added bootstrap for HDFS	3 hours ago
snaff.sh	Fixed bootstrap for HDFS	3 hours ago
snaff.sh~	Fixed bootstrap for HDFS	3 hours ago
snaffer.py	Fixed timestamp issue	4 hours ago
snaffer.py~	Fixed timestamp issue	4 hours ago
snap-1.png	Added screenshot	2 hours ago
sniffer.py	initial commit	29 days ago

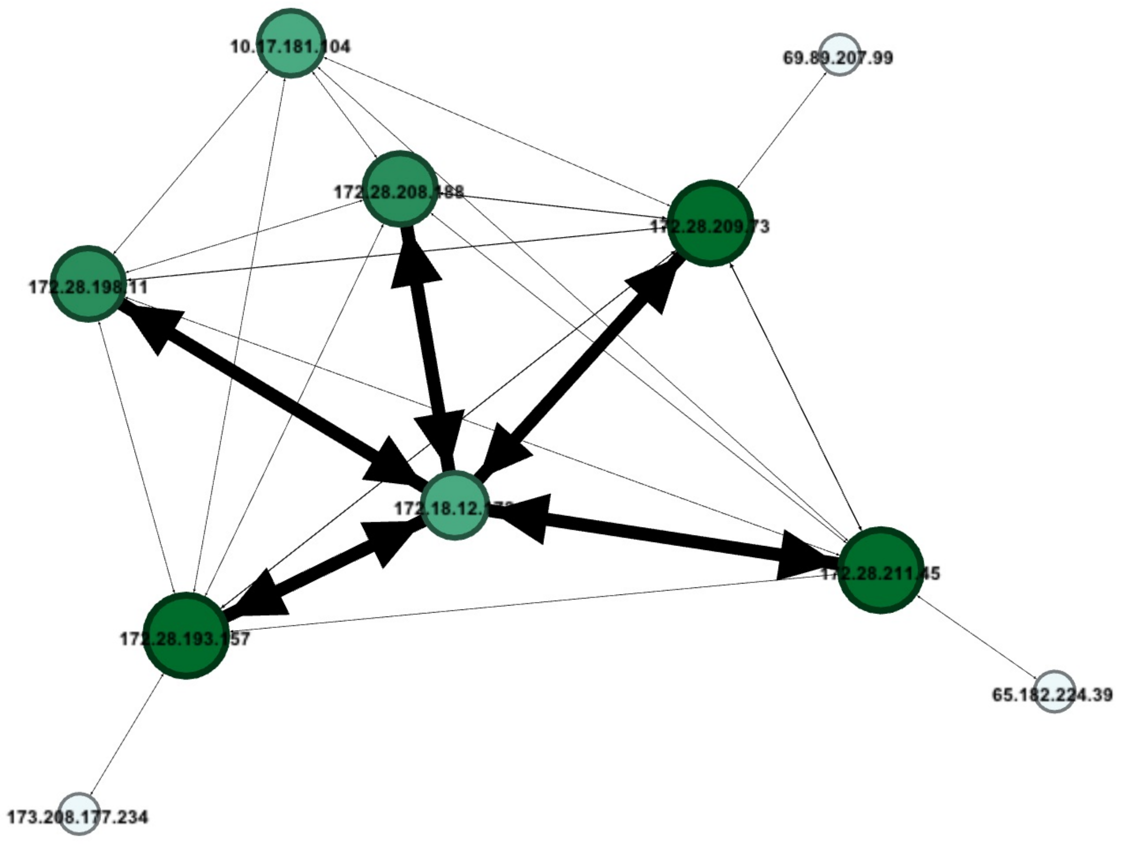
README.md

Our goal:

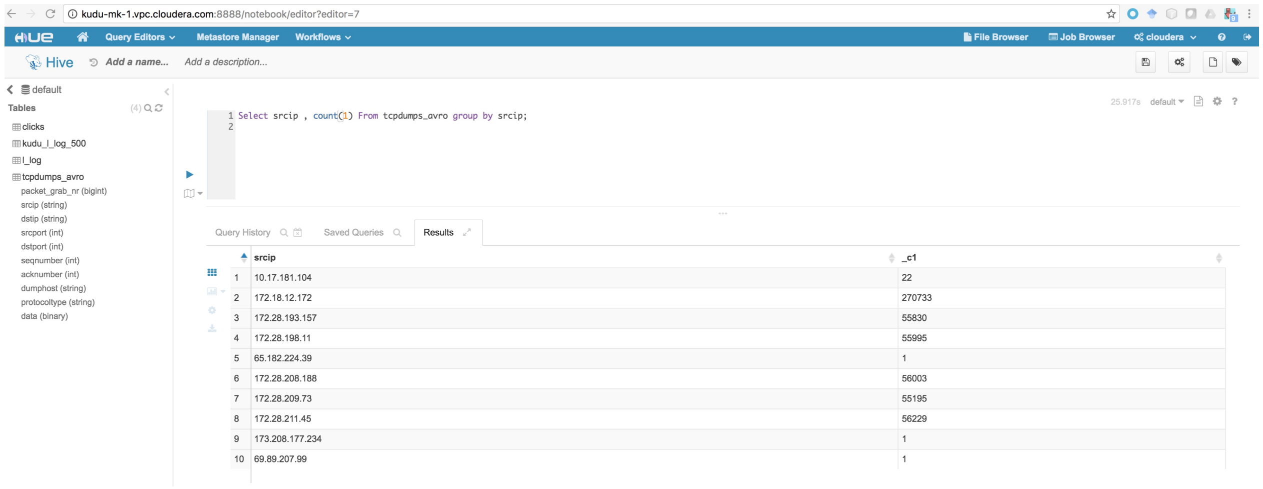
Packet inspection with Hive and Spark, analysis of the communication graph via GraphX and Gephi.

Results:

A communication graph shows the hotspots. Time dependent views allow application fine tuning.



Simple statistics is done via Hive.



```
Select srcip , count(1) From tcpdumps_avro group by srcip;
```

	srcip	_c1
1	10.17.181.104	22
2	172.18.12.172	270733
3	172.28.193.157	55830
4	172.28.198.11	55995
5	65.182.224.39	1
6	172.28.208.188	56003
7	172.28.209.73	55195
8	172.28.211.45	56229
9	173.208.177.234	1
10	69.89.207.99	1

More advanced packet content inspection follows soon ... (via Apache Spark).

Our Tool: Snaffer

A Python based TCP Sniffer which writes AVRO files for analysis in Spark or Impala.

Preparation: Install Dependencies

The following steps are executed by the bootstrap.sh script.

```
sudo yum install python-devel
sudo yum install gcc
sudo yum install gcc-c++
sudo yum install libpcap-devel
sudo easy_install pip
sudo pip install avro
sudo pip install pcapy
wget http://www.coresecurity.com/system/files/pcapy-0.10.8.tar_.gz
tar -xf pcapy-0.10.8.tar_.gz
cd pcapy-0.10.8
sudo python setup.py install
```

More details about pcapy: https://www.coresecurity.com/corelabs-research/open-source-tools/pcapy

TXT Output Format

```
sudo python sniffer.py eth0
```

```
srcIP|dstIP|protocolType|srcPort|dstPort|seqNumber|ackNumber|data
```

```
172.18.13.178|172.28.196.65|TCP|34|54985|7180|1436448910|3130769187|+$QGET /cmf/keepSessionActive?_=147515
172.28.196.65|172.18.13.178|TCP|34|7180|54985|3130769187|1436450264|+$R|
172.18.13.178|172.28.196.65|TCP|34|54985|7180|1436450264|3130769187|+$Q=(direct)|utmcmd=(none)|
172.28.196.65|172.18.13.178|TCP|34|7180|54985|3130769187|1436450291|+$R|
172.28.196.65|172.18.13.178|TCP|34|7180|54985|3130769187|1436450291|+$RHTTP/1.1 200 OKContent-Type: applica
```

Avro Output Format

```
sudo python snaffer.py eth0 1000
```

Collects 1000 packets and writes into an Avro file using the Avro schema in: schema/packet.asvc.

```
{
    "namespace":"com.cloudera.security.checks",
    "type":"record",
    "doc":"This Schema describes a DATA PACKET",
    "name":"Packet",
    "fields":[
        {"name":"packet_grab_nr", "type": "long"    },
        {"name":"srcIP",          "type": "string"  },
        {"name":"dstIP",          "type": "string"  },
        {"name":"srcPort",        "type": "int"     },
        {"name":"dstPort",        "type": "int"     },
        {"name":"seqNumber",      "type": "int"     },
        {"name":"ackNumber",      "type": "int"     },
        {"name":"dumpHost",       "type": "string"  },
        {"name":"protocolType",   "type": "string"  },
        {"name":"data",           "type": ["null", "bytes"] }
    ]
}
```

Output is written to folder dump using the following filename pattern:

packetdump_HOSTNAME_DEVICE_STARTTIME.avro

Example:

packetdump_quickstart.cloudera_lo_2016-10-28 09:00:04.avro

Packet Analysis in Hadoop

1.) Prepare DUMP-Space (only once)

```
hadoop fs -mkdir /user/cloudera/TCPDUMP/
hadoop fs -mkdir /user/cloudera/TCPDUMP/raw
hadoop fs -mkdir /user/cloudera/TCPDUMP/META
hadoop fs -put ./schema/* /user/cloudera/TCPDUMP/META
```

Create the Hive table:

```
CREATE EXTERNAL TABLE tcpdumps_avro
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION '/user/cloudera/TCPDUMP/raw'
TBLPROPERTIES ('avro.schema.url'='hdfs://127.0.0.1:8020/user/cloudera/TCPDUMP/META/packet.avsc')
```

2.) Upload new dumps to HDFS

Rename the file (all ":" have to be replaced by "_").

```
hadoop fs -put ./dump/*.avro /user/cloudera/TCPDUMP/raw
rm ./dump/*
```

3.) Count Packets in Hive and links between cluster hosts.

```
SELECT data FROM tcpdumps_avro;
SELECT srcip AS Source, dstip AS Target, count(dstip) AS Weight FROM tcpdumps_avro group by srcip, dstip;
```

Limitations

- currently no timestamp per packet available, should be also part of the record
- device should also be part of the record
- limiting the sniff-periode is based on nr of packages, should be a time interval in the future