IPFIX and DPI Information in a Big Data Environment

Katherine Prevost
Timothy Shimeall

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Homeland Security under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

DM22-1141

Outline

The Problem

- Classic Packet and Flow Data
- Enriching Flow Data

Deep Packet Inspection

- Analyzing DPI Data
- Design Points of YAF and Mothra

Example: DNS Abuse

- Typical DNS
- DNS protocol abuse

The Problem

How can we more precisely identify aspects of network behavior without giving up the size and coverage benefits of network flow?

Classic Packet and Flow Data

Full packet captures

Classic flow records

- Maximum fidelity
  - Everything observed is preserved
- Large size means historical record is limited
  - Transporting data is challenging
  - Storing data is challenging
- Messages can be examined in full detail

- Essential information only
  - Who spoke, when, and how
- Small records enables long retention period
  - Simplifies gathering data together
  - Reduces cost of preserving data
- Limited detail means painstaking analysis is required to determine meaning

Enriching Flow Data

Enhanced flow statistics

- How can we further characterize the flow as a whole?
- Packet size histograms, entropy measures, etc.

Application labeling

- What protocol is actually observed on this port?

Partial payload

- What does the start of the conversation look like?
- What do certain types of conversations look like?

Deep packet inspection

- What are salient details from additional protocol layers?
- (This is what we're talking about today.)

Deep Packet Inspection

Classic flow includes information from multiple network layers:

- Layer 2 (IP) address information and statistics
  - source and destination addresses, bytes, packets
- Layer 3 (TCP/UDP/ICMP/etc.) address and protocol information and statistics
  - source and destination ports, TCP flags, ICMP types and codes

We can enrich these records by including additional protocols:

- DNS query and response information
- Protocol versions and other unsecured details (HTTP, SMTP, SSH, etc.)
- Information about encryption and authentication keys in use (various TLS/SSL)

(Note: another space for choices)

Analyzing DPI Data

DPI information doesn't always line up one-to-one with flows

- Example: Multiple SMTP messages, HTTP requests, DNS resource records
- Sometimes even more: multiple file attachments per SMTP message

Richer data means you can ask more complicated questions

- More kinds of things to ask about
- More ways to ask about those things
- More ways to ask the wrong question for what you're trying to find out

A powerful language is needed to clearly express complicated queries

Design Points of YAF and Mothra

YAF

- Uses the IETF standard IPFIX binary protocol to encode flow data
  - We find that binary records typically take around 1/3 the space of text-based formats
- Produces large records containing structured DPI information
  - Is capable of producing partial packet capture data (we're not using this today)

Mothra

- A library and set of tools for use with Apache Spark
  - Tools for storing and partitioning flow data for later retrieval (we're not using this today)
- Reads binary IPFIX records directly, including structured DPI information
- Makes IPFIX fields including structured DPI information available to normal Apache Spark queries
- The full power of Apache Spark is available for analyzing data

Example Data

The CIC-Bell-DNS-EXF-2021 dataset is used for examples under the following license, available at https://www.unb.ca/cic/datasets/dns-exf-2021.html:

License

You may redistribute, republish, and mirror the CIC-Bell-DNS-EXF-2021 dataset in any form. However, any use or redistribution of the data must include a citation to the CIC-Bell-DNS-EXF-2021 dataset and the following paper:

Samaneh Mahdavifar, Amgad Hanafy Salem, Princy Victor, Miguel Garzon, Amir H. Razavi, Natasha Hellberg, Arash Habibi Lashkari, **"Lightweight Hybrid Detection of Data Exfiltration using DNS based on Machine Learning"**, The 11th IEEE International Conference on Communication and Network Security (ICCNS), Dec. 3-5, 2021, Beijing Jiaotong University, Weihai, China.

Typical (benign) DNS – SiLK & Mothra

```
$ rwfilter light_benign.rw --aport=53 --pass=stdout | \
  rwuniq --fields=sip --values=flows,bytes,packets --packets=10- \
         --bytes=200- --sort
          sIP|Flows|  Bytes|Packets|
     8.8.8.8|15531|2010590|  22102|
192.168.20.38|15531|1340081|  22105|

$ echo ":load dnsIDBenign.sc" | \
  spark-shell --packages org.cert.netsa:mothra_2.12:1.6.0
+-----------------------------------+---------+-----+----+----+------+-------+
|dnsName                            |dnsRRType|flows|#sIP|#dIP|bytes |packets|
+-----------------------------------+---------+-----+----+----+------+-------+
|[252.0.0.224.in-addr.arpa.]        |[12]     |2835 |1   |1   |416539|5901   |
|[150.20.168.192.in-addr.arpa.]     |[12]     |982  |1   |1   |242585|3125   |
|[200.20.168.192.in-addr.arpa.]     |[12]     |895  |1   |1   |134756|1836   |
|[15.20.168.192.in-addr.arpa.]      |[12]     |901  |1   |1   |133490|1844   |
|[100.20.168.192.in-addr.arpa.]     |[12]     |757  |1   |1   |112173|1533   |
|[2.20.168.192.in-addr.arpa.]       |[12]     |635  |1   |1   |91734 |1288   |
{ ... snip ... }
```

Abusive DNS (data exfiltration)

```
$ rwfilter light_compress.rw --aport=53 --pass=stdout | \
  rwuniq --fields=sip --values=flows,bytes,packets --packets=10- \
         --bytes=200- --sort
            sIP|   Records|               Bytes|        Packets|
       8.8.8.8|        20|                3653|             20|
  192.168.20.38|      9729|             2956321|          10085|
  192.168.20.72|      9709|             3122803|          10065|
$ echo ":load dnsIDAbuse.sc" | \
  spark-shell --packages org.cert.netsa:mothra_2.12:1.6.0
+-----------------------------------+---------+-----+----+----+------+-------+
|dnsName                            |dnsRRType|flows|#sIP|#dIP|bytes |packets|
+-----------------------------------+---------+-----+----+----+------+-------+
|[252.0.0.224.in-addr.arpa.]        |[12]     |1260 |1   |1   |191398|2696   |
|[2.20.168.192.in-addr.arpa.]       |[12]     |255  |1   |1   |130725|1615   |
|[150.20.168.192.in-addr.arpa.]     |[12]     |416  |1   |1   |63606 |866    |
|[200.20.168.192.in-addr.arpa.]     |[12]     |388  |1   |1   |57686 |788    |
|[15.20.168.192.in-addr.arpa.]      |[12]     |379  |1   |1   |56492 |781    |
|[100.20.168.192.in-addr.arpa.]     |[12]     |340  |1   |1   |50738 |694    |
|[3.20.168.192.in-addr.arpa.]       |[12]     |125  |1   |1   |17750 |250    |
|[250.255.255.239.in-addr.arpa.]    |[12]     |32   |1   |1   |4736  |64     |
{ ... snip ... }
```

Conclusion/Questions

There are a variety of approaches for enriching flow data

Using flow data with DPI information balances storage volume against precision

This will improve the degree of actionability for the results

This may also slow down production of some results as data volume increases

dnsIDExfil.sc

```
import org.cert.netsa.mothra.datasources._
import ipfix.IPFIXFields

val data_dir = ".../path/to/data"

// In dnsIDBenign.sc:
val data_file = s"$data_dir/light_benign.ipfix"

// In dnsIDAbuse.sc:
// val data_file =
//   s"$data_dir/light_compressed.ipfix"

val data = {
  spark.read.fields(
    IPFIXFields.default, IPFIXFields.dpi.dns
  ).ipfix(data_file)
}

val result = {
  data
    .filter(
      ($"silkAppLabel" === 53) &&
      (size($"dnsRecordList") > 0))
    .select(
      $"startTime",
      $"sourceIPAddress",
      $"destinationIPAddress",
      $"octetCount",
      $"packetCount",
      $"dnsRecordList.dnsName" as "dnsName",
      $"dnsRecordList.dnsRRType" as "dnsRRType",
      $"dnsRecordList.dnsQueryResponse" as "dnsQR",
      $"dnsRecordList.dnsResponseCode" as "dnsResponse")
    .groupBy(
      $"dnsName", $"dnsRRType")
    .agg(
      count($"*") as "flows",
      countDistinct($"sourceIPAddress") as "#sIP",
      countDistinct($"destinationIPAddress") as "#dIP",
      sum($"octetCount") as "bytes",
      sum($"packetCount") as "packets")
    .filter(
      ($"packets" > 20) &&
      ($"bytes" / $"packets" > 70) &&
      (size($"dnsName") < 3))
    .orderBy($"bytes".desc)
}

result.show(100, false)
```