HideZeroOne
INE – Cyber Sec
www.hide01.ir

Incident Handling & Response Professional

Intrusion Detection By Analyzing Flows

Section 02 | Module 03

OUTLINE

Table of Contents

Module 03 | Intrusion Detection By Analyzing Flows

3.1 Network Flows: Definition, Strengths & Limitations

3.2 Network Flow Analysis Toolkit

3.3 Practical Flow Analysis

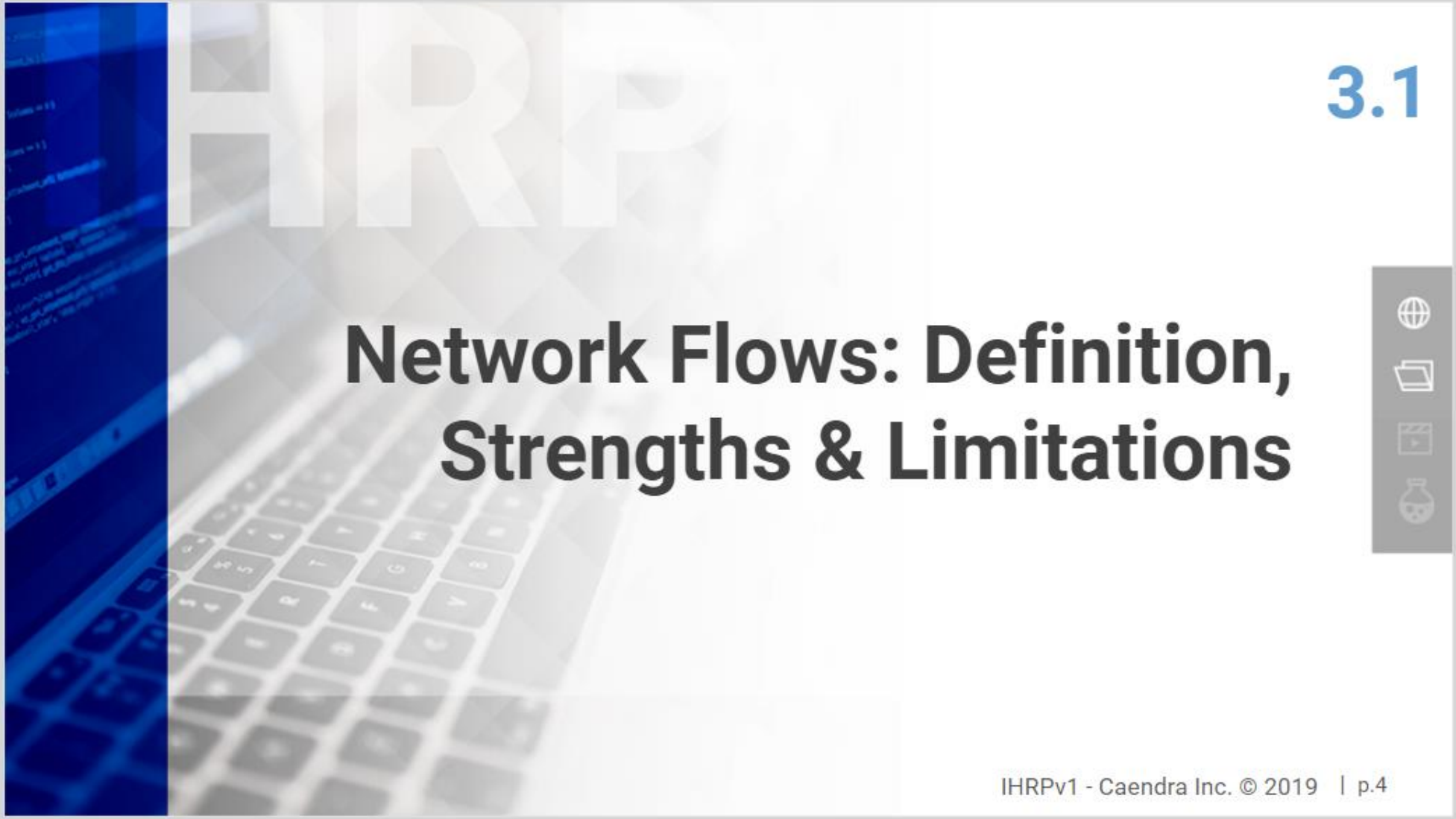
OUTLINE

Learning Objectives

By the end of this module, you should have a better understanding of:

- ✓ Network flow tracking, including its strengths, limitations and tools
- ✓ How to analyze network flows and leverage them for situational awareness
- ✓ How to enrich network flows with other data sources

OUTLINE



3.1

Network Flows: Definition, Strengths & Limitations

OUTLINE

3.1 Network Flows: Definition, Strengths & Limitations

After an incident has occurred, the faster we identify the attackers and the compromised assets, the quicker we will be able to proceed to the containment and eradication phases.

Collecting evidence inside a heterogeneous network and/or analyzing every captured packet (if there are any) is not only demanding, but time consuming as well.

3.1 Network Flows: Definition, Strengths & Limitations

Fortunately, most networking devices nowadays are able to track and export network flows. Well-orchestrated network flow tracking can provide incident responders with a bird's-eye view of all the communications that took (or currently take) place.

Such a view can help incident responders discover not only the attacking as well as the compromised machines but also any abnormal network behavior.

3.1.1 Network Flows: Definition & NetFlow Overview

But what are network flows exactly?

A network flow is essentially a record of a communication between two hosts. A network flow usually contains the source IP address, the destination IP address and the TCP/UDP ports used during a “conversation”. Other information, such as the router or switch interface where the flow was spotted, the number of bytes transferred etc. can also be contained in a network flow, as you will see in just a bit.

3.1.1 Network Flows: Definition & NetFlow Overview

When it comes to network flow tracking, Cisco's NetFlow technology is the most commonly used one, but other vendors have also approached the network flow tracking topic by adopting jFlow, cFlow, sFlow and IPFIX, which are similar in terms of the provided functionality.

NetFlow reports on traffic statistics, such as sender, receiver, packets, and bytes. It doesn't report the actual protocol payload though.

On your right, you can see the different NetFlow versions.

Version	Comment
v1	First implementation, now obsolete, and restricted to IPv4 (without IP mask and AS Numbers).
v2	Cisco internal version, never released.
v3	Cisco internal version, never released.
v4	Cisco internal version, never released.
v5	Most common version, available (as of 2009) on many routers from different brands, but restricted to IPv4 flows.
v6	No longer supported by Cisco. Encapsulation information (?).
v7	Like version 5 with a source router field. Used (only?) on Cisco Catalyst switches.
v8	Several aggregation form, but only for information that is already present in version 5 records
v9	Template Based, available (as of 2009) on some recent routers. Mostly used to report flows like IPv6, MPLS, or even plain IPv4 with BGP nexthop.
v10	aka IPFIX, IETF Standardized NetFlow 9 with several extensions like Enterprise-defined fields types, and variable length fields.

http://www.ciscopress.com/articles/article.asp?p=2812391&seqNum=3

https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

OUTLINE

3.1.1 Network Flows: Definition & NetFlow Overview

Most firewalls, routers and switches can export NetFlow logs to a collecting server.

As already discussed, unlike packet capture, with NetFlow you can't ensure that, for example, the traffic spotted over port 80 is actually HTTP traffic, since it has no visibility into the actual protocol payload.

3.1.1 Network Flows: Definition & NetFlow Overview

Some attributes NetFlow can export are:

- IP source address
- IP destination address
- Source port
- Destination port
- Layer 3 protocol type
- Class of service
- Router or switch interface

3.1.1 Network Flows: Definition & NetFlow Overview

NetFlow V5 is the most commonly used version and is supported by numerous different router manufacturers.

The number of fields NetFlow V5 can export are limited though.

3.1.1 Network Flows: Definition & NetFlow Overview

NetFlow V9, on the other hand, is a template-based flow with no restriction on the number of fields to be exported.

Unlike V5 where the headers are hardcoded, V9 offers a dynamic header. In order for the collector to decode the datagrams, it must receive a template from the NetFlow exporter which defines the headers. The collector cannot decode the datagrams until the corresponding template is received.

3.1.1 Network Flows: Definition & NetFlow Overview

IPFIX is an open-source implementation based on NetFlow Version 9 and was created to meet the need for a universal standard to export IP flow information from network devices.

It is on the IETF standards, and it is implemented by multiple vendors.

Specifications of IPFIX are documented in RFC 7011, RFC 7015, and RFC 5103.

3.1.1 Network Flows: Definition & NetFlow Overview

A NetFlow setup for monitoring usually consists of 3 components:

- **Exporter**: Router, Switch, Firewall, … etc.
- **Collector**: Software for receiving and storing NetFlow
- **Analysis**: Software for analyzing the flow

3.1.1 Network Flows: Definition & NetFlow Overview

NetFlow exports data in UDP datagrams.

The NetFlow RFC 3954 does not specify a NetFlow listening port; however, Wireshark and most other solutions assume NetFlow on port 2055 and IPFIX on port 4739.

3.1.1 Network Flows: Definition & NetFlow Overview

IPv4: What NetFlow is mostly interested in

Bit offset	0-3	4-7	8-13	14-15	16-18	19-31
0	Version	Internet Header Length	Differentiated Services Code Point	Explicit Congestion Notification	Total Length	
32	Identification				Flags	Fragment Offset
64	Time to Live		Protocol		Header Checksum	
96	Source IP Address					
128	Destination IP address					
160	Options					
160 or 192+	Data					

http://www.ciscopress.com/articles/article.asp?p=2812391&seqNum=3

OUTLINE

3.1.1 Network Flows: Definition & NetFlow Overview

TCP: What NetFlow is mostly interested in

Octet	0								1								2								3							
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port																Destination port															
32	Sequence number																															
64	Acknowledgment number (if ACK is set)																															
96	Data offset				Reserved				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window size														
128	Checksum																Urgent pointer (if URG is set)															
160 ...	Options																															

http://www.ciscopress.com/articles/article.asp?p=2812391&seqNum=3

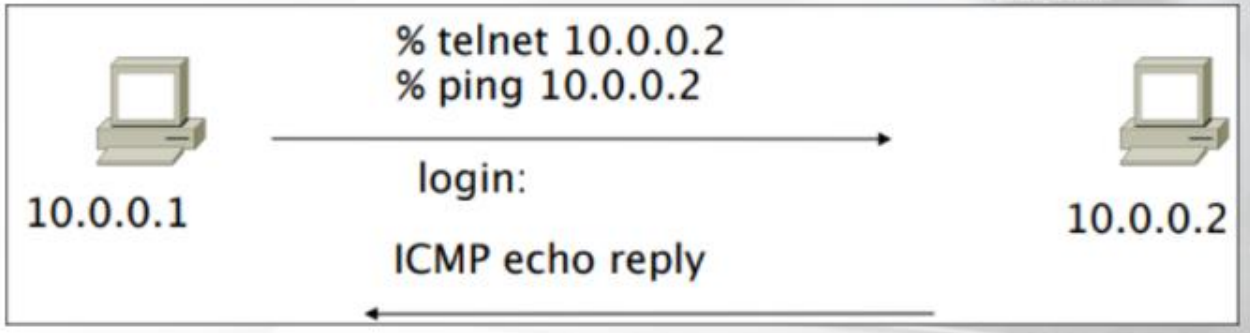
3.1.1 Network Flows: Definition & NetFlow Overview

UDP: What NetFlow is mostly interested in

Bit Offset	0-15	16-31
0	Source port	Destination port
32	Length	Checksum
64 ...	Data	

3.1.1 Network Flows: Definition & NetFlow Overview

Network flows can be unidirectional or bidirectional, in terms of representation. Suppose that the below communication occurred. Let's see this "conversation" through the lens of network flows.



3.1.1 Network Flows: Definition & NetFlow Overview

Unidirectional Flow

Active Flows

Flow	Source IP	Destination IP	Prot.	srcPort	dstPort
1	10.0.0.1	10.0.0.2	TCP	32000	23
2	10.0.0.2	10.0.0.1	TCP	23	32000
3	10.0.0.1	10.0.0.2	ICMP	0	0
4	10.0.0.2	10.0.0.1	ICMP	0	0

3.1.1 Network Flows: Definition & NetFlow Overview

Bidirectional Flow

Active Flows

Flow	Source IP	Destination IP	Prot.	srcPort	dstPort
1	10.0.0.1	10.0.0.2	TCP	32000	23
2	10.0.0.1	10.0.0.2	ICMP	0	0

3.1.2 Network Flows: Strengths & Limitations

Let us now summarize the strengths and limitations of network flow tracking, before we start covering the network flow analysis toolkit.

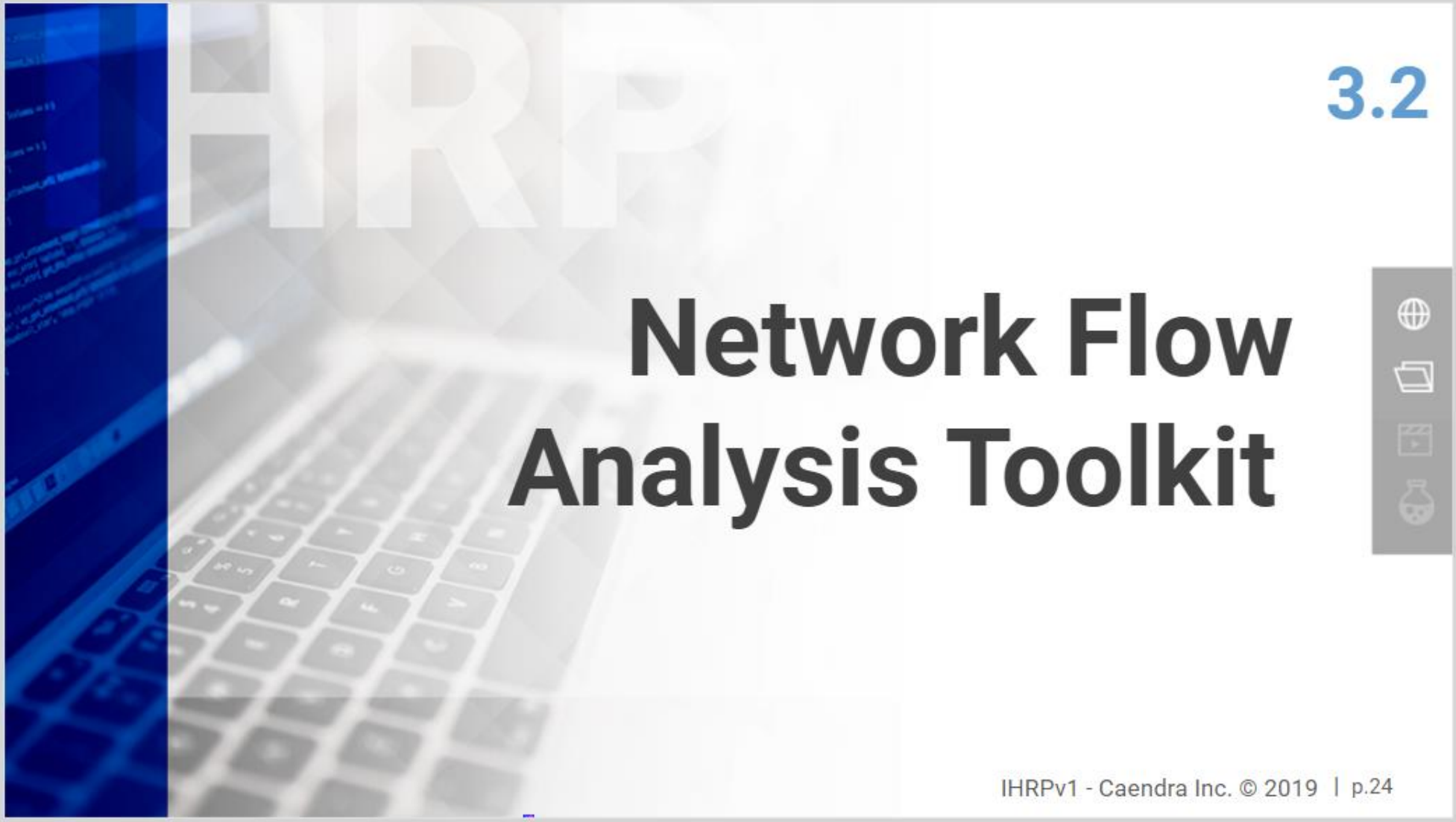
3.1.2 Network Flows: Strengths & Limitations

Network Flow Tracking Strengths

1. Capturing communication information in a storage-effective and time-effective manner
2. Still useful even in encrypted communications (when, where, how much and at what time information are still available)
3. Useful to baseline an entire environment
4. Flows can be obtained from multiple places within the network

Network Flow Tracking Limitations

1. Not a substitute of full packet capture, since it does **not** contain the content of each message that was sent
2. Analyzing network flows doesn't always pay dividends
3. Selecting/deploying the appropriate flow collector needs careful planning



3.2

Network Flow Analysis Toolkit

OUTLINE

3.2 Network Flow Analysis Toolkit

When it comes to network flow analysis, three (3) tools deserve our attention.

1. YAF
2. SiLK
3. FlowViewer

https://tools.netsa.cert.org/yaf/
https://tools.netsa.cert.org/silk/
https://ensight.eos.nasa.gov/FlowViewer/

3.2.1 Network Flow Analysis Toolkit: YAF

The first tool we will cover from the network flow analysis arsenal is YAF.

YAF is Yet Another Flowmeter. According to its creators, *YAF processes packet data deriving from a PCAP traffic capture file or a live capture (from an interface using PCAP) into bidirectional flows and then exports those flows to IPFIX Collecting Processes or in an IPFIX-based file format. YAF's output can be used with the SiLK flow analysis tools, super_mediator, Pipeline 5, and any other IPFIX compliant toolchain.*

https://tools.netsa.cert.org/yaf/
https://tools.netsa.cert.org/silk/

3.2.1 Network Flow Analysis Toolkit: YAF

YAF's optional features include:

- Application labeling
  - Rules located in */usr/local/etc/yafApplabelRules.conf*
- OS detection (supports passive OS fingerprinting via libP0f and DHCP)
  - DHCP fingerprints located in */usr/local/etc/dhcp_fingerprints.conf*
  - Output viewed with *yaf-file-mediator*
- Deep packet inspection
  - Enabled by specifying, *--plugin-name=/usr/local/lib/yaf/dpacketplugin.la*
  - You can also specify a protocol to perform DPI. *--plugin-opts="53 80 21"*
  - DPI rule configuration is located in */usr/local/etc/yafDPIRules.conf*
  - Output viewed with *yaf-file-mediator*

https://tools.netsa.cert.org/yaf/applabel.html
http://tools.netsa.cert.org/yaf/yafdhcp.html
https://tools.netsa.cert.org/confluence/display/tt/YAF+2.x+IPFIX+File+Mediator
https://tools.netsa.cert.org/yaf/yafdpi.html

3.2.1 Network Flow Analysis Toolkit: YAF

PCAP -> IPFIX with YAF

```
>> yaf --in filename.pcap --out filename.yaf
```

- If you want application labeling add the below
  - *--applabel-rules= /usr/local/etc/yafApplabelRules.conf --max-payload 300*
- If you want OS detection (via P0f) add the below
  - *--p0fprint --p0f-fingerprints /usr/local/etc/ --max-payload 300*
- If you want OS detection (via DHCP fingerprints) add the below
  - *--plugin-name=/usr/local/lib/yaf/dhcp_fp_plugin.la*
- To perform DPI on top on the conversion add the below
  - *--plugin-name=/usr/local/lib/yaf/dpacketplugin.la*

3.2.2 Network Flow Analysis Toolkit: SiLK

The definitive tool when it comes to network flow analysis is SiLK. According to its creators, *the SiLK tool suite supports the efficient collection, storage, and analysis of network flow data, enabling network security analysts to rapidly query large historical traffic data sets.*

3.2.2 Network Flow Analysis Toolkit: SiLK

Just like Linux commands, SiLK commands are piped to form a workflow. For effectively using SiLK, one should become familiar with crawling with *rwcut* and walking through flow files with *rwcut* and *rwfilter*.

https://tools.netsa.cert.org/silk/rwcut.html
https://tools.netsa.cert.org/silk/rwfilter.html

3.2.2 Network Flow Analysis Toolkit: SiLK

Let's now cover some important SiLK commands starting with *rwfileinfo*, *that* provides metadata information about a SiLK file.

```
>> rwfileinfo filename.rwf
```

```
student@ubuntu:~/Data/isilk$ rwfileinfo ITOC-Border.rwf
ITOC-Border.rwf:
  format(id)          FT_RWGENERIC(0x16)
  version             16
  byte-order          littleEndian
  compression(id)     none(0)
  header-length       104
  record-length       52
  record-version      5
  silk-version        2.3.1
  count-records       2907505
  file-size           151190364
  command-lines
                    1 rwipfix2silk --silk-output=ITOC-Border
```

https://tools.netsa.cert.org/silk/rwfileinfo.html

3.2.2 Network Flow Analysis Toolkit: SiLK

You can view flow records in clear text with *rwcut*, as follows.

```
>> rwcut --num-rec=10 filename.rwf
>> rwcut --num-rec=10 --
fields=sip,dip,proto,sport,dport,stime filename.rwf
```

```
student@ubuntu:~/Data/isilk$ rwcut -num-rec=10 ITOC-Border.rwf
        sIP|                dIP|sPort|dPort|pro|   packets|     bytes|   flags|               sTime|      dur|               eTime|sen|
 10.1.60.203|       10.1.60.187|50398|   80|  6|         5|       383|FS PA   |2009/04/20T11:35:19.439|    0.006|2009/04/20T11:35:19.445|  0|
 10.1.60.187|       10.1.60.203|   80|50398|  6|         5|       674|FS PA   |2009/04/20T11:35:19.440|    0.005|2009/04/20T11:35:19.445|  0|
 10.1.60.203|       10.1.60.73|50189| 5222|  6|         6|       433|FS PA   |2009/04/20T11:35:19.446|    0.009|2009/04/20T11:35:19.455|  0|
```

```
student@ubuntu:~/Data/isilk$ rwcut --num-rec=10 --fields=sip,dip,proto,sport,dport,stime ITOC-Border.rwf
                sIP|                dIP|pro|sPort|dPort|               sTime|
        10.1.60.203|       10.1.60.187|  6|50398|   80|2009/04/20T11:35:19.439|
        10.1.60.187|       10.1.60.203|  6|   80|50398|2009/04/20T11:35:19.440|
```

3.2.2 Network Flow Analysis Toolkit: SiLK

You can count how much traffic matched specific keys as well as what protocols were running, as follows. (The two commands are identical)

```
        >> rwtotal --proto --skip-zero filename.rwf
>> rwuniq --field=proto --values=records,bytes,packets --
                  sort-output filename.rwf
```

```
student@ubuntu:~/Data/isilk$ rwtotal --proto --skip-zero ITOC-Border.rwf
   protocol|        Records|            Bytes|          Packets|
           1|           5421|         54041934|           605793|
           2|              1|               20|                1|
           3|              1|               20|                1|
           4|              1|               20|                1|
           5|              1|               20|                1|
           6|        2849321|        505291682|          4447354|
```

3.2.2 Network Flow Analysis Toolkit: SiLK

If you want to identify the most common servers and source ports with at least 50 flows, you can do so as follows.

```
>> rwfilter --proto=6,17 --pass=stdout filename.rwf |
rwuniq --field=sip,sport --flows=50 --bytes --values=dip-
        distinct| sort -nr -k 3,3 -t '|' | head
```

```
student@ubuntu:~/Data/isilk$ rwfilter --proto=6,17 --pass=stdout ITOC-Border.rwf
| rwuniq --field=sip,sport --flows=50 --bytes --values=dip-distinct| sort -nr -k
3,3 -t '|' | head
                    10.1.60.73| 5222|      8752|          21306614|
 103265|
                   10.1.60.187|   80|      6195|          22639527|
  47554|
                   10.1.60.187|  443|      4781|         214836019|
  48621|
```

3.2.2 Network Flow Analysis Toolkit: SiLK

If you want to see how the distribution of bytes, packets and bytes/packet looks like, you can do so with *rwstats* as follows.

```
>> rwstats --overall-stats filename.rwf
```

```
student@ubuntu:~/Data/isilk$ rwstats --overall-stats ITOC-Border.rwf
FLOW STATISTICS--ALL PROTOCOLS:  2907505 records
*BYTES min 20; max 9201413
  quartiles LQ 16.63914 Med 33.27829 UQ 50.77612 UQ-LQ 34.13698
   interval_max|count<=max|%_of_input|   cumul_%|
             40|   1747388| 60.099226| 60.099226|
             60|    804074| 27.655120| 87.754346|
            100|     48691|  1.674666| 89.429012|
            150|     31585|  1.086327| 90.515339|
```

https://tools.netsa.cert.org/silk/rwstats.html

3.2.2 Network Flow Analysis Toolkit: SiLK

If you want to identify the top 10 destination ports, you can do so with *rwstats* as follows.

```
>> rwstats --fields=dport --count=10 filename.rwf
```

```
student@ubuntu:~/Data/isilk$ rwstats --fields=dport --count=10 ITOC-Border.rwf
INPUT: 2907505 Records for 65536 Bins and 2907505 Total Records
OUTPUT: Top 10 Bins by Records
dPort|   Records|  %Records|   cumul_%|
   80|    506112| 17.407090| 17.407090|
55829|    390866| 13.443347| 30.850437|
  443|    367123| 12.626737| 43.477174|
 5222|    104799|  3.604431| 47.081604|
```

3.2.2 Network Flow Analysis Toolkit: SiLK

rwcount is useful for examining traffic across time. Consider the *--load-scheme* switch during your investigations.

```
>> rwcount --bin-size=300 filename.rwf | more
```

Date|	Records|	Bytes|	Packets
2009/04/20T11:35:00|	110.68|	73053.09|	676.57
2009/04/20T11:40:00|	157.49|	97584.80|	891.19
2009/04/20T11:45:00|	143.38|	89717.25|	843.08
2009/04/20T11:50:00|	158.04|	92293.02|	880.72

https://tools.netsa.cert.org/silk/rwcount.html

3.2.2 Network Flow Analysis Toolkit: SiLK

rwfilter is SiLK's go to command for filtering, since there are switches for every flow attribute. For example, to identify top webservers you can execute the below.

```
>> rwfilter filename.rwf --sport=80,443,8080 --protocol=6 -
    -packets=4- --ack-flag=1 --pass=stdout | rwstats --
            fields=sip --percentage=1 --bytes
```

```
student@ubuntu:~/Data/isilk$ rwfilter ITOC-Border.rwf --sport=80,443,8080 --protocol=6 --packets=4- --ack-flag=1 --pass=stdout | rwstats --fields=sip --percentage=1 --bytes
INPUT: 71226 Records for 15 Bins and 250140493 Total Bytes
OUTPUT: Top 4 bins by Bytes (1% == 2501404)
                 sIP|      Bytes|    %Bytes|   cumul_%|
       10.1.60.187|  234969338| 93.934946| 93.934946|
         10.1.90.5|    2686023|  1.073806| 95.008752|
         10.1.20.4|    2581250|  1.031920| 96.040672|
        10.2.20.60|    2570521|  1.027631| 97.068303|
```

3.2.2 Network Flow Analysis Toolkit: SiLK

Scanning activity can be detected within a SiLK dataset, through the *rwscan* command.

```
>> rwsort --fields=sip,proto,dip filename.rwf |rwscan --
                    scan-model=2 | more
```

```
student@ubuntu:~/Data/isilk$ rwsort --fields=sip,proto,dip ITOC-.rwf|rwscan --sca
ITOC-Border.rwf   ITOC-Netflow.ZIP  ITOC-NSA.rwf
student@ubuntu:~/Data/isilk$ rwsort --fields=sip,proto,dip ITOC-NSA.rwf|rwscan --
             sip| proto|                  stime|                  etime|     fl
     10.1.10.10|     6|     2009-04-21 08:45:05|     2009-04-24 14:24:50|
```

https://tools.netsa.cert.org/silk/rwscan.html

3.2.2 Network Flow Analysis Toolkit: SiLK

rwfilter can also detect scanning attempts. For example, one could specify the following criteria: i) size less than 2048 bytes, ii) 1 to 3 packets and iii) no RST or FIN flags.

```
>> rwfilter filename.rwf --bytes=0-2048 --packets=1-3 --
   flags-all=/RF --pass=stdout|rwuniq --fields=sip --
values=dip-distinct,records| sort -k 3,3 -n -r -t '|'| head
                              -n 30
```

```
student@ubuntu:~/Data/isilk$ rwfilter ITOC-NSA.rwf --bytes=0-2048 --packets=1-3 -
-flags-all=/RF --pass=stdout|rwuniq --fields=sip --values=dip-distinct,records| s
ort -k 3,3 -n -r -t '|'| head -n 30
                          10.1.10.5|        50|     11065|
                          10.1.30.5|        66|      9729|
```

3.2.2 Network Flow Analysis Toolkit: SiLK

SiLK has a powerful feature called *IP Sets*. An IP set is a binary representation of an arbitrary collection of IP addresses. IP sets showcase their true potential when used in conjunction with *rwfilter*.

OUTLINE

3.2.2 Network Flow Analysis Toolkit: SiLK

That being said, one may want to associate a value with each address in an IP set. For example, one may want to associate the IP addresses that engage in web traffic with the volume of flows, packets or bytes of web traffic that each address carries. This can be done through another powerful SiLK feature *IP Bags*.

Bags are essentially extended sets.

3.2.2 Network Flow Analysis Toolkit: SiLK

Let's take for example the command flow below that can detect a scanning attempt.

```
>> rwsort --fields=sip,proto,dip filename.rwf |rwscan --
                    scan-model=2 | more
```

One can turn the result of the above into an *IP Bag*, as follows.

```
  >>rwsort --fields=sip,proto,dip filename.rwf |rwscan --
scan-model=2 --no-titles | cut -d '|' -f 1,5 | rwbagbuild -
           -bag-input=stdin > rwscan-output.bag
```

3.2.2 Network Flow Analysis Toolkit: SiLK

The output can be viewed, as follows.

```
>> rwbagcat rwscan-output.bag | sort -t '|' -k 2,2 -rn |
                                head
```

```
student@ubuntu:~/Data/isilk$ rwbagcat rwscan-output.bag | sort -t '|' -k 2,2 -rn
| head
      10.1.60.3|                399469|
   10.2.190.249|                262774|
```

3.2.2 Network Flow Analysis Toolkit: SiLK

We barely scratched the surface of SiLK's capabilities. SiLK is an extremely capable and customizable tool. You can learn much more about it by referring to the below resources.

- https://schd.ws/hosted_files/flocon2017/fa/flocon-2016-silk-tutorial.pptx
- https://tools.netsa.cert.org/silk/analysis-handbook.pdf
- RVAsec: Jason Smith - Applied Detection and Analysis Using Flow Data

3.2.3 Network Flow Analysis Toolkit: FlowViewer

FlowViewer is a NetFlow analyzer. It is actually a front-end for flow-tools. FlowViewer is great for reporting on historical data.

https://ensight.eos.nasa.gov/FlowViewer/
https://github.com/adsr/flow-tools

3.2.3 Network Flow Analysis Toolkit: FlowViewer

FlowViewer reports are quite clear and straightforward. See an example below.

In this case, we have applied a filter for TCP flags = 27.
NetFlow records contain a field reporting the cumulative OR-ed TCP flags seen on the flow. If we "sum" (using OR) all the flags involved in a TCP connection (SYN [2] + ACK [16] + PSH [8] + FIN [1]) we have 27.

```
      Report: 132 Columns                                  Sort Field: n/a
  Start Time: March 8, 2010 9:00:00 CET                      End Time: March 8, 2010 16:20:00 CET
      Device: [redacted]                                     Exporter:
      Source: 192.168.0.154                             Destination: 192.168.1.184
 Source Port:                                      Destination Port:
  Source I/F:                                       Destination I/F:
   Source AS:                                        Destination AS:
   TOS Field:                                             TCP Flags: 27
  Include if: Any part of flow in Time Period             Protocols:
Lines Cutoff: 100                                      Octets Cutoff:

Start            End              Sif   SrcIPaddress    SrcP  DIf   DstIPaddress    DstP    P Fl Pkts       Octets

0308.10:34:43.799 0308.10:34:44.995 66    192.168.0.154   1888  59    192.168.1.184   1433  6   3 15          2032
0308.11:00:31.339 0308.11:00:32.419 66    192.168.0.154   2472  59    192.168.1.184   1433  6   3 15          2034
0308.11:03:15.899 0308.11:03:16.715 66    192.168.0.154   2533  59    192.168.1.184   1433  6   3 15          2034
0308.11:18:00.811 0308.11:18:01.719 66    192.168.0.154   2877  59    192.168.1.184   1433  6   3 15          2042
0308.11:22:52.407 0308.11:22:53.483 66    192.168.0.154   3004  59    192.168.1.184   1433  6   3 15          2034
0308.11:26:47.751 0308.11:27:28.367 66    192.168.0.154   3107  59    192.168.1.184   1433  6   3 1420        1963241
0308.11:36:46.107 0308.11:36:47.195 66    192.168.0.154   3287  59    192.168.1.184   1433  6   3 15          2034
0308.11:41:42.115 0308.11:41:43.815 66    192.168.0.154   3297  59    192.168.1.184   1433  6   3 15          2034
```

3.3

Practical Flow Analysis

OUTLINE

3.3 Practical Flow Analysis

Let's now leverage network flows to detect some real-world attacks. Whenever we are provided with additional data, we will use them to enrich the network flows.

3.3.1 Practical Flow Analysis: Case 1

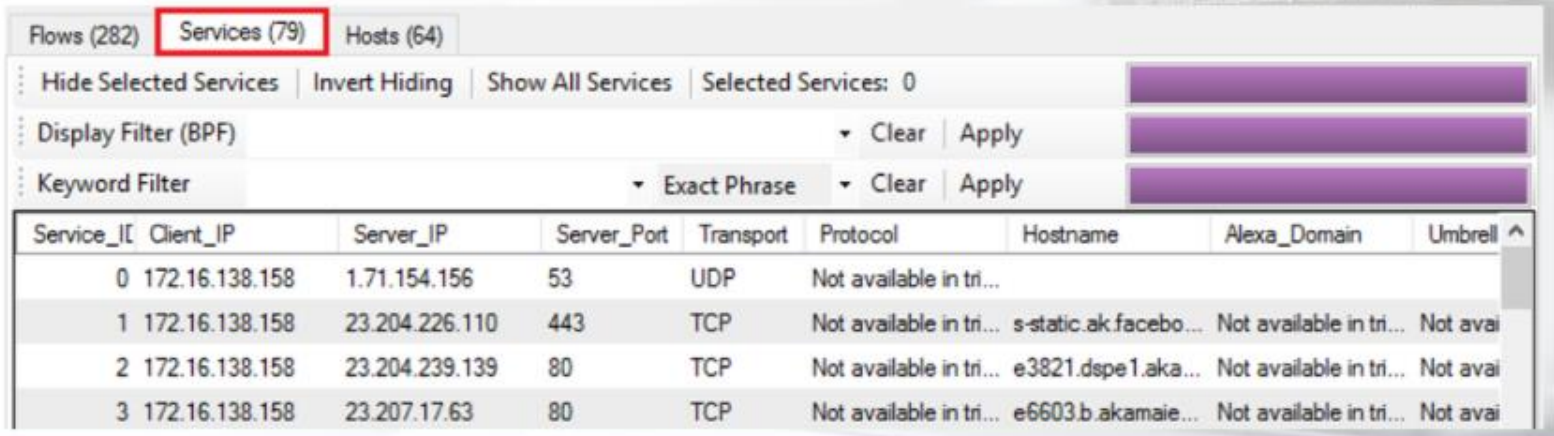
Case 1: Consider the provided *2015-03-09-traffic-analysis-exercise.pcap* file. Try to identify if there is anything suspicious going on by analyzing network flows **only**.

Hint: Load the provided PCAP into CapLoader.

3.3.1 Practical Flow Analysis: Case 1

Detection

CapLoader provides incident responders with the capability to quickly aggregate and view all the network flows found in a PCAP file. Specifically, each row inside the *Services* tab represents a unique combination of *Client-IP*, *Server-IP*, *Server-port* and *Transport-protocol*. Note that one row can be multiple flows merged together.

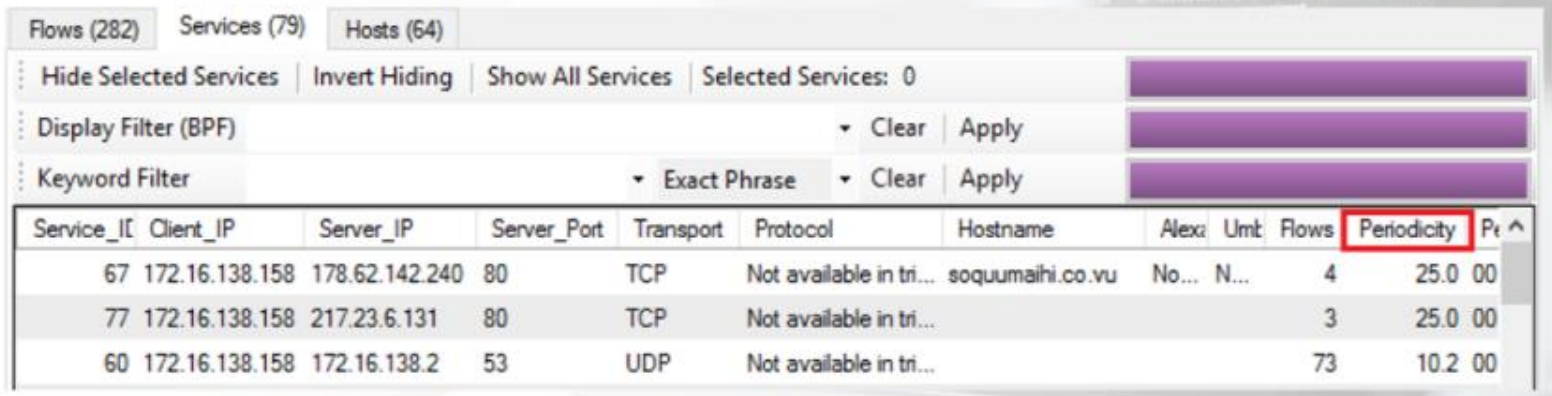


OUTLINE

3.3.1 Practical Flow Analysis: Case 1

Detection

CapLoader also features periodic flow detection. This means that one can detect malicious traffic based not on blacklists or IDS signatures but based on the periodicity of flows. You can see the above in action, by sorting the rows inside the *Services* tab based on the *Periodicity* column.

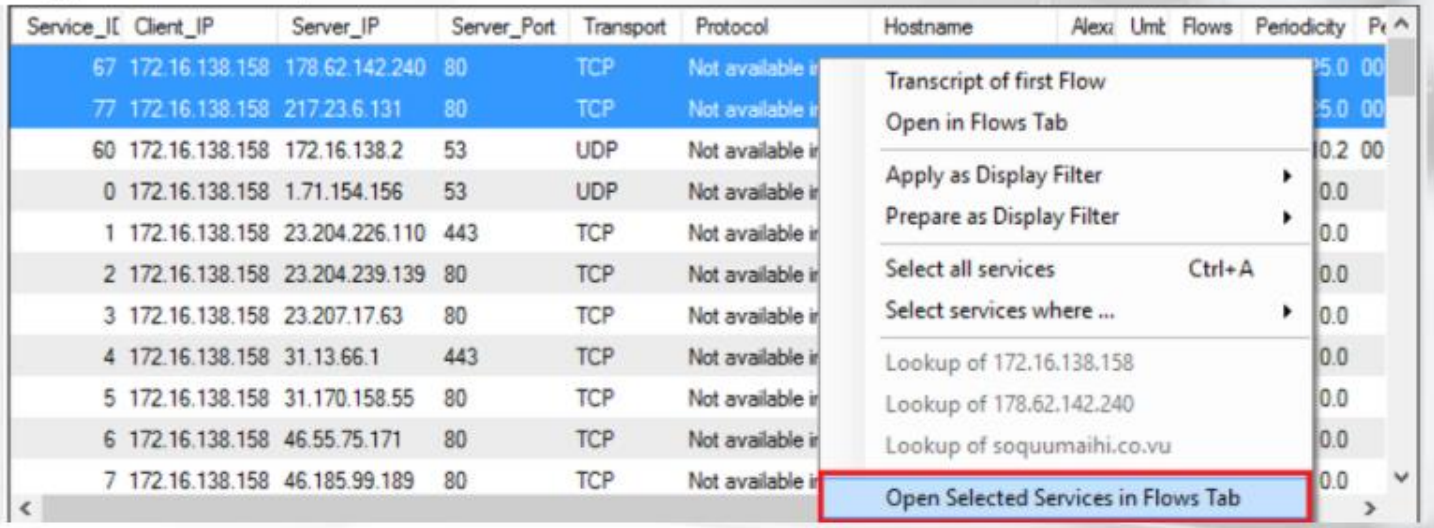


OUTLINE

3.3.1 Practical Flow Analysis: Case 1

Detection

Any Periodicity value greater than 20 can be considered periodic. To analyze the two most periodic entries, you can select both of them, then right click on them and finally, choose *Open Selected Services in Flows Tab*.



OUTLINE

3.3.1 Practical Flow Analysis: Case 1

Detection

By looking at the flows and their duration, it looks like we are dealing with a beaconing malware. The key phrase here is “looks like”. Network flows cannot provide us with solid evidence regarding an incident. They can speed up our analysis though and also serve as a great indicator towards the right direction.

Flows (282) | Services (79) | Hosts (64)

Hide Selected Flows | Invert Hiding | Show All Flows | Selected Flows: 0

Display Filter (BPF) | Clear | Apply

Keyword Filter | Exact Phrase | Clear | Apply

Flow_ID	Client_IP	Client_	Server_IP	Server_Port	Trar	Hostname		Start	End	Duration
45	172.16.138.158	49190	178.62.142.240	80	TCP	soquumaihi.co.vu	N	2015-03-09 21:54:23	2015-03-09 21:54:37	00:00:13.8634690
64	172.16.138.158	49209	178.62.142.240	80	TCP	soquumaihi.co.vu	N	2015-03-09 21:54:25	2015-03-09 21:54:37	00:00:11.5873970
68	172.16.138.158	49214	178.62.142.240	80	TCP	soquumaihi.co.vu	N	2015-03-09 21:54:37	2015-03-09 21:54:49	00:00:11.6434690
70	172.16.138.158	49219	178.62.142.240	80	TCP	soquumaihi.co.vu	N	2015-03-09 21:54:49	2015-03-09 21:55:02	00:00:13.8146540
75	172.16.138.158	49231	217.23.6.131	80	TCP			2015-03-09 21:55:16	2015-03-09 21:55:16	00:00:00.2791870
79	172.16.138.158	49244	217.23.6.131	80	TCP			2015-03-09 21:56:01	2015-03-09 21:56:02	00:00:00.2661330
83	172.16.138.158	49248	217.23.6.131	80	TCP			2015-03-09 21:56:54	2015-03-09 21:56:55	00:00:00.2788980

OUTLINE

3.3.1 Practical Flow Analysis: Case 1

Detection

If you are curious, the provided PCAP contains traffic from a host infected by the Nuclear exploit kit, which is indeed beaconing.

PCAP taken from: http://www.malware-traffic-analysis.net/2015/03/09/

3.3.2 Practical Flow Analysis: Case 2

Case 2: The organization you work for has tasked you with analyzing some collected NetFlows, to identify the interactions of system *192.168.5.100*, which was compromised on August 16th 2016.

Take some time to study *nfdump*'s man page: http://manpages.ubuntu.com/manpages/cosmic/en/man1/nfdump.1.html

3.3.2 Practical Flow Analysis: Case 2

Detection

Let's start by using *nfdump* to see a protocol overview.

```
>> nfdump -o long -R . -A proto 'ip 192.168.5.100'
```

```
Date first seen          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Flags Tos  Packets    Bytes
Flows
2016-06-27 21:58:49.817 4305964.221 IGMP          0.0.0.0:0     ->         0.0.0.0:0     ......   0      192     892
0     3
2016-08-16 13:12:23.134  9853.592 ICMP          0.0.0.0:0     ->       0.0.0.0:0.0     ......   0       19     960
    9
2016-06-27 19:50:20.901 4334018.631 UDP           0.0.0.0:0     ->         0.0.0.0:0     ......   0    11600    2.1
M  8753
2016-06-27 19:53:17.801 4333841.855 TCP           0.0.0.0:0     ->         0.0.0.0:0     ......   0    2.1 M    1.9
G 16692
```


OUTLINE

3.3.2 Practical Flow Analysis: Case 2

Detection

Let's start by using *nfdump* to see a protocol overview.

```
>> nfdump -o long -R . -A proto 'ip 192.168.5.100'
```

```
Date first seen          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Flags Tos  Packets    Bytes
Flows
2016-06-27 21:58:49.817 4305964.221 IGMP          0.0.0.0:0     ->          0.0.0.0:0     ......   0      192     892
0     3
2016-08-16 13:12:23.134  9853.592 ICMP          0.0.0.0:0     ->        0.0.0.0:0.0     ......   0       19     960
    9
2016-06-27 19:50:20.901 4334018.631 UDP          0.0.0.0:0     ->          0.0.0.0:0     ......   0    11600    2.1
M  8753
2016-06-27 19:53:17.801 4333841.855 TCP          0.0.0.0:0     ->          0.0.0.0:0     ......   0    2.1 M    1.9
G 16692
```

Nothing curious-looking so far.

OUTLINE

3.3.2 Practical Flow Analysis: Case 2

Detection

Let's continue by checking ICMP traffic.

```
>> nfdump -o long -R . 'ip 192.168.5.100 and proto icmp'
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 13:12:23.134	4189.521	ICMP	192.168.5.100:0	->	13.80.12.54:8.0	.....	0	2	120	1
2016-08-16 14:16:47.410	4189.446	ICMP	192.168.5.100:0	->	13.80.12.54:8.0	.....	0	2	120	1
2016-08-16 14:46:38.142	4192.291	ICMP	192.168.5.100:0	->	192.168.56.1:8.0	.....	0	2	92	1
2016-08-16 14:46:38.142	4192.292	ICMP	192.168.5.100:0	->	192.168.56.1:13.0	.....	0	2	92	1
2016-08-16 14:46:41.285	4192.279	ICMP	192.168.5.100:0	->	192.168.56.10:8.0	.....	0	2	92	1
2016-08-16 14:46:41.282	4192.283	ICMP	192.168.5.100:0	->	192.168.56.10:13.0	.....	0	2	92	1
2016-08-16 14:46:42.422	4188.012	ICMP	188.1.232.65:0	->	192.168.5.100:3.1	.....	0	3	168	1
2016-08-16 14:46:44.423	4192.302	ICMP	192.168.5.100:0	->	192.168.56.15:8.0	.....	0	2	92	1
2016-08-16 14:46:44.423	4192.303	ICMP	192.168.5.100:0	->	192.168.56.15:13.0	.....	0	2	92	1

3.3.2 Practical Flow Analysis: Case 2

Detection

Let's continue by checking ICMP traffic.

```
>> nfdump -o long -R . 'ip 192.168.5.100 and proto icmp'
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 13:12:23.134	4189.521	ICMP	192.168.5.100:0	->	13.80.12.54:8.0	.....	0	2	120	1
2016-08-16 14:16:47.410	4189.446	ICMP	192.168.5.100:0	->	13.80.12.54:8.0	.....	0	2	120	1
2016-08-16 14:46:38.142	4192.291	ICMP	192.168.5.100:0	->	192.168.56.1:8.0	.....	0	2	92	1
2016-08-16 14:46:38.142	4192.292	ICMP	192.168.5.100:0	->	192.168.56.1:13.0	.....	0	2	92	1
2016-08-16 14:46:41.285	4192.279	ICMP	192.168.5.100:0	->	192.168.56.10:8.0	.....	0	2	92	1
2016-08-16 14:46:41.282	4192.283	ICMP	192.168.5.100:0	->	192.168.56.10:13.0	.....	0	2	92	1
2016-08-16 14:46:42.422	4188.012	ICMP	188.1.232.65:0	->	192.168.5.100:3.1	.....	0	3	168	1
2016-08-16 14:46:44.423	4192.302	ICMP	192.168.5.100:0	->	192.168.56.15:8.0	.....	0	2	92	1
2016-08-16 14:46:44.423	4192.303	ICMP	192.168.5.100:0	->	192.168.56.15:13.0	.....	0	2	92	1

We notice some pings (ICMP type 8, code 0) and two ICMP timestamp requests (ICMP type 13, code 0). We have already covered that ICMP timestamp requests can be used for malicious purposes.

3.3.2 Practical Flow Analysis: Case 2

Detection

Let's look into UDP as well.

```
>> nfdump -o long -R . -A proto,dstport -O bytes 'ip
      192.168.5.100 and proto udp' | head -10
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 11:43:05.180	43254.352	UDP	0.0.0.0:0	->	0.0.0.0:53	.....	0	4406	289737	3970
2016-06-28 02:24:32.625	4307914.637	UDP	0.0.0.0:0	->	0.0.0.0:8572	.....	0	496	47954	234
2016-06-27 20:54:55.360	4309508.877	UDP	0.0.0.0:0	->	0.0.0.0:3544	.....	0	507	43359	63
2016-06-27 19:50:20.901	4327722.503	UDP	0.0.0.0:0	->	0.0.0.0:137	.....	0	415	34098	23
2016-06-27 22:15:16.998	4304687.239	UDP	0.0.0.0:0	->	0.0.0.0:49410	.....	0	206	28222	8
2016-06-28 04:24:32.963	4300365.056	UDP	0.0.0.0:0	->	0.0.0.0:40018	.....	0	87	15045	28
2016-08-16 11:50:31.397	11653.753	UDP	0.0.0.0:0	->	0.0.0.0:1900	.....	0	92	14766	46
2016-08-16 11:43:00.596	40562.491	UDP	0.0.0.0:0	->	0.0.0.0:5355	.....	0	239	13582	121
2016-06-27 21:07:42.908	4297087.208	UDP	0.0.0.0:0	->	0.0.0.0:53552	.....	0	93	12741	5

3.3.2 Practical Flow Analysis: Case 2

Detection

Let's look into UDP as well.

```
>> nfdump -o long -R . -A proto,dstport -O bytes 'ip
      192.168.5.100 and proto udp' | head -10
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 11:43:05.180	43254.352	UDP	0.0.0.0:0	->	0.0.0.0:53	.....	0	4406	289737	3970
2016-06-28 02:24:32.625	4307914.637	UDP	0.0.0.0:0	->	0.0.0.0:8572	.....	0	496	47954	234
2016-06-27 20:54:55.360	4309508.877	UDP	0.0.0.0:0	->	0.0.0.0:3544	.....	0	507	43359	63
2016-06-27 19:50:20.901	4327722.503	UDP	0.0.0.0:0	->	0.0.0.0:137	.....	0	415	34098	23
2016-06-27 22:15:16.998	4304687.239	UDP	0.0.0.0:0	->	0.0.0.0:49410	.....	0	206	28222	8
2016-06-28 04:24:32.963	4300365.056	UDP	0.0.0.0:0	->	0.0.0.0:40018	.....	0	87	15045	28
2016-08-16 11:50:31.397	11653.753	UDP	0.0.0.0:0	->	0.0.0.0:1900	.....	0	92	14766	46
2016-08-16 11:43:00.596	40562.491	UDP	0.0.0.0:0	->	0.0.0.0:5355	.....	0	239	13582	121
2016-06-27 21:07:42.908	4297087.208	UDP	0.0.0.0:0	->	0.0.0.0:53552	.....	0	93	12741	5

We notice MDNS-related, NBNS-related and UPNP-related flows. They seem normal though considering 192.168.5.100 is a Windows system.

3.3.2 Practical Flow Analysis: Case 2

Detection

To look into TCP and also sort by the number of flows, we execute:

```
>> nfdump -o long -R . -A proto,dstport -O flows 'ip
      192.168.5.100 and proto tcp' | head -10
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-06-27 19:54:24.039	4333271.235	TCP	0.0.0.0:0	->	0.0.0.0:80	.....	0	90403	7.0 M	2170
2016-08-16 14:49:39.451	4196.544	TCP	0.0.0.0:0	->	0.0.0.0:62604	.....	0	1774	71056	1754
2016-06-27 19:53:18.860	4333840.796	TCP	0.0.0.0:0	->	0.0.0.0:443	.....	0	36102	5.7 M	1227
2016-08-16 14:49:41.507	4194.926	TCP	0.0.0.0:0	->	0.0.0.0:62605	.....	0	410	16400	410
2016-06-27 23:02:11.313	4297677.538	TCP	0.0.0.0:0	->	0.0.0.0:22	.....	0	75846	89.0 M	250
2016-06-27 20:08:32.485	4319017.104	TCP	0.0.0.0:0	->	0.0.0.0:12345	.....	0	1.1 M	1.5 G	91
2016-06-27 19:57:56.447	4320052.050	TCP	0.0.0.0:0	->	0.0.0.0:12350	.....	0	1467	79577	19
2016-06-27 20:39:09.655	4303228.531	TCP	0.0.0.0:0	->	0.0.0.0:50006	.....	0	51	28786	11
2016-06-27 20:34:46.102	4303492.082	TCP	0.0.0.0:0	->	0.0.0.0:50000	.....	0	1004	1.3 M	11

3.3.2 Practical Flow Analysis: Case 2

Detection

To look into TCP and also sort by the number of flows, we execute:

```
>> nfdump -o long -R . -A proto,dstport -O flows 'ip
      192.168.5.100 and proto tcp' | head -10
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-06-27 19:54:24.039	4333271.235	TCP	0.0.0.0:0	->	0.0.0.0:80	.....	0	90403	7.0 M	2170
2016-08-16 14:49:39.451	4196.544	TCP	0.0.0.0:0	->	0.0.0.0:62604	.....	0	1774	71056	1754
2016-06-27 19:53:18.860	4333840.796	TCP	0.0.0.0:0	->	0.0.0.0:443	.....	0	36102	5.7 M	1227
2016-08-16 14:49:41.507	4194.926	TCP	0.0.0.0:0	->	0.0.0.0:62605	.....	0	410	16400	410
2016-06-27 23:02:11.313	4297677.538	TCP	0.0.0.0:0	->	0.0.0.0:22	.....	0	75846	89.0 M	250
2016-06-27 20:08:32.485	4319017.104	TCP	0.0.0.0:0	->	0.0.0.0:12345	.....	0	1.1 M	1.5 G	91
2016-06-27 19:57:56.447	4320052.050	TCP	0.0.0.0:0	->	0.0.0.0:12350	.....	0	1467	79577	19
2016-06-27 20:39:09.655	4303228.531	TCP	0.0.0.0:0	->	0.0.0.0:50006	.....	0	51	28786	11
2016-06-27 20:34:46.102	4303492.082	TCP	0.0.0.0:0	->	0.0.0.0:50000	.....	0	1004	1.3 M	11

Conversations to port 12345 stand out

3.3.2 Practical Flow Analysis: Case 2

Detection

To identify with whom 192.168.5.100 exchanged data on port 12345, we execute:

```
>> nfdump -o long -R . -A proto,srcip,dstip,dstport 'src ip
     192.168.5.100 and proto tcp and dst port 12345'
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 14:49:41.839	4194.136	TCP	192.168.5.100:0	->	192.168.5.10:12345	.....	0	2	92	2
2016-06-27 20:08:32.485	4319017.104	TCP	192.168.5.100:0	->	36.98.102.89:12345	.....	0	1.1 M	1.5 G	85
2016-08-16 14:49:44.104	4194.179	TCP	192.168.5.100:0	->	192.168.5.15:12345	.....	0	2	92	2
2016-08-16 15:59:31.538	0.115	TCP	192.168.5.100:0	->	192.168.5.1:12345	.....	0	2	92	2

OUTLINE

3.3.2 Practical Flow Analysis: Case 2

Detection

To identify with whom 192.168.5.100 exchanged data on port 12345, we execute:

```
>> nfdump -o long -R . -A proto,srcip,dstip,dstport 'src ip
     192.168.5.100 and proto tcp and dst port 12345'
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 14:49:41.839	4194.136	TCP	192.168.5.100:0	->	192.168.5.10:12345	.....	0	2	92	2
2016-06-27 20:08:32.485	4319017.104	TCP	192.168.5.100:0	->	36.98.102.89:12345	.....	0	1.1 M	1.5 G	85
2016-08-16 14:49:44.104	4194.179	TCP	192.168.5.100:0	->	192.168.5.15:12345	.....	0	2	92	2
2016-08-16 15:59:31.538	0.115	TCP	192.168.5.100:0	->	192.168.5.1:12345	.....	0	2	92	2

We notice that 36.98.102.89 is the main destination of the traffic on port 12345. Assume that we have already checked the case of port 12345 being a source port.

3.3.2 Practical Flow Analysis: Case 2

Detection

Network flows provide us with a bird's eye view of the whole network. Let's leverage that to see all 192.168.5.100 interactions with other machines on the intranet.

```
>> nfdump -o long -R . -A proto,srcip,dstip,dstport -O
 flows 'src ip 192.168.5.100 and proto tcp and dst net
                192.168.5.0/24' | head -10
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-06-27 23:02:11.313	4297677.538	TCP	192.168.5.100:0	->	192.168.5.10:22	.....	0	75840	89.0 M	244
2016-08-16 14:49:46.298	4195.050	TCP	192.168.5.100:0	->	192.168.5.15:8089	.....	0	4	184	4
2016-08-16 14:49:43.619	4199.837	TCP	192.168.5.100:0	->	192.168.5.15:648	.....	0	4	184	4
2016-08-16 15:59:37.920	6.028	TCP	192.168.5.100:0	->	192.168.5.15:3880	.....	0	4	184	4

3.3.2 Practical Flow Analysis: Case 2

Detection

Network flows provide us with a bird's eye view of the whole network. Let's leverage that to see all 192.168.5.100 interactions with other machines on the intranet.

```
>> nfdump -o long -R . -A proto,srcip,dstip,dstport -O
 flows 'src ip 192.168.5.100 and proto tcp and dst net
                192.168.5.0/24' | head -10
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-06-27 23:02:11.313	4297677.538	TCP	192.168.5.100:0	->	192.168.5.10:22	.....	0	75840	89.0 M	244
2016-08-16 14:49:46.298	4195.050	TCP	192.168.5.100:0	->	192.168.5.15:8089	.....	0	4	184	4
2016-08-16 14:49:43.619	4199.837	TCP	192.168.5.100:0	->	192.168.5.15:648	.....	0	4	184	4
2016-08-16 15:59:37.920	6.028	TCP	192.168.5.100:0	->	192.168.5.15:3880	.....	0	4	184	4

We notice traffic towards port 22 (SSH) of the 192.168.5.10 host. We should note this host down for further investigation.

3.3.2 Practical Flow Analysis: Case 2

Detection

If we try the same as previously and also include source ports, we may gain a better understanding of what happened.

```
>> nfdump -o long -R . -A proto,srcip,srcport,dstip -O
 flows 'src ip 192.168.5.100 and proto tcp and dst net
                192.168.5.0/24' | head -10
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 14:49:39.451	4196.709	TCP	192.168.5.100:62604	->	192.168.5.10:0	.....	0	1918	88228	1918
2016-08-16 14:49:41.996	4196.428	TCP	192.168.5.100:41476	->	192.168.5.15:0	.....	0	1702	78292	1702
2016-08-16 14:49:43.757	4205.362	TCP	192.168.5.100:41477	->	192.168.5.15:0	.....	0	1122	51612	1122

3.3.2 Practical Flow Analysis: Case 2

Detection

If we try the same as previously and also include source ports, we may gain a better understanding of what happened.

```
>> nfdump -o long -R . -A proto,srcip,srcport,dstip -O
 flows 'src ip 192.168.5.100 and proto tcp and dst net
                192.168.5.0/24' | head -10
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 14:49:39.451	4196.709	TCP	192.168.5.100:62604	->	192.168.5.10:0	.....	0	1918	88228	1918
2016-08-16 14:49:41.996	4196.428	TCP	192.168.5.100:41476	->	192.168.5.15:0	.....	0	1702	78292	1702
2016-08-16 14:49:43.757	4205.362	TCP	192.168.5.100:41477	->	192.168.5.15:0	.....	0	1122	51612	1122

We notice that the majority of the traffic derives from source ports 62604, 41476 and 41477. Each of those source ports connects to one IP address.

3.3.2 Practical Flow Analysis: Case 2

Detection

We can list all interactions from source port 62604, as follows.

```
>> nfdump -o long -R . 'src ip 192.168.5.100 and proto tcp
 and src port 62604 and dst net 192.168.5.0/24' | head -20
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 14:49:39.686	4194.296	TCP	192.168.5.100:62604	->	192.168.5.10:135	....S.	0	1	46	1
2016-08-16 14:49:39.686	4194.296	TCP	192.168.5.100:62604	->	192.168.5.10:443	....S.	0	1	46	1
2016-08-16 14:49:39.686	4194.297	TCP	192.168.5.100:62604	->	192.168.5.10:25	....S.	0	1	46	1
2016-08-16 14:49:39.686	4194.297	TCP	192.168.5.100:62604	->	192.168.5.10:1723	....S.	0	1	46	1
2016-08-16 14:49:39.686	4194.297	TCP	192.168.5.100:62604	->	192.168.5.10:445	....S.	0	1	46	1

3.3.2 Practical Flow Analysis: Case 2

Detection

We can list all interactions from source port 62604, as follows.

```
>> nfdump -o long -R . 'src ip 192.168.5.100 and proto tcp
 and src port 62604 and dst net 192.168.5.0/24' | head -20
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 14:49:39.686	4194.296	TCP	192.168.5.100:62604	->	192.168.5.10:135	....S.	0	1	46	1
2016-08-16 14:49:39.686	4194.296	TCP	192.168.5.100:62604	->	192.168.5.10:443	....S.	0	1	46	1
2016-08-16 14:49:39.686	4194.297	TCP	192.168.5.100:62604	->	192.168.5.10:25	....S.	0	1	46	1
2016-08-16 14:49:39.686	4194.297	TCP	192.168.5.100:62604	->	192.168.5.10:1723	....S.	0	1	46	1
2016-08-16 14:49:39.686	4194.297	TCP	192.168.5.100:62604	->	192.168.5.10:445	....S.	0	1	46	1

We notice very short flows and packets having the SYN bit set. No doubt the compromised host is being used to scan the intranet.

OUTLINE

3.3.2 Practical Flow Analysis: Case 2

Detection

Finally, we can also list connections from the intranet to the internet, as follows.

```
>> nfdump -o long -R . -A proto,srcip,dstip -O flows 'src
      ip 192.168.5.100 and proto tcp and ! dst net
                192.168.5.0/24' | head -20
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-08-16 11:50:07.935	11798.417	TCP	192.168.5.100:0	->	208.73.211.70:0	.....	0	366	81872	121
2016-06-27 19:56:55.016	4303025.732	TCP	192.168.5.100:0	->	54.229.228.176:0	.....	0	20908	1.1 M	114
2016-06-27 20:08:32.485	4319017.104	TCP	192.168.5.100:0	->	36.98.102.89:0	.....	0	1.1 M	1.5 G	85
2016-06-27 19:55:50.710	4313185.516	TCP	192.168.5.100:0	->	40.115.1.44:0	.....	0	1190	330607	73

3.3.2 Practical Flow Analysis: Case 2

Detection

By analyzing flows we have identified the following regarding 192.168.5.100.

1. Conversations with the 36.98.102.89 host, port 12345
2. It scanned some intranet hosts
3. It connected to the SSH port of the 192.168.5.10 host

3.3.2 Practical Flow Analysis: Case 2

Detection

By now, you should have an idea of how we can analyze network flows. But what about enriching them?

Suppose we were also given Squid proxy logs. Let's see how we can leverage them to enrich the provided network flows.

http://www.squid-cache.org/

OUTLINE

3.3.2 Practical Flow Analysis: Case 2

Detection

There are usually two important Squid proxy logs, *cache.log*, which is the internal log of the caching proxy itself and *access.log*, which contains all the passing URLs.

Let's focus on *access.log*.

3.3.2 Practical Flow Analysis: Case 2

Detection

For a better understanding let's remove all "benign" entries from *access.log*, as follows.

```
>> cat access.log | grep -v "ubuntu.com" | grep -v
"opensuse" | grep -v "openSUSE" | grep -v "novell.com"
```

```
1467994225.265    100 192.168.5.10 TCP_MISS/301 661 GET http://www.dfn-cert.de/index.html - HIER_DIRECT/193.174.13.92 text/html
1467994225.371     96 192.168.5.10 TCP_TUNNEL/200 17744 CONNECT www.dfn-cert.de:443 - HIER_DIRECT/193.174.13.92 -
1467998887.429      3 193.174.12.200 TCP_DENIED/403 3926 GET http://www.heise.de/ - HIER_NONE/- text/html
1468234574.617    266 192.168.5.15 TCP_MISS/200 185310 GET http://www.heise.de/ - HIER_DIRECT/193.99.144.85 text/html
1469198547.567    306 192.168.5.15 TCP_REFRESH_MODIFIED/200 181483 GET http://www.heise.de/ - HIER_DIRECT/193.99.144.85 text/html
1471356766.997     43 192.168.5.10 TCP_MISS/503 4151 GET http://bl/? - HIER_NONE/- text/html
1471356988.431  59783 192.168.5.10 TCP_MISS/503 4163 GET http://blog.mysportclub.ex/wp-content/uploads/hk/files/binaries-only.zip - HIER_DIR
ECT/54.229.228.176 text/html
1471357647.942  60185 192.168.5.10 TCP_MISS/503 4143 GET http://54.229.228.176/wp-content/uploads/hk/files/binaries-only.zip - HIER_DIRECT/5
4.229.228.176 text/html
```

3.3.2 Practical Flow Analysis: Case 2

Detection

For a better understanding let's remove all "benign" entries from *access.log*, as follows.

```
>> cat access.log | grep -v "ubuntu.com" | grep -v
"opensuse" | grep -v "openSUSE" | grep -v "novell.com"
```

```
1467994225.265    100 192.168.5.10 TCP_MISS/301 661 GET http://www.dfn-cert.de/index.html - HIER_DIRECT/193.174.13.92 text/html
1467994225.371     96 192.168.5.10 TCP_TUNNEL/200 17744 CONNECT www.dfn-cert.de:443 - HIER_DIRECT/193.174.13.92 -
1467998887.429      3 193.174.12.200 TCP_DENIED/403 3926 GET http://www.heise.de/ - HIER_NONE/- text/html
1468234574.617    266 192.168.5.15 TCP_MISS/200 185310 GET http://www.heise.de/ - HIER_DIRECT/193.99.144.85 text/html
1469198547.567    306 192.168.5.15 TCP_REFRESH_MODIFIED/200 181483 GET http://www.heise.de/ - HIER_DIRECT/193.99.144.85 text/html
1471356766.997     43 192.168.5.10 TCP_MISS/503 4151 GET http://bl/? - HIER_NONE/- text/html
1471356988.431  59783 192.168.5.10 TCP_MISS/503 4163 GET http://blog.mysportclub.ex/wp-content/uploads/hk/files/binaries-only.zip - HIER_DIR
ECT/54.229.228.176 text/html
1471357647.942  60185 192.168.5.10 TCP_MISS/503 4143 GET http://54.229.228.176/wp-content/uploads/hk/files/binaries-only.zip - HIER_DIRECT/5
4.229.228.176 text/html
```

A file named *binaries-only.zip* was obviously downloaded by the intranet host 192.168.5.10 from the 54.229.228.176 remote server.

3.3.2 Practical Flow Analysis: Case 2

Detection

Since we haven't came across this destination when analyzing the network flows. Let's now see what network flows have to say about this destination, as follows.

```
>> nfdump -o long -R . -A proto,srcip,srcport,dstip 'src ip
                54.229.228.176 and proto tcp'
```

```
Date first seen          Duration Proto      Src IP Addr:Port           Dst IP Addr:Port   Flags Tos  Packets    Bytes Flows
2016-06-27 19:56:31.305 4303049.443 TCP     54.229.228.176:80    ->     192.168.5.100:0     ......   0    34081   46.2 M   114
2016-06-27 23:33:04.934 4294942.354 TCP     54.229.228.176:80    ->      192.168.5.10:0     ......   0     1452    2.0 M     1
```


OUTLINE

3.3.2 Practical Flow Analysis: Case 2

Detection

Since we haven't came across this destination when analyzing the network flows. Let's now see what network flows have to say about this destination, as follows.

```
>> nfdump -o long -R . -A proto,srcip,srcport,dstip 'src ip
                54.229.228.176 and proto tcp'
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2016-06-27 19:56:31.305	4303049.443	TCP	54.229.228.176:80	->	192.168.5.100:0	.....	0	34081	46.2 M	114
2016-06-27 23:33:04.934	4294942.354	TCP	54.229.228.176:80	->	192.168.5.10:0	.....	0	1452	2.0 M	1

We notice that the 192.168.5.10 intranet host downloaded something from the 54.229.228.176 remote server, but we already knew that from our Squid log analysis. What we didn't know was that the 192.168.5.100 intranet host also downloaded something from the same server. This is obvious from the number of bytes transferred. In addition, we didn't see the 192.168.5.100 host downloading something when analyzing the Squid logs. The proxy was somehow bypassed, which is quite suspicious.

3.3.2 Practical Flow Analysis: Case 2

As you saw, the additional Squid proxy logs helped us move our investigation deeper (a.k.a enriched the network flows).

Credits go to ENISA for the data set we just analyzed.

3.3.3 Practical Flow Analysis: Case 3

Network flow tracking can facilitate the detection of **anomalous DNS activity**. Examples:

- Categorize DNS packets: DNS requests, DNS responses and unknown.
- Detect large fluctuations in DNS-related packet size per hour
- HTTP flows not preceded by a DNS request

3.3.3 Practical Flow Analysis: Case 3

Detection

Undoubtedly, if an incident responder comes across a flow report like the below, he/she should be concerned.

```
 Source Port:                                  Destination Port: 53
  Source I/F:                                   Destination I/F:
   Source AS:                                    Destination AS:
   TOS Field:                                         TCP Flag:
  Include if: Any part of flow in Time Period         Protocols: 17
Lines Cutoff: 100                                  Octets Cutoff:
```

Source	Destination	Flows	Octets	Packets	Avg Rate(bps)
10.100.100.24	172.24.100.13	7821	3.82 MB	20841	2,304
10.100.100.24	172.24.100.11	6269	2.55 MB	19068	1,982

3.3.3 Practical Flow Analysis: Case 3

Detection

Undoubtedly, if an incident responder comes across a flow report like the below, he/she should be concerned.

```
 Source Port:                                  Destination Port: 53
  Source I/F:                                   Destination I/F:
   Source AS:                                    Destination AS:
   TOS Field:                                         TCP Flag:
  Include if: Any part of flow in Time Period        Protocols: 17
Lines Cutoff: 100                                  Octets Cutoff:
```

Source	Destination	Flows	Octets	Packets	Avg Rate(bps)
10.100.100.24	172.24.100.13	7821	3.82 MB	20841	2,304
10.100.100.24	172.24.100.11	6269	2.55 MB	19068	1,982

We notice that the 10.100.100.24 intranet host has issued an immense number of DNS requests. Such a behavior is suspicious if and only if the 10.100.100.24 host is an end-user workstation and not a system that performs name resolutions repeatedly by design.

3.3.4 Practical Flow Analysis: Case 4

Network flow tracking can facilitate the detection of **anomalous SMB activity**. For example, consider the following network flow reports.

3.3.4 Practical Flow Analysis: Case 4

Detection

Flow report regarding inbound SMB traffic to 10.200.100.0/24 VLAN

```
     Source:                                        Destination: 10.200.100.0/24
Source Port:                                   Destination Port: 445
  Source I/F:                                    Destination I/F:
    Source AS:                                     Destination AS:
    TOS Field:                                          TCP Flag:
  Include if: Any part of flow in Time Period           Protocols:
Lines Cutoff: 100                                  Octets Cutoff:
Source          Destination     Flows     Octets          Packets     Avg Rate(bps)

10.100.100.101  10.200.100.97   116       30.42 KB        522         276
10.100.100.101  10.200.100.95   116       30.42 KB        522         276
10.100.100.101  10.200.100.68   116       30.42 KB        522         276
10.100.100.101  10.200.100.65   116       30.42 KB        522         276
10.100.100.101  10.200.100.73   116       30.42 KB        522         276
10.100.100.101  10.200.100.90   116       30.42 KB        522         276
10.100.100.101  10.200.100.82   116       30.42 KB        522         276
10.100.100.101  10.200.100.88   116       30.42 KB        522         276
10.100.100.101  10.200.100.77   116       30.42 KB        522         276
10.100.100.101  10.200.100.99   116       30.42 KB        522         276
10.100.100.101  10.200.100.75   116       30.42 KB        522         276
```

Flow report regarding outbound SMB traffic from the 10.200.100.0/24 VLAN

```
     Source: 10.200.100.0/24                        Destination:
Source Port: 445                               Destination Port:
  Source I/F:                                    Destination I/F:
    Source AS:                                     Destination AS:
    TOS Field:                                          TCP Flag:
  Include if: Any part of flow in Time Period           Protocols:
Lines Cutoff: 100                                  Octets Cutoff:
Source          Destination     Flows     Octets          Packets     Avg Rate(bps)

10.200.100.102  10.100.100.101  116       27.13 KB        290         246
10.200.100.104  10.100.100.101  116       24.07 KB        232         219
10.200.100.99   10.100.100.101  116       24.07 KB        232         219
10.200.100.95   10.100.100.101  116       24.07 KB        232         219
10.200.100.78   10.100.100.101  116       24.07 KB        232         219
10.200.100.113  10.100.100.101  116       24.07 KB        232         219
10.200.100.88   10.100.100.101  116       24.07 KB        232         219
10.200.100.89   10.100.100.101  116       24.07 KB        232         219
10.200.100.77   10.100.100.101  116       24.07 KB        232         219
10.200.100.70   10.100.100.101  116       24.07 KB        232         219
10.200.100.71   10.100.100.101  116       24.07 KB        232         219
```


OUTLINE

3.3.4 Practical Flow Analysis: Case 4

Detection

Flow report regarding inbound SMB traffic to 10.200.100.0/24 VLAN

```
     Source:                                    Destination: 10.200.100.0/24
Source Port:                                 Destination Port: 445
 Source I/F:                                  Destination I/F:
  Source AS:                                   Destination AS:
   TOS Field:                                       TCP Flag:
  Include if: Any part of flow in Time Period       Protocols:
Lines Cutoff: 100                              Octets Cutoff:
Source          Destination     Flows     Octets          Packets     Avg Rate(bps)

10.100.100.101  10.200.100.97   116       30.42 KB        522         276
10.100.100.101  10.200.100.95   116       30.42 KB        522         276
10.100.100.101  10.200.100.68   116       30.42 KB        522         276
10.100.100.101  10.200.100.65   116       30.42 KB        522         276
10.100.100.101  10.200.100.73   116       30.42 KB        522         276
10.100.100.101  10.200.100.90   116       30.42 KB        522         276
10.100.100.101  10.200.100.82   116       30.42 KB        522         276
10.100.100.101  10.200.100.88   116       30.42 KB        522         276
10.100.100.101  10.200.100.77   116       30.42 KB        522         276
10.100.100.101  10.200.100.99   116       30.42 KB        522         276
10.100.100.101  10.200.100.75   116       30.42 KB        522         276
```

Flow report regarding outbound SMB traffic from the 10.200.100.0/24 VLAN

```
     Source: 10.200.100.0/24                    Destination:
Source Port: 445                             Destination Port:
 Source I/F:                                  Destination I/F:
  Source AS:                                   Destination AS:
   TOS Field:                                       TCP Flag:
  Include if: Any part of flow in Time Period       Protocols:
Lines Cutoff: 100                              Octets Cutoff:
Source          Destination     Flows     Octets          Packets     Avg Rate(bps)

10.200.100.102  10.100.100.101  116       27.13 KB        290         246
10.200.100.104  10.100.100.101  116       24.07 KB        232         219
10.200.100.99   10.100.100.101  116       24.07 KB        232         219
10.200.100.95   10.100.100.101  116       24.07 KB        232         219
10.200.100.78   10.100.100.101  116       24.07 KB        232         219
10.200.100.113  10.100.100.101  116       24.07 KB        232         219
10.200.100.88   10.100.100.101  116       24.07 KB        232         219
10.200.100.89   10.100.100.101  116       24.07 KB        232         219
10.200.100.77   10.100.100.101  116       24.07 KB        232         219
10.200.100.70   10.100.100.101  116       24.07 KB        232         219
10.200.100.71   10.100.100.101  116       24.07 KB        232         219
```

In a Windows-based environment, traffic on port 445 (TCP) between workstations or between workstations and file servers is common. In this case though, we notice a machine reaching port 445 (TCP) on numerous other workstations.

We can't be certain that something is wrong, but note that such behavior has been spotted in the past by malware, such as the devastating conficker worm that used port 445 (TCP) to spread.

In addition, the small number of packets and their "distribution" across the VLAN is also curious-looking.

http://www.csl.sri.com/users/vinod/papers/Conficker/

3.3.5 Practical Flow Analysis: Case 5

Visualizing network flow data can oftentimes result in quicker and more efficient intrusion detection. A great tool for analyzing as well as visualizing network flow data is iSiLK. iSiLK is essentially a graphical front-end for the SiLK suite of tools.

https://tools.netsa.cert.org/isilk/index.html

3.3.5 Practical Flow Analysis: Case 5

Let's take for example, the SiLK command flow we used to identify top webservers.

```
>> rwfilter filename.rwf --sport=80,443,8080 --protocol=6 -
    -packets=4- --ack-flag=1 --pass=stdout | rwstats --
            fields=sip --percentage=1 --bytes
```

```
student@ubuntu:~/Data/isilk$ rwfilter ITOC-Border.rwf --sport=80,443,8080 --protocol=6 --packets=4- --ack-flag=1 --pass=stdout | rwstats --fields=sip --percentage=1 --bytes
INPUT: 71226 Records for 15 Bins and 250140493 Total Bytes
OUTPUT: Top 4 bins by Bytes (1% == 2501404)
                 sIP|      Bytes|    %Bytes|   cumul_%|
       10.1.60.187|  234969338| 93.934946| 93.934946|
         10.1.90.5|    2686023|  1.073806| 95.008752|
         10.1.20.4|    2581250|  1.031920| 96.040672|
        10.2.20.60|    2570521|  1.027631| 97.068303|
```

3.3.5 Practical Flow Analysis: Case 5

Let's do the same through iSiLK.

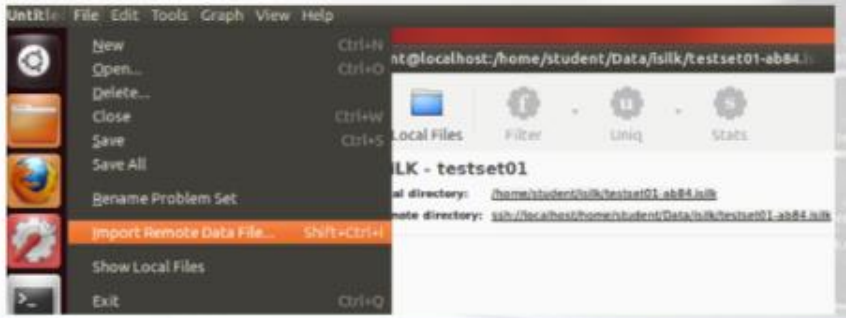
- http://tools.netsa.cert.org/isilk/isilk-admin-guide.pdf
- http://tools.netsa.cert.org/isilk/isilk-user-guide.pdf

3.3.5 Practical Flow Analysis: Case 5

First, execute the below to start iSiLK.

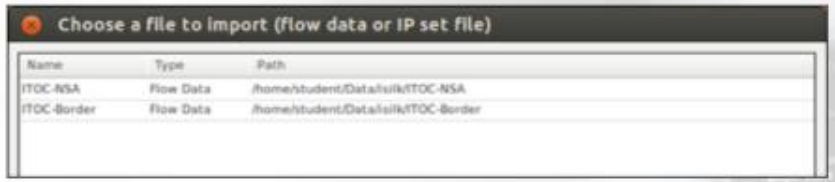
```
>> python isilk.py
```

Then, click on File -> Import Remote Data File

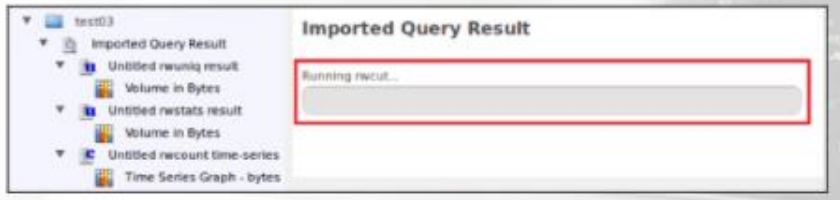


3.3.5 Practical Flow Analysis: Case 5

Choose the file you want to import and click OK. Example:



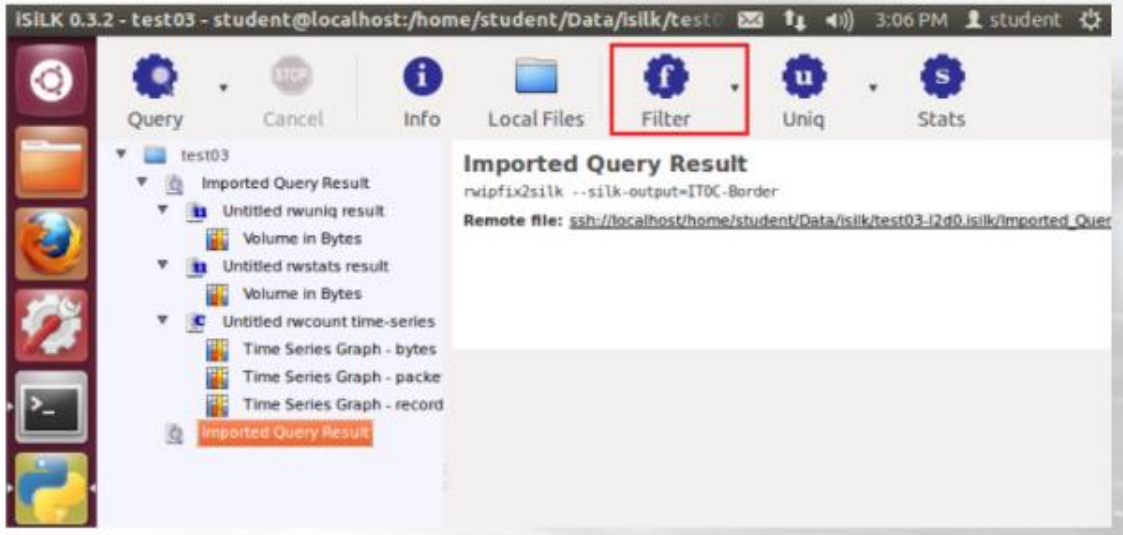
You should see something similar to the below.



OUTLINE

3.3.5 Practical Flow Analysis: Case 5

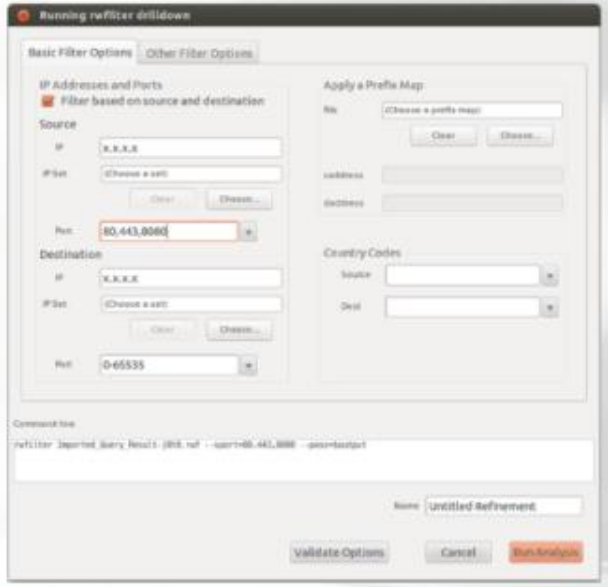
To replicate the first SiLK command, click on *Filter*.



OUTLINE

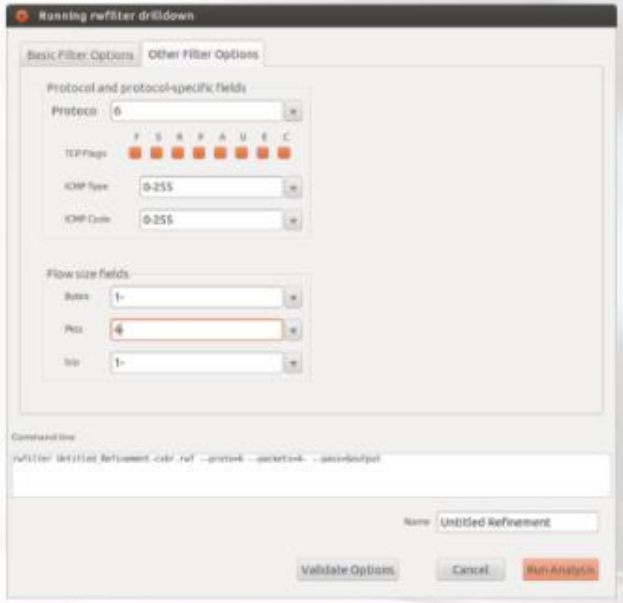
3.3.5 Practical Flow Analysis: Case 5

Then, specify the wanted source ports.



3.3.5 Practical Flow Analysis: Case 5

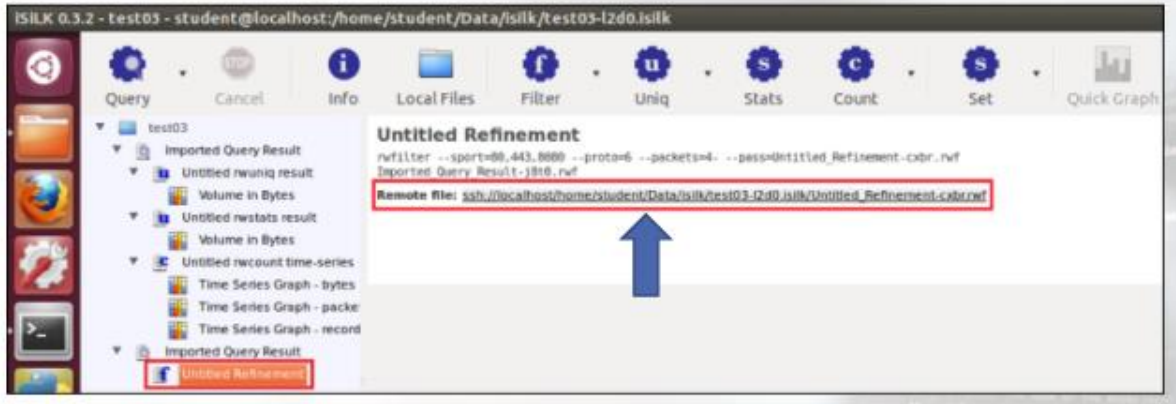
Also, specify the other filter options as follows and click *Run Analysis*.



OUTLINE

3.3.5 Practical Flow Analysis: Case 5

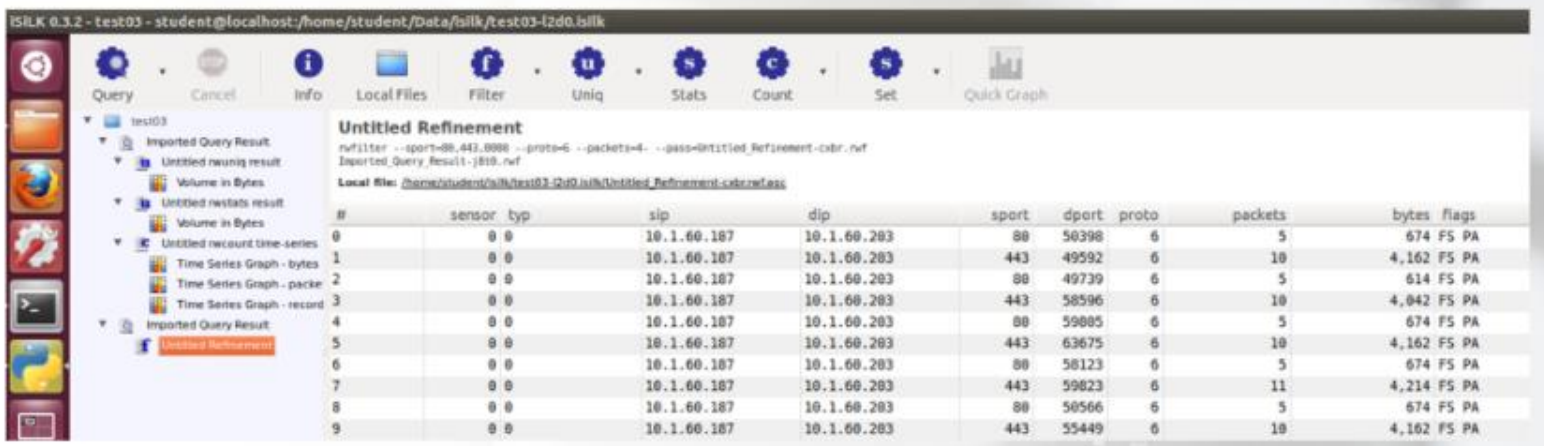
You will see something similar to the below. Click on it...



OUTLINE

3.3.5 Practical Flow Analysis: Case 5

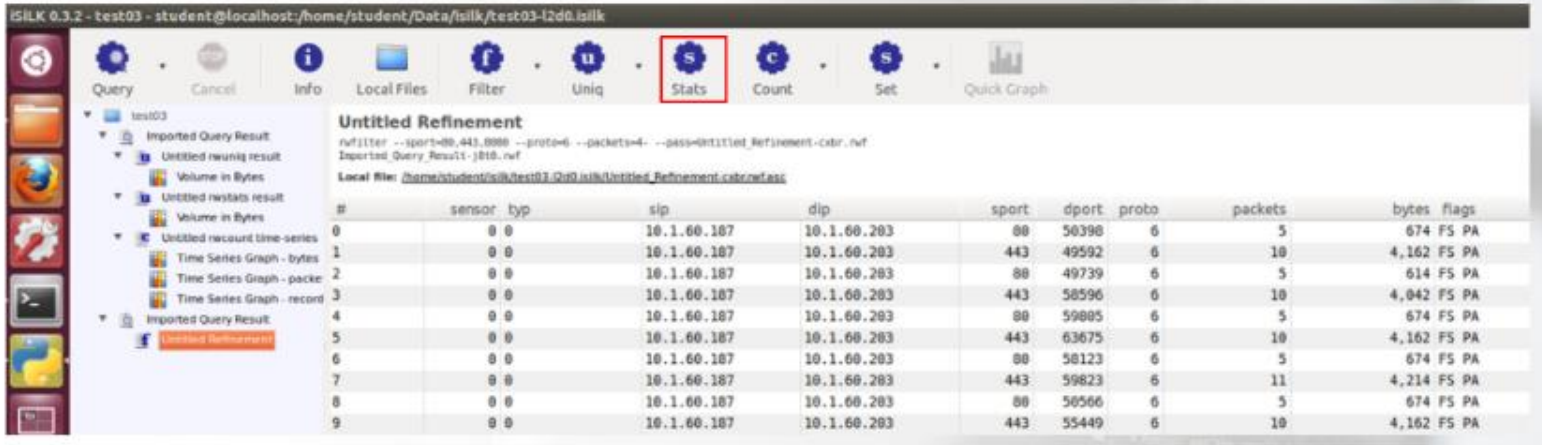
The *rwfilter*'s result will appear.



#	sensor	typ	sip	dip	sport	dport	proto	packets	bytes	flags
0	0	0	10.1.60.187	10.1.60.203	80	50398	6	5	674	FS PA
1	0	0	10.1.60.187	10.1.60.203	443	49592	6	10	4,162	FS PA
2	0	0	10.1.60.187	10.1.60.203	80	49739	6	5	614	FS PA
3	0	0	10.1.60.187	10.1.60.203	443	58596	6	10	4,042	FS PA
4	0	0	10.1.60.187	10.1.60.203	80	59805	6	5	674	FS PA
5	0	0	10.1.60.187	10.1.60.203	443	63675	6	10	4,162	FS PA
6	0	0	10.1.60.187	10.1.60.203	80	58123	6	5	674	FS PA
7	0	0	10.1.60.187	10.1.60.203	443	59823	6	11	4,214	FS PA
8	0	0	10.1.60.187	10.1.60.203	80	50566	6	5	674	FS PA
9	0	0	10.1.60.187	10.1.60.203	443	55449	6	10	4,162	FS PA

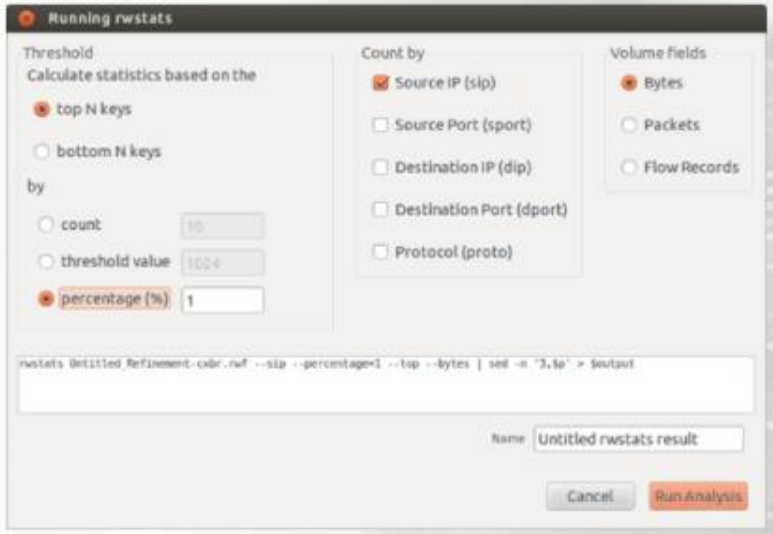
3.3.5 Practical Flow Analysis: Case 5

Now, to replicate the second part of the command flow, click on *Stats*.



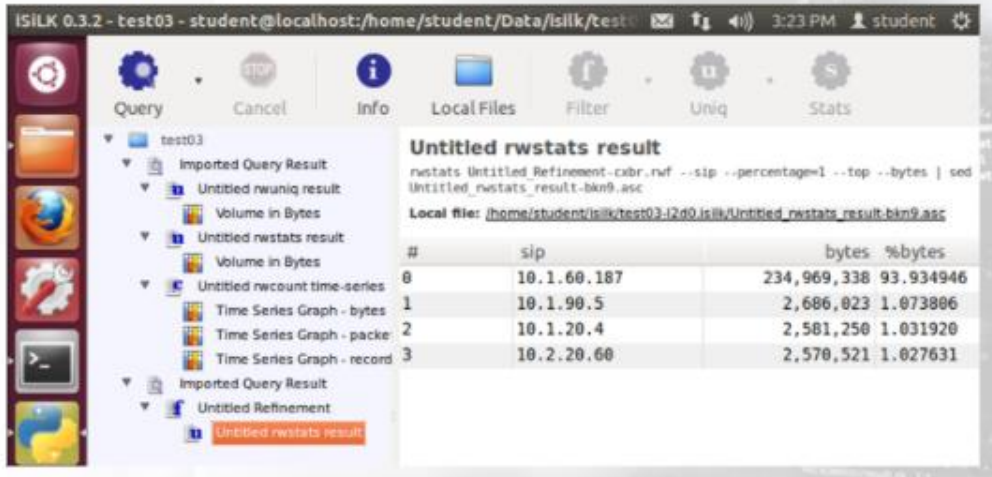
3.3.5 Practical Flow Analysis: Case 5

Finally, specify the running *rwstats* and click *Run Analysis*.



3.3.5 Practical Flow Analysis: Case 5

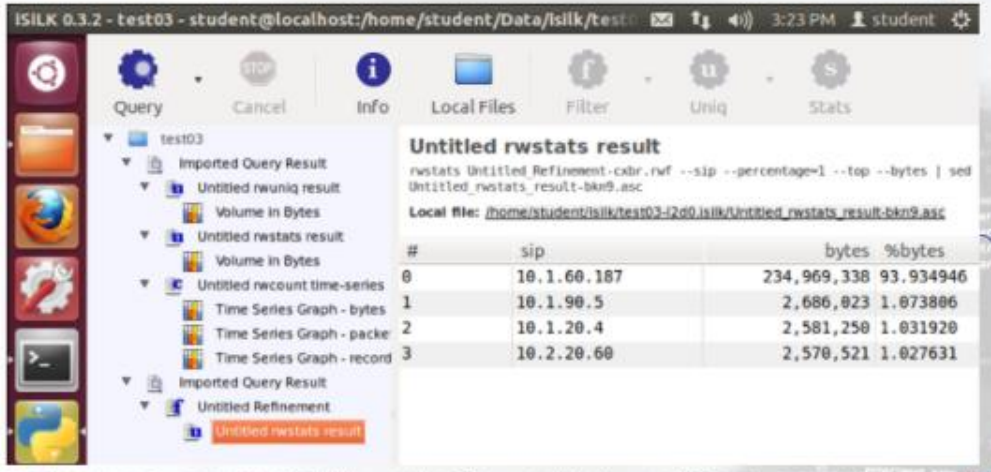
If you now click on the remote file that was created, the final results will appear.



OUTLINE

3.3.5 Practical Flow Analysis: Case 5

The results are the same as the ones create by SiLK.




```
student@ubuntu:~/Data/silk$ rwfilter ITOC-Border.rwf --sport=80,443,8080 --protocol=6 --packets=4- --ack-flag=1 --pass=stdout | rwstats --fields=sip --percentage=1 --bytes
INPUT: 71226 Records for 15 Bins and 250140493 Total Bytes
OUTPUT: Top 4 bins by Bytes (1% == 2501404)
                                sIP|            Bytes|    %Bytes|   cumul_%|
                       10.1.60.187|        234969338| 93.934946| 93.934946|
                         10.1.90.5|          2686023|  1.073806| 95.008752|
                         10.1.20.4|          2581250|  1.031920| 96.040672|
                        10.2.20.60|          2570521|  1.027631| 97.068303|
```


OUTLINE

3.3.5 Practical Flow Analysis: Case 5

iSiLK has many network visualization features and is great for learning SiLK. Spend some time to get comfortable with it.

OUTLINE

References

OUTLINE

References

NetFlow

https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

NetFlow for Cybersecurity

http://www.ciscopress.com/articles/article.asp?p=2812391&seqNum=3

YAF

https://tools.netsa.cert.org/yaf/

SiLK

https://tools.netsa.cert.org/silk/

References

FlowViewer

https://ensight.eos.nasa.gov/FlowViewer/

Application labeling

https://tools.netsa.cert.org/yaf/applabel.html

DHCP

http://tools.netsa.cert.org/yaf/yafdhcp.html

yaf-file-mediator

https://tools.netsa.cert.org/confluence/display/tt/YAF+2.x+IPFIX+File+Mediator

References

Deep packet inspection

https://tools.netsa.cert.org/yaf/yafdpi.html

rwcut

https://tools.netsa.cert.org/silk/rwcut.html

rwfilter

https://tools.netsa.cert.org/silk/rwfilter.html

rwfileinfo

https://tools.netsa.cert.org/silk/rwfileinfo.html

References

rwstats

https://tools.netsa.cert.org/silk/rwstats.html

rwcount

https://tools.netsa.cert.org/silk/rwcount.html

rwscan

https://tools.netsa.cert.org/silk/rwscan.html

Network Traffic Analysis - SiLK

https://schd.ws/hosted_files/flocon2017/fa/flocon-2016-silk-tutorial.pptx

References

Network Traffic Analysis with SiLK

https://tools.netsa.cert.org/silk/analysis-handbook.pdf

RVAsec: Jason Smith - Applied Detection and Analysis Using Flow Data

https://www.youtube.com/watch?v=ndfcfHiszHY

FlowViewer

https://ensight.eos.nasa.gov/FlowViewer/

flow-tools

https://github.com/adsr/flow-tools

References

CapLoader

https://www.netresec.com/?page=CapLoader#trial

PCAP: Traffic Analysis Exercise

http://www.malware-traffic-analysis.net/2015/03/09/

nfdump's man page

http://manpages.ubuntu.com/manpages/cosmic/en/man1/nfdump.1.html

Squid proxy

http://www.squid-cache.org/

References

ENISA

https://www.enisa.europa.eu/

conficker worm

http://www.csl.sri.com/users/vinod/papers/Conficker/

iSiLK

https://tools.netsa.cert.org/isilk/index.html

Development & Deployment Guide for iSiLK Version 0.1.2

http://tools.netsa.cert.org/isilk/isilk-admin-guide.pdf

References

User's Guide for iSiLK version 0.3.2

http://tools.netsa.cert.org/isilk/isilk-user-guide.pdf

OUTLINE