

Organizing and analyzing logdata with entropy

Sergey Bratus, Ph.D.

Institute for Security Technology Studies
Dartmouth College

Outline

- 1 Log browsing moves
 - Pipes and tables
 - Trees are better than pipes and tables!
- 2 Data organization
 - Trying to define the browsing problem
 - Entropy
 - Measuring co-dependence
 - Mutual Information
 - The tree building algorithm
- 3 Examples

What is this about?

Why?

- 1 To design a better interface for browsing logs & packets
- 2 A **smarter** interface that reacts to statistical properties of the data.
 - Show “anomalies” first
 - Show off correlations and where they break

How?

Design the browsing interface around

- Trees: natural for decision & classification
- Basic statistics for frequency distribution and correlation
 - Entropy, conditional entropy, mutual information, ...

How it started

- My wife ran a Tor node (kudos to Roger)
- Kept getting frantic messages from admins:

Your machine is compromised! There is IRC traffic! (IRC=evil)

- OK, but how would we really check if there is something besides the “normal” Tor mix?
- Ethereum isn't much help: how many page-long filters can you juggle?
- Wanted a tool that made **classification** simple.

Disclaimer

- 1 These are really simple tricks.
- 2 Not a survey of research literature (but see last slides).
 - You can do much cooler stuff with entropy & friends.
- 3 These tricks are for off-line browsing (“*analysis*”), not IDS/IPS magic.
 - but they might help you understand that magic.

Outline

- 1 Log browsing moves
 - Pipes and tables
 - Trees are better than pipes and tables!
- 2 Data organization
 - Trying to define the browsing problem
 - Entropy
 - Measuring co-dependence
 - Mutual Information
 - The tree building algorithm
- 3 Examples

The UNIX pipe length contest

What does this do?

```
grep 'Accepted password' /var/log/secure |
awk '{print $11}' | sort | uniq -c | sort -nr
```

/var/log/secure:

```
Jan 13 21:11:11 zion sshd[3213]: Accepted password for root from 209.61.200.11
Jan 13 21:30:20 zion sshd[3263]: Failed password for neo from 68.38.148.149
Jan 13 21:34:12 zion sshd[3267]: Accepted password for neo from 68.38.148.149
Jan 13 21:36:04 zion sshd[3355]: Accepted publickey for neo from 129.10.75.101
Jan 14 00:05:52 zion sshd[3600]: Failed password for neo from 68.38.148.149
Jan 14 00:05:57 zion sshd[3600]: Accepted password for neo from 68.38.148.149
Jan 14 12:06:40 zion sshd[5160]: Accepted password for neo from 68.38.148.149
Jan 14 12:39:57 zion sshd[5306]: Illegal user asmith from 68.38.148.149
Jan 14 14:50:36 zion sshd[5710]: Accepted publickey for neo from 68.38.148.149
```

And the question is:

```
44 68.38.148.149
12 129.10.75.101
 2 129.170.166.85
 1 66.183.80.107
 1 209.61.200.11
```

Successful logins via ssh using
password by IP address

...where is my WHERE clause?

What is this?

```
SELECT COUNT(*) as cnt, ip FROM logdata
GROUP BY ip ORDER BY cnt DESC
```

var.log.secure

serial	date	time	host	daemon	message	pid	ip	user
<input type="checkbox"/> 10	2006-01-13	21:11:11	zion	sshd[3213]	Accepted password for root from 209.61.200.11	3213	209.61.200.11	root
<input type="checkbox"/> 11	2006-01-13	21:30:20	zion	sshd[3263]	Failed password for neo from 68.38.148.149	3263	68.38.148.149	neo
<input type="checkbox"/> 12	2006-01-13	21:34:12	zion	sshd[3267]	Accepted password for neo from 68.38.148.149	3267	68.38.148.149	neo
<input type="checkbox"/> 13	2006-01-13	21:36:04	zion	sshd[3355]	Accepted publickey for neo from 129.10.75.101	3355	129.10.75.101	neo
<input type="checkbox"/> 14	2006-01-14	00:05:52	zion	sshd[3600]	Failed password for neo from 68.38.148.149	3600	68.38.148.149	neo
<input type="checkbox"/> 15	2006-01-14	00:05:57	zion	sshd[3600]	Accepted password for neo from 68.38.148.149	3600	68.38.148.149	neo
<input type="checkbox"/> 16	2006-01-14	12:06:40	zion	sshd[5160]	Accepted password for neo from 68.38.148.149	5160	68.38.148.149	neo
<input type="checkbox"/> 17	2006-01-14	12:39:57	zion	sshd[5306]	Illegal user asmith from 68.38.148.149	5306	68.38.148.149	asmith
<input type="checkbox"/> 18	2006-01-14	14:50:36	zion	sshd[5710]	Accepted publickey for neo from 68.38.148.149	5710	68.38.148.149	neo

	cnt	ip
<input type="checkbox"/>	44	68.38.148.149
<input type="checkbox"/>	12	129.10.75.101
<input type="checkbox"/>	2	129.170.166.85
<input type="checkbox"/>	1	66.183.80.107
<input type="checkbox"/>	1	209.61.200.11

(Successful logins via ssh using password by IP address)

Must... parse... syslog...

Wanted:

Free-text syslog records → named fields

Reality check

- *printf* format strings are at developers' discretion
- 120+ types of remote connections & user auth in Fedora Core

Pattern language

sshd:

Accepted %auth for %user from %host

Failed %auth for %user from %host

Failed %auth for illegal %user from %host

ftpd:

%host: %user[%pid]: FTP LOGIN FROM %host [%ip], %user

“The great cycle”



- 1 Filter
- 2 Group
- 3 Count
- 4 Sort
- 5 Rinse Repeat

```
grep user1 /var/log/messages | grep ip1 | grep ...
awk -f script ... | sort | uniq -c | sort -n
```

```
SELECT * FROM logtbl WHERE user = 'user1' AND ip = 'ip1'
GROUP BY ... ORDER BY ...
```

Outline

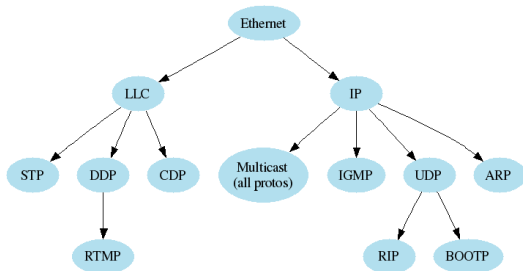
- 1 Log browsing moves
 - Pipes and tables
 - Trees are better than pipes and tables!
- 2 Data organization
 - Trying to define the browsing problem
 - Entropy
 - Measuring co-dependence
 - Mutual Information
 - The tree building algorithm
- 3 Examples

Can we do better than pipes & tables?

Humans naturally think in **classification trees**:

- Protocol hierarchies (e.g., Wireshark)
- Firewall decision trees (e.g., iptables chains)

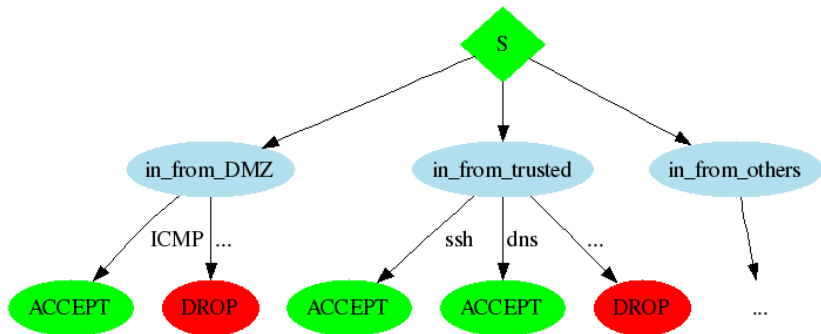
▼ Ethernet	100.00%
▼ Logical-Link Control	70.83%
Spanning Tree Protocol	50.00%
▼ Datagram Delivery Protocol	18.75%
Routing Table Maintenance Protocol	18.75%
Cisco Discovery Protocol	2.08%
▼ Internet Protocol	12.50%
Protocol Independent Multicast	4.17%
Internet Group Management Protocol	2.08%
▼ User Datagram Protocol	6.25%
Routing Information Protocol	4.17%
Bootstrap Protocol	2.08%
Address Resolution Protocol	16.67%



Can we do better than pipes & tables?

Humans naturally think in **classification trees**:

- Protocol hierarchies (e.g., Wireshark)
- Firewall decision trees (e.g., iptables chains)



Use tree views to show logs!

Pipes, SQL queries → branches / paths

Groups ↔ nodes (sorted by count / weight), records ↔ leaves.

TreeView

File Edit Template

[1339] Snort portscan alerts

- ⊕ **[1135]** dst_port: 445 src_ip: [55] dst_ip: [75]
- ⊕ **[70]** dst_port: 80 src_ip: [8] dst_ip: [30]
- ⊕ **[26]** dst_port: 21 src_ip: 80.141.141.173 dst_ip: [11]
- ⊕ **[22]** dst_port: 4899 src_ip: 218.103.195.242 dst_ip: [22]
- ⊕ **[20]** dst_port: 4000 src_ip: [2] dst_ip: [8]
- ⊕ **[15]** dst_port: 139 src_ip: 129.170.125.243 dst_ip: [8]
- ⊕ **[15]** dst_port: 443 src_ip: 211.5.239.5 dst_ip: [9]
- ⊕ **[12]** dst_port: 1524 src_ip: 192.139.15.34 dst_ip: [12]
- ⊕ **[9]** dst_port: 1 src_ip: 209.15.84.72 dst_ip: [9]
- ⊕ **[3]** dst_port: 8000 src_ip: 194.208.40.120 dst_ip: [2]
- ⊕ **[3]** dst_port: 1080 src_ip: 194.208.40.120 dst_ip: [2]
- ⊕ **[3]** dst_port: 3128 src_ip: 194.208.40.120 dst_ip: [2]
- ⊕ **[3]** dst_port: 8100 src_ip: 194.208.40.120 dst_ip: [2]
- ⊕ **[3]** dst_port: 8080 src_ip: 194.208.40.120 dst_ip: [2]
- ⊖ **[3]** src_ip: 194.208.40.120 dst_ip: [2]
 - Apr 15 19:54:10 annon snort: 194.208.40.120 4743 -> 129
 - Apr 15 19:55:00 annon snort: 194.208.40.120 4914 -> 129
 - Apr 15 19:55:06 annon snort: 194.208.40.120 4914 -> 129

Attributes

Field	#	Value
_day	15	
_minute	54	
type	SYN	
_month	Apr	
loghost	annon	
_hour	19	
_line	Apr 15 19:54:10 annon snort: 194.208.40.120 4743 -> 129	
dst_port	8080	
src_ip	194.208.40.120	
src_port	4743	
flags	*****S*	
program	snort	
dst_ip	129.170.166.39	
_second	10	

Use tree views to show logs!

Pipes, SQL queries → branches / paths

Groups ↔ nodes (sorted by count / weight), records ↔ leaves.

The screenshot shows the Wireshark interface with a packet list and a Min Entropy Tree view. The packet list shows four packets from 192.168.2.3 to 195.138.145.122. The Min Entropy Tree view shows a tree structure of nodes representing different fields and their values. The selected node is 'tcp.analysis.zero_window #undef' with a count of 18943. The 'Values' pane shows the distribution of values for this field, with 'Zero Window 1' having a count of 1.

No. .	Time	Source	Destination	Protocol	HwAddr
1	0.000000	192.168.2.3	195.138.145.122	UDP	00:11:50:38:81:70
2	0.031164	192.168.2.3	195.138.145.122	UDP	00:11:50:38:81:70
3	0.061035	192.168.2.3	195.138.145.122	UDP	00:11:50:38:81:70
4	0.072645	195.138.145.122	192.168.2.3	UDP	00:04:60:76:d9:ce

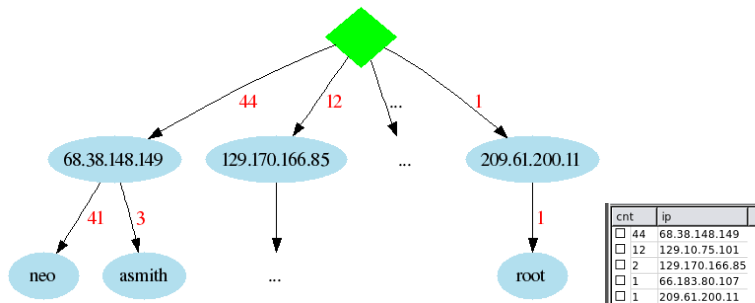
Field name	Ethernet formula	Unique values	Entropy	Values summary
tcp.flags.cwr		1	0.000000	0... = Congestion Window Reduced (CWR): N
ip.checksum_bad		1	0.000000	Bad : False
ip.hdr_len		1	0.000000	20
ip.flags.rb		1	0.000000	0... = Reserved bit: Not set
tcp.flags.ecn		1	0.000000	0... = ECN-Echo: Not set

Value	Count	Field name	Cond. entropy	Field 1 name	Field 2 name	Count
#undef	18943	frame.marked	0.000000	#undef	#undef	18941
Zero Window 1	1	image-jif.Xthumbnail	0.001072	#undef	TCP Sack Option: True	2
		image-jif.Ythumbnail	0.001072	Zero Window	#undef	1
		tcp.options.sack	0.001072			
		tcp.options.sack_re	0.001072			
		image-jif.marker	0.001072			

Use tree views to show logs!

Pipes, SQL queries → branches / paths

Groups ↔ **nodes** (sorted by count / weight), **records** ↔ **leaves**.
 Queries pick out a leaf or a node in the tree.

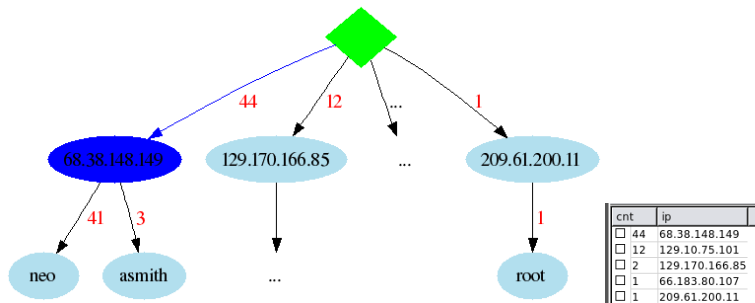


grep 68.38.148.149 /var/log/secure | **grep** asmith | **grep** ...

Use tree views to show logs!

Pipes, SQL queries → branches / paths

Groups ↔ nodes (sorted by count / weight), records ↔ leaves.
 Queries pick out a **leaf** or a **node** in the tree.

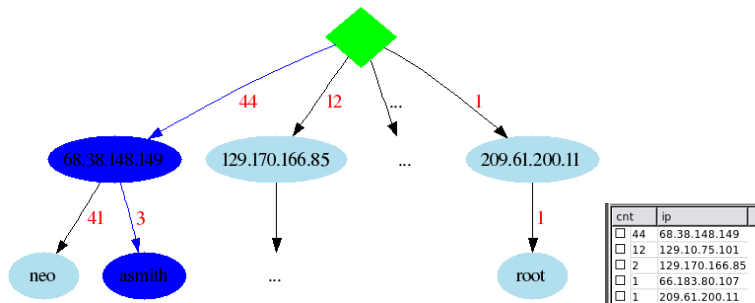


grep 68.38.148.149 /var/log/secure | **grep** asmith | **grep** ...

Use tree views to show logs!

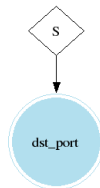
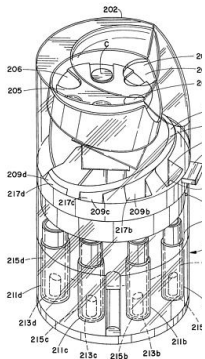
Pipes, SQL queries → branches / paths

Groups ↔ nodes (sorted by count / weight), records ↔ leaves.
 Queries pick out a **leaf** or a **node** in the tree.

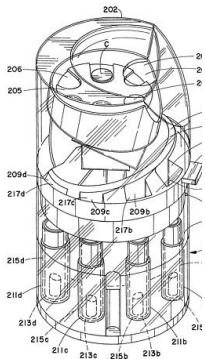


grep 68.38.148.149 /var/log/secure | **grep asmith** | **grep ...**

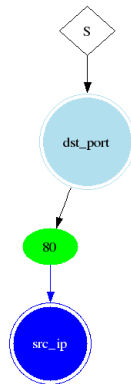
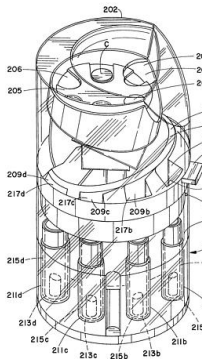
A “coin sorter” for records/packets



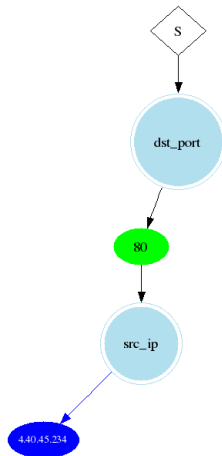
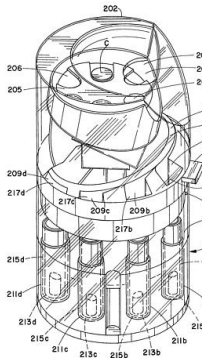
A “coin sorter” for records/packets



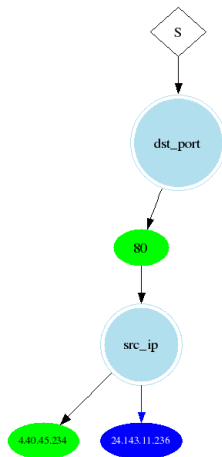
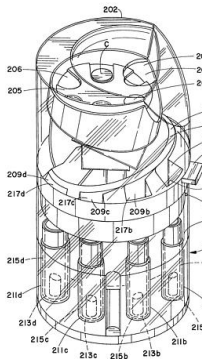
A “coin sorter” for records/packets



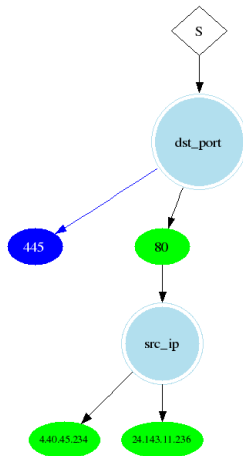
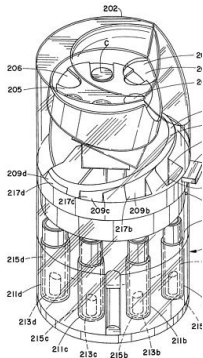
A “coin sorter” for records/packets



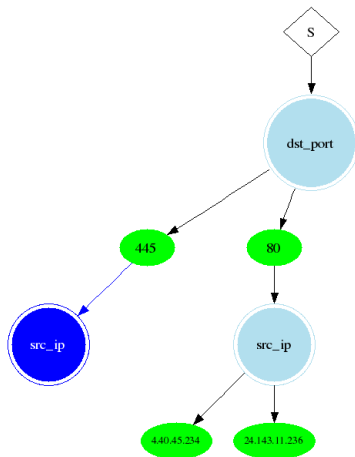
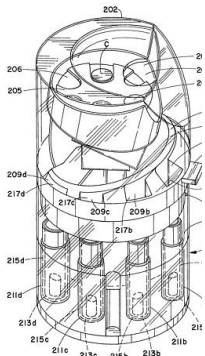
A “coin sorter” for records/packets



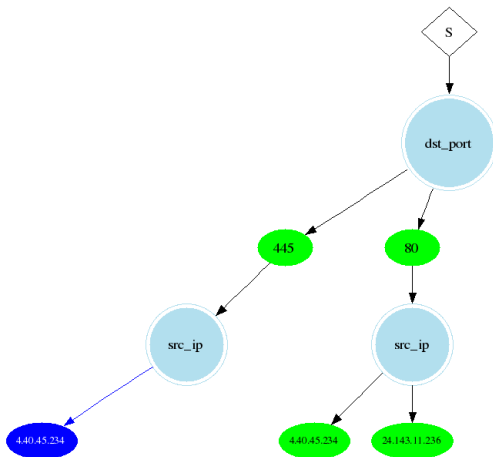
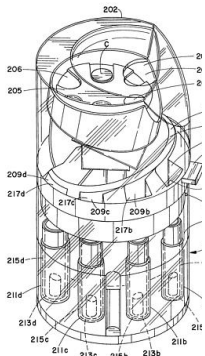
A “coin sorter” for records/packets



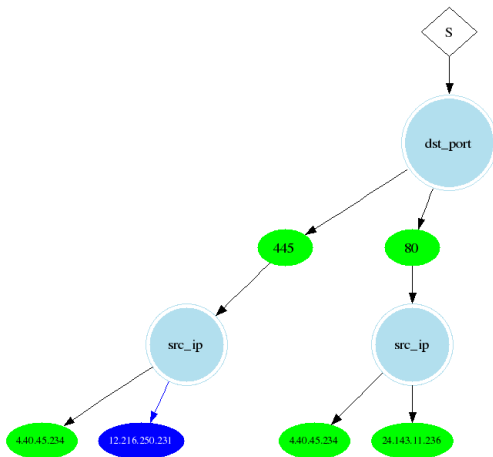
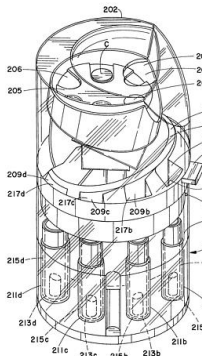
A “coin sorter” for records/packets



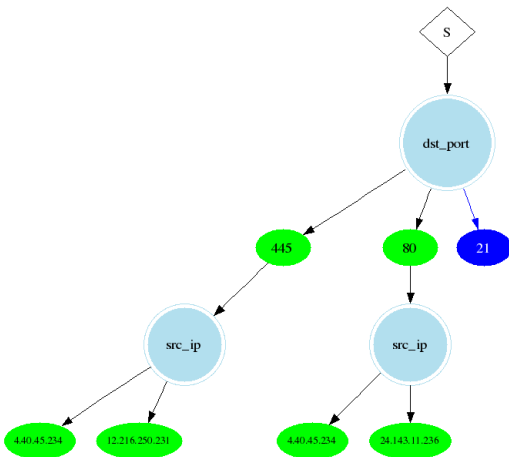
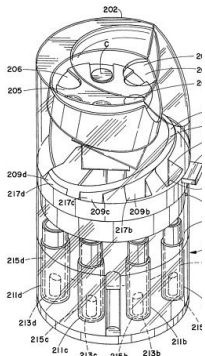
A “coin sorter” for records/packets



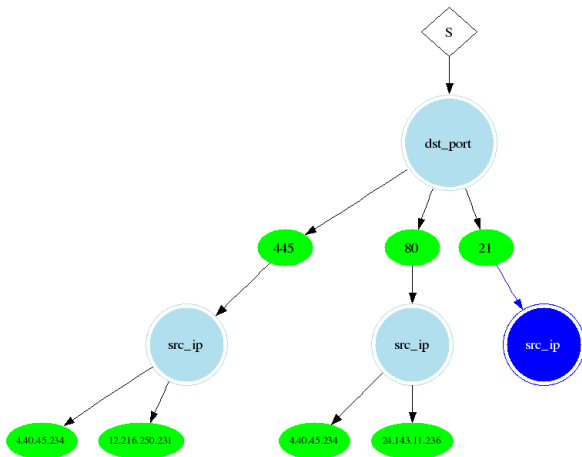
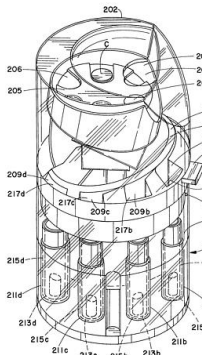
A “coin sorter” for records/packets



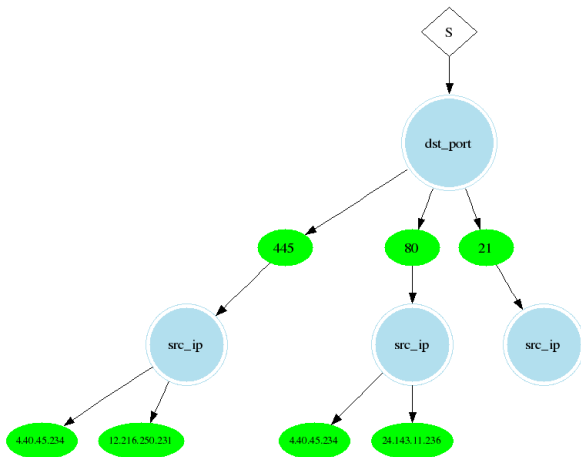
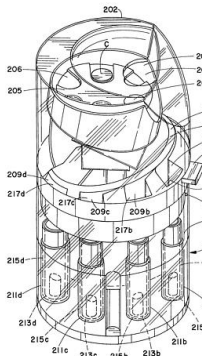
A “coin sorter” for records/packets



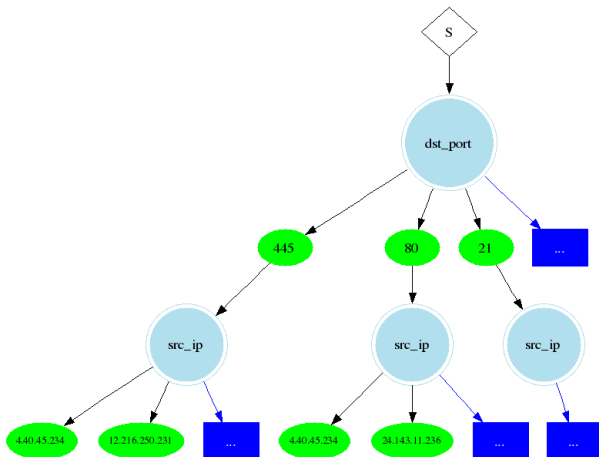
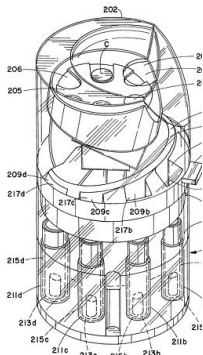
A “coin sorter” for records/packets



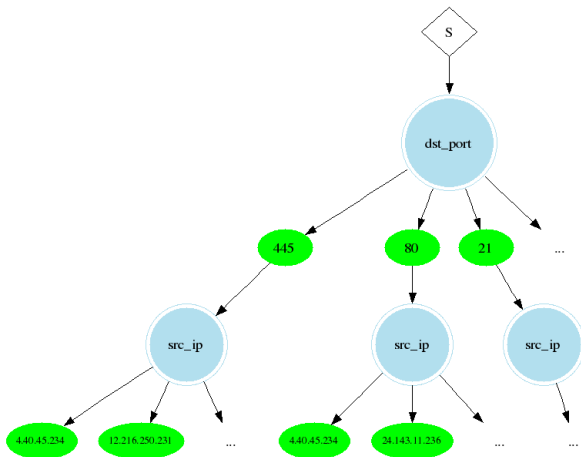
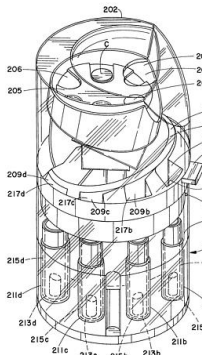
A “coin sorter” for records/packets



A “coin sorter” for records/packets

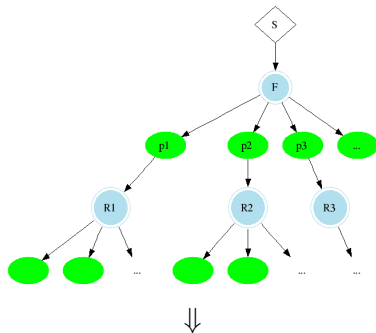


A “coin sorter” for records/packets



Classify → Save → Apply

TreeView2 source: syslog.0	
File	Edit View Help
root	
1647	LOGIN ON %tty BY %user FROM %host
1588	login by john from h000502032ae9.ne.jsp.net
1511	Aug 2002
1484	Sep 2002
1511	Oct 2002
1511	Nov 2002
1711	Dec 2002
1714	Jan 2003
1111	Feb 2003
111	Mar 2003
1201	Mar 25 12:31:31 mystic syslog: LOGIN ON nyp0 BY john FROM h000502032ae9.ne.jsp.net
1201	Mar 25 20:36:48 mystic syslog: LOGIN ON nyp0 BY john FROM h000502032ae9.ne.jsp.net
1301	login by igor from we-24-31-59-152.we.jsp.net
1311	login by john from h0010b565hb03.ne.jsp.net
1171	FAILED LOGIN %num=(d+)* FROM %host FOR %user, %reason
181	login by jos from h000502032ae9.ne.jsp.net
131	login by from h000502032ae9.ne.jsp.net
131	login by johns from h000502032ae9.ne.jsp.net
131	login by john*[]D from h000502032ae9.ne.jsp.net
131	login by [null] from h000502032ae9.ne.jsp.net
131	login by rj from h000502032ae9.ne.jsp.net



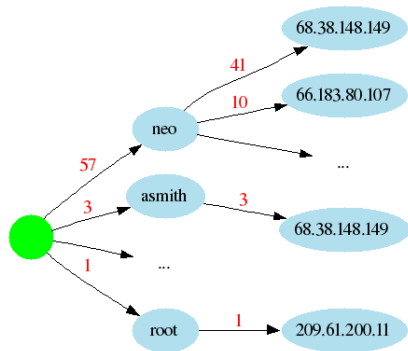
- 1 Build a classification tree from a dataset
- 2 Save template
- 3 Reuse on another dataset

TreeView2 source: Kerl/data/sst-with-users	
File	Edit View Help
1411	617 logins from mediaone.net
1441	host: h000502032ae9.ne.mediaone.net user: 7 tty: 5
1589	user: josh tty: 4
181	user: jos tty: ()
131	user: tty: ()
131	user: johs tty: ()
131	user: (null) tty: ()
131	user: rj tty: ()
131	user: josh* []D tty: ()
1111	host: we-24-31-59-152.we.mediaone.net user: [oleg] tty: 2
111	host: h0010b565hb03.ne.mediaone.net user: [josh] tty: ltypo()

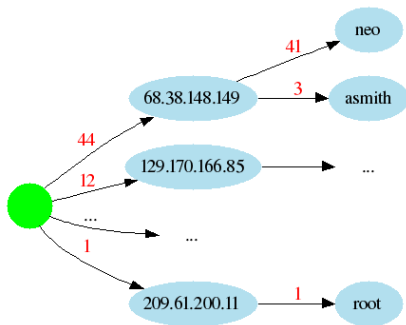
Fields	Slate	Features	Top	Index
Feature	E	#	Entropy	
H[Host#User]			(H 0.3691, 1.84)	
H[User#Host]			(H 0.0001, 0.00)	

Which tree to choose?

user \rightarrow ip?



ip \rightarrow user?



Goal: best grouping

How to choose the “best” grouping (tree shape) for a dataset?

Outline

- 1 Log browsing moves
 - Pipes and tables
 - Trees are better than pipes and tables!
- 2 Data organization
 - Trying to define the browsing problem
 - Entropy
 - Measuring co-dependence
 - Mutual Information
 - The tree building algorithm
- 3 Examples

Trying to define the browsing problem

- The lines you need are only 20 **PgDns** away:
- ...each one surrounded by a page of chaff...
- ...in a twisty maze of messages, all alike...
- ...but slightly different, in ways you don't expect.

Wireshark interface showing a network capture. The packet list pane displays the following packets:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.101.2	192.168.1.40	DNS	Standard query response A 192.168.4.2
2	0.000509	192.168.1.40	192.168.4.2	TCP	32790 > www [SYN] Seq=0 Ack=0 Win=5840 Len=0 MS
3	2.997399	192.168.1.40	192.168.4.2	TCP	32790 > www [SYN] Seq=0 Ack=0 Win=5840 Len=0 MS
4	5.211690	192.168.101.2	192.168.1.40	DNS	Standard query response A 192.168.7.2
5	5.211934	192.168.1.40	192.168.7.2	TCP	32791 > www [SYN] Seq=0 Ack=0 Win=5840 Len=0 MS
6	5.217693	192.168.7.2	192.168.1.40	TCP	www > 32791 [SYN, ACK] Seq=0 Ack=1 Win=17376 Len=0
7	5.217792	192.168.1.40	192.168.7.2	TCP	32791 > www [ACK] Seq=1 Ack=1 Win=5840 Len=0 TS
8	5.217865	192.168.1.40	192.168.7.2	HTTP	GET / HTTP/1.1
9	5.419530	192.168.7.2	192.168.1.40	TCP	www > 32791 [ACK] Seq=1 Ack=406 Win=17376 Len=0
10	6.634156	192.168.7.2	192.168.1.40	HTTP	HTTP/1.1 200 OK[Unreassembled Packet]
11	6.634441	192.168.1.40	192.168.7.2	TCP	32791 > www [ACK] Seq=406 Ack=1449 Win=8688 Len=0
12	6.635032	192.168.7.2	192.168.1.40	HTTP	Continuation or non-HTTP traffic
13	6.635237	192.168.7.2	192.168.1.40	HTTP	Continuation or non-HTTP traffic

The packet details pane for the selected packet (No. 1) shows:

- Frame 1 (112 bytes on wire, 112 bytes captured)
- Raw packet data
- Internet Protocol, Src Addr: 192.168.101.2 (192.168.101.2), Dst Addr: 192.168.1.40 (192.168.1.40)
- User Datagram Protocol, Src Port: domain (53), Dst Port: 32771 (32771)
- Domain Name System (response)

The response data is shown in hexadecimal and ASCII:

```

0000 45 00 00 70 45 79 40 00 3f 11 0e 89 c0 a8 65 02  E...pEy@. ?.....e.
0010 c0 a8 01 28 00 35 80 03 00 5c 7e 7e a1 82 85 00  ...(.S. .)\.....
0020 00 01 00 01 00 01 00 01 05 6f 72 65 65 06 06 72  .....green.r
0030 ef ef 74 66 75 02 6a 70 00 00 01 00 01 c0 0c 00  ootfu.jp .....
  
```

Trying to define the browsing problem

Wireshark interface showing a packet capture. The packet list pane displays the following data:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.101.2	192.168.1.40	DNS	Standard query response A 192.168.4.2
2	0.000509	192.168.1.40	192.168.4.2	TCP	32790 > www [SYN] Seq=0 Ack=0 Win=5840 Len=0 M...
3	2.997399	192.168.1.40	192.168.4.2	TCP	32790 > www [SYN] Seq=0 Ack=0 Win=5840 Len=0 M...
4	5.211690	192.168.101.2	192.168.1.40	DNS	Standard query response A 192.168.7.2
5	5.211934	192.168.1.40	192.168.7.2	TCP	32791 > www [SYN] Seq=0 Ack=0 Win=5840 Len=0 M...
6	5.217693	192.168.7.2	192.168.1.40	TCP	www > 32791 [SYN, ACK] Seq=0 Ack=1 Win=17376 L...
7	5.217792	192.168.1.40	192.168.7.2	TCP	32791 > www [ACK] Seq=1 Ack=1 Win=5840 Len=0 M...
8	5.217865	192.168.1.40	192.168.7.2	HTTP	GET / HTTP/1.1
9	5.419530	192.168.7.2	192.168.1.40	TCP	www > 32791 [ACK] Seq=1 Ack=406 Win=17376 L...
10	6.634156	192.168.7.2	192.168.1.40	HTTP	HTTP/1.1 200 OK[Unreassembled Packet]
11	6.634441	192.168.1.40	192.168.7.2	TCP	32791 > www [ACK] Seq=406 Ack=1449 Win=8688 L...
12	6.635032	192.168.7.2	192.168.1.40	HTTP	Continuation or non-HTTP traffic
13	6.635237	192.168.7.2	192.168.1.40	HTTP	Continuation or non-HTTP traffic

The packet details pane shows the following information for the selected packet (No. 13):

- Frame 1 (112 bytes on wire, 112 bytes captured)
- Raw packet data
- Internet Protocol, Src Addr: 192.168.101.2 (192.168.101.2), Dst Addr: 192.168.1.40 (192.168.1.40)
- User Datagram Protocol, Src Port: domain (53), Dst Port: 32771 (32771)
- Domain Name System (response)

The packet bytes pane shows the following hex and ASCII data:

```

0000 45 00 00 70 45 79 40 00 3f 11 0e 89 c0 a8 65 02  E...pEy@. ?.....e.
0010 c0 a8 01 28 00 35 80 03 00 5c 7e 7e a1 82 85 00  ...(.S.. ).....
0020 00 01 00 01 00 01 00 01 05 67 72 65 65 06 72  .....green.r
0030 ef ef 74 66 75 02 6a 70 00 00 01 00 01 c0 0c 00  ootfu.jp .....
  
```

- The lines you need are only 20 **PgDns** away:
- ...each one surrounded by a page of chaff...
- ...in a twisty maze of messages, all alike...
- ...but slightly different, in ways you don't expect.

Uncertainty

Old tricks

Wireshark interface showing a packet capture list and details pane. The packet list is sorted by Destination IP address. A red arrow points to the 'Destination' column header.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.40	192.168.1.40	ICMP	Standard query response s=192.168.1.40
2	0.000000	192.168.1.40	192.168.1.40	TCP	32790 → www [SN] Seq=0 Win=5840 Len=0 MS
3	2.897399	192.168.1.40	192.168.1.40	TCP	32790 → www [SN] Seq=0 Win=5840 Len=0 MS
4	5.211690	192.168.101.2	192.168.1.40	DNS	Standard query response s=192.168.7.2
5	5.211934	192.168.1.40	192.168.7.2	TCP	32791 → www [SN] Seq=0 Ack=0 Win=5840 Len=0 MS
6	5.217693	192.168.7.2	192.168.1.40	TCP	www → 32791 [SN, ACK] Seq=0 Ack=1 Win=17376 Len=0
7	5.217762	192.168.1.40	192.168.7.2	TCP	32791 → www [ACK] Seq=1 Win=5840 Len=0 TS
8	5.217895	192.168.1.40	192.168.7.2	HTTP	GET / HTTP/1.1
9	5.410930	192.168.7.2	192.168.1.40	TCP	www → 32791 [ACK] Seq=0 Ack=495 Win=17376 Len=0
10	6.634156	192.168.7.2	192.168.1.40	HTTP	HTTP/1.1 200 OK [text/css] Public
11	6.634441	192.168.1.40	192.168.7.2	TCP	32791 → www [ACK] Seq=406 Ack=1408 Win=8988 Len=0
12	6.635932	192.168.7.2	192.168.1.40	HTTP	Continuation or non-HTTP traffic
13	6.635237	192.168.7.2	192.168.1.40	HTTP	Continuation or non-HTTP traffic

Frame 1 (112 bytes on wire, 112 bytes captured)
 Raw packet data
 Internet Protocol, Src Addr: 192.168.101.2 (192.168.101.2), Dest Addr: 192.168.1.40 (192.168.1.40)
 User Datagram Protocol, Src Port: domain (53), Dest Port: 32771 (32771)
 Domain Name System (Response)
 0000 45 00 00 70 45 79 40 00 3f 11 04 85 c0 a8 05 02 6...00yh 7.....
 0010 c0 a8 01 28 00 35 80 03 00 5c 74 74 a1 82 85 00 ...(.S.)..green.
 0020 00 01 00 01 00 00 01 05 67 72 85 85 6e 06 72green.r
 0030 4f 4f 74 69 7c 52 5a 70 00 00 01 00 01 c0 0c 00 getfu-IP

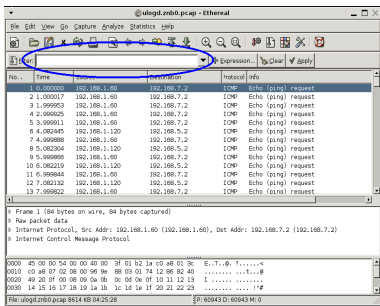
Sorting, grouping & filtering:

- Shows max and min values in a field
- Groups together records with the same values
- Drills down to an “interesting” group

Key problems:

- 1 Where to start? Which column or protocol feature to pick?
- 2 How to group? Which grouping helps best to understand the overall data?
- 3 How to automate guessing (1) and (2)?

Old tricks



Sorting, grouping & filtering:

- Shows max and min values in a field
- Groups together records with the same values
- Drills down to an “interesting” group

Key problems:

- 1 Where to start? Which column or protocol feature to pick?
- 2 How to group? Which grouping helps best to understand the overall data?
- 3 How to automate guessing (1) and (2)?

Old tricks

No.	Time	Source	Destination	Protocol	Info
0	0.000000	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
1	0.000017	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
2	1.999953	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
3	1.999965	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
4	2.999925	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
5	3.999911	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
6	4.002945	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
7	4.999888	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
8	5.002304	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
9	5.999895	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
10	6.002219	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
11	6.999844	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
12	7.002132	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
13	7.999822	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request

Frame 1 (64 bytes on wire, 84 bytes captured)
 Raw packet data
 Internet Protocol, Src Addr: 192.168.1.60 (192.168.1.60), Dest Addr: 192.168.7.2 (192.168.7.2)
 Internet Control Message Protocol

```

0000 45 00 00 54 00 00 40 00 3f 01 b2 1a c0 a8 01 3c  E..T..φ. ?.....<
0010 c0 a8 07 02 06 00 96 9e 86 03 01 74 12 86 82 40  .....T...φ
0020 49 20 0f 00 08 00 0a 0b 0c 0d 0e 0f 10 11 12 13  I.....
0030 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23  .....!??
  
```

File: ulogd.zmb0.pcap 8614 KB 04:25:28 P: 65943 D: 65943 M: 0

Sorting, grouping & filtering:

- Shows max and min values in a field
- Groups together records with the same values
- Drills down to an “interesting” group

Key problems:

- 1 Where to start? Which column or protocol feature to pick?
- 2 How to group? Which grouping helps best to understand the overall data?
- 3 How to automate guessing (1) and (2)?

Estimating uncertainty

Trivial observations

- Most lines in a large log will not be examined directly, ever.
- One just needs to convince oneself that he's seen everything interesting.
- "Jump straight to the interesting stuff", compress the rest.

Example

AAABAAAAAAAA...

The problem:

Must deal with **uncertainty** vs. **redundancy** of the data.

Measure it!

There is a **measure** of uncertainty/redundancy: **entropy**.

Estimating uncertainty

Trivial observations

- Most lines in a large log will not be examined directly, ever.
- One just needs to convince oneself that he's seen everything interesting.
- “Jump straight to the interesting stuff”, compress the rest.

Example

AAABAAAAAAAA...

The problem:

Must deal with **uncertainty** vs. **redundancy** of the data.

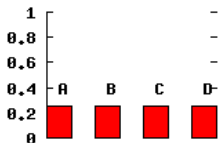
Measure it!

There is a **measure** of uncertainty/redundancy: **entropy**.

Entropy intuitions

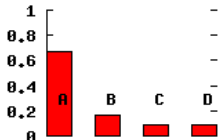
The number of bits to encode a data item under optimal encoding (asymptotically, in a very long stream)

Uniform distribution



ABBBACDBAAADBBDCAAACBCDACBBADAD
 CBBBAABADA... \Rightarrow 2 bits/symbol

Non-uniform



BAADBAAAAAABAAACABBABAAAAA
 BAAAADBAAC... \Rightarrow 1.42 bits/symbol

Entropy of English: 0.6 to 1.6 bits per char (best compression).

The entropy of English?

Depending on the model, 0.6 to 1.6 bits per character.

letters, unigrams	XFOML RXKHRJFFJUJ ZLPWCFWKCJY FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD ZEWRTZYNASADXESYJRQY WGECIJJ
bigrams	OCRO HLI RGWR NMIELWIS EU LL NBNESEBYATH EEI ALHENHTTPA OOBTTVA NAH BRL OR L RW NILI E NNSBATEI AI NGAE ITF NNR ASAEV OIE BAINTHA HYROO POER SETRYGAIETRWCO
trigrams	ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONSIVE TUOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE
words, unigrams	REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE HAD MESSAGES
bigrams	THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHOEVER
trigrams*	THE BEST FILM ON TELEVISION TONIGHT IS THERE NO-ONE HERE WHO HAD A LITTLE BIT OF FLUFF

Shannon's experiment

Based on **how likely humans are to be wrong when predicting the next letter / word** (the average number of guesses made to guess the next letter /word correctly)

<http://math.ucsd.edu/~crypto/java/ENTROPY/>

Automating old tricks (1)

“Look at the most frequent and least frequent values” in a column or list.

- What if there are many columns and batches of data?
- Which column to start with? How to rank them?

It would be nice to begin with “easier to understand” columns or features.

Suggestion:

- 1 Start with a data summary based on the columns with simplest value frequency charts (histograms).
- 2 Simplicity → less uncertainty → smaller **entropy**.

Automating old tricks (1)

“Look at the most frequent and least frequent values” in a column or list.

- What if there are many columns and batches of data?
- Which column to start with? How to rank them?

It would be nice to begin with “easier to understand” columns or features.

Suggestion:

- 1 Start with a data summary based on the columns with simplest value frequency charts (histograms).
- 2 Simplicity → less uncertainty → smaller **entropy**.

Automating old tricks (1)

“Look at the most frequent and least frequent values” in a column or list.

- What if there are many columns and batches of data?
- Which column to start with? How to rank them?

It would be nice to begin with “easier to understand” columns or features.

Suggestion:

- 1 Start with a data summary based on the columns with simplest value frequency charts (histograms).
- 2 Simplicity → less uncertainty → smaller **entropy**.

Trivial observations, visualized

uilogd.znb0.pcap - Ethereal

View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No.	Source	Destination	Protocol	Info
1	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
2	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
3	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
4	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
5	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
6	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
7	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
8	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
9	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
10	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
11	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
12	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
13	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request

Frame 1 (84 bytes on wire, 84 bytes captured)

- Raw packet data
- Internet Protocol, Src Addr: 192.168.1.60 (192.168.1.60), Dst Addr: 192.168.7.2 (192.168.7.2)
- Internet Control Message Protocol

```

0000 45 00 00 54 00 00 40 00 3f 01 b2 1a c0 a8 01 3c  E..T..@. ?.....<
0010 c0 a8 07 02 08 00 96 9e 98 03 01 74 12 86 82 40  .....t..t..@
0020 45 20 0f 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13  I .....
0030 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23  .....!"#
  
```

File: uilogd.znb0.pcap 8614 KB 04:25:28 P: 60943 D: 60943 M: 0

Outline

- 1 Log browsing moves
 - Pipes and tables
 - Trees are better than pipes and tables!
- 2 Data organization
 - Trying to define the browsing problem
 - Entropy
 - Measuring co-dependence
 - Mutual Information
 - The tree building algorithm
- 3 Examples

Start simple: Ranges

The screenshot shows the Wireshark interface with the following details:

- Filter:** `dest ip 192.168.1.1 - 192.168.10.100` (indicated by a red arrow from a callout box)
- Packet List:**

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
2	1.000017	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
3	1.999953	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
4	2.999925	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
5	3.999911	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
6	4.082445	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
7	4.999688	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
8	5.082304	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
9	5.999866	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
10	6.082219	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
11	6.999844	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
12	7.082132	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
13	7.999622	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
- Packet Details:**
 - Frame 1 (84 bytes on wire (84 bytes captured))
 - Raw packet data
 - Internet Protocol, Src Addr: 192.168.1.60 (192.168.1.60), Dst Addr: 192.168.7.2 (192.168.7.2)
 - Internet Control Message Protocol
- Packet Bytes:**

```

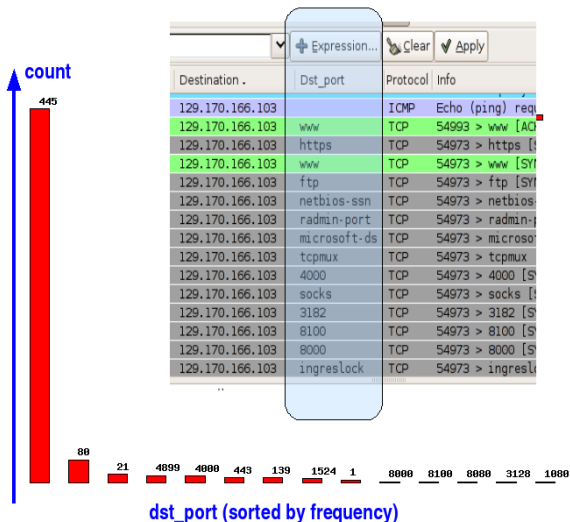
0000 45 00 00 34 00 00 40 00 3f 01 b2 1a c0 a8 01 3c  E..T..@. 7.....<
0010 c0 a8 07 02 08 00 96 9e 88 03 01 74 12 86 82 40  .....t.t..@
0020 49 20 0f 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13  I .....
0030 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23  .....!#

```
- Status Bar:** File: ulogd.znb0.pcap 8614 KB 04:25:28 | P: 60943 D: 60943 M: 0

Additional callouts:

- dest ip 192.168.1.1 - 192.168.10.100** (points to the filter)
- dst port 1-8100** (points to the filter)
- src ip 192.168.1.1 - 192.168.255.255** (points to the filter)

A frequency histogram



Start simple: Histograms

The screenshot displays the Wireshark interface with a packet capture of ICMP Echo (ping) requests. The packet list pane shows 13 packets, all originating from 192.168.1.60 and destined for 192.168.7.2. The details pane shows the raw packet data, including the Internet Protocol and Internet Control Message Protocol fields.

Annotations and histograms:

- dest ip:** 192.168.1.1 - 192.168.10.100 (Histogram at top left)
- dst port:** 1-8100 (Histogram at top right)
- src ip:** 192.168.1.1 - 192.168.255.255 (Histogram at bottom left)

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
2	1.000017	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
3	1.999953	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
4	2.999925	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
5	3.999911	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
6	4.082445	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
7	4.999688	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
8	5.082304	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
9	5.999866	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
10	6.082219	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
11	6.999844	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
12	7.082132	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
13	7.999622	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request

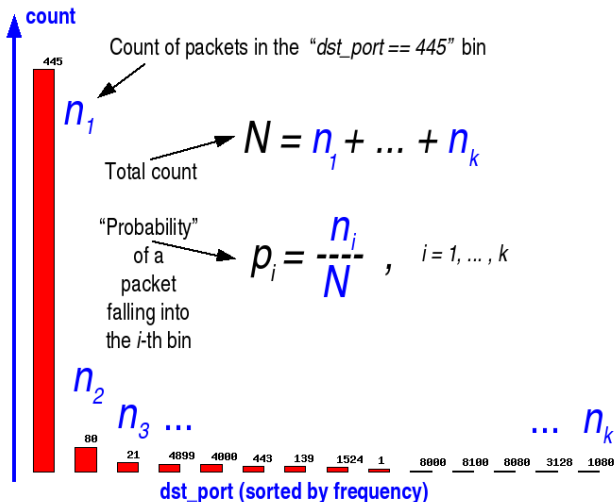
Frame 1 (84 bytes on wire (84 bytes captured) on interface 0):
 Raw packet data
 Internet Protocol, Src Addr: 192.168.1.60 (192.168.1.60), Dst Addr: 192.168.7.2 (192.168.7.2)
 Internet Control Message Protocol

```

0000 45 00 00 34 00 00 40 00 3f 01 b2 1a c0 a8 01 3c  E..T..@. 7.....<
0010 c0 a8 07 02 08 00 96 9e 88 03 01 74 12 86 82 40  .....t...@
0020 49 20 0f 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13  I .....
0030 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23  .....!#
  
```

File: ulogd.znb0.pcap 8614 KB 04:25:28 P: 60943 D: 60943 M: 0

Probability distribution



Definition of entropy

Let a random variable X take values x_1, x_2, \dots, x_k with probabilities p_1, p_2, \dots, p_k .

Definition (Shannon, 1948)

The **entropy** of X is

$$H(X) = \sum_{i=1}^k p_i \cdot \log_2 \frac{1}{p_i}$$

Recall that the probability of value x_i is $p_i = n_i/N$ for all $i = 1, \dots, k$.

- 1 Entropy measures the uncertainty or **lack of information** about the values of a variable.
- 2 Entropy is related to the **number of bits** needed to encode the missing information (to full certainty).

Why logarithms?

Fact:

The least number of bits needed to encode numbers between 1 and N is $\log_2 N$.

Example

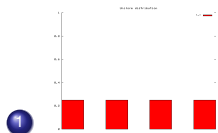
- You are to receive one of N objects, equally likely to be chosen.
- What is the measure of your uncertainty?

Answer in the spirit of Shannon:

The number of bits needed to communicate the number of the object (and thus remove all uncertainty), i.e. $\log_2 N$.

If some object is more likely to be picked than others, uncertainty decreases.

Entropy on a histogram



Interpretation

Entropy is a measure of **uncertainty** about the value of X

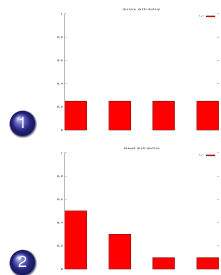
1 $X = (.25 \ .25 \ .25 \ .25) : H(X) = 2$ (bits)

2 $X = (.5 \ .3 \ .1 \ .1) : H(X) = 1.685$

3 $X = (.8 \ .1 \ .05 \ .05) : H(X) = 1.022$

4 $X = (1 \ 0 \ 0 \ 0) : H(X) = 0$

Entropy on a histogram



Interpretation

Entropy is a measure of **uncertainty** about the value of X

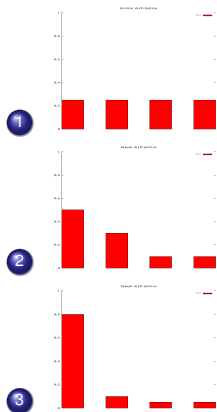
1 $X = (.25 \ .25 \ .25 \ .25) : H(X) = 2$ (bits)

2 $X = (.5 \ .3 \ .1 \ .1) : H(X) = 1.685$

3 $X = (.8 \ .1 \ .05 \ .05) : H(X) = 1.022$

4 $X = (1 \ 0 \ 0 \ 0) : H(X) = 0$

Entropy on a histogram



Interpretation

Entropy is a measure of **uncertainty** about the value of X

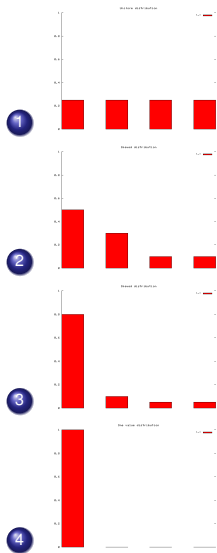
1 $X = (.25 \ .25 \ .25 \ .25) : H(X) = 2$ (bits)

2 $X = (.5 \ .3 \ .1 \ .1) : H(X) = 1.685$

3 $X = (.8 \ .1 \ .05 \ .05) : H(X) = 1.022$

4 $X = (1 \ 0 \ 0 \ 0) : H(X) = 0$

Entropy on a histogram

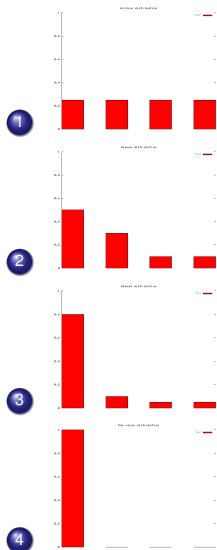


Interpretation

Entropy is a measure of **uncertainty** about the value of X

- 1** $X = (.25 \ .25 \ .25 \ .25) : H(X) = 2$ (bits)
- 2** $X = (.5 \ .3 \ .1 \ .1) : H(X) = 1.685$
- 3** $X = (.8 \ .1 \ .05 \ .05) : H(X) = 1.022$
- 4** $X = (1 \ 0 \ 0 \ 0) : H(X) = 0$

Entropy on a histogram



Interpretation

Entropy is a measure of **uncertainty** about the value of X

① $X = (.25 \ .25 \ .25 \ .25) : H(X) = 2$ (bits)

② $X = (.5 \ .3 \ .1 \ .1) : H(X) = 1.685$

③ $X = (.8 \ .1 \ .05 \ .05) : H(X) = 1.022$

④ $X = (1 \ 0 \ 0 \ 0) : H(X) = 0$

For only one value, the entropy is 0.

When all N values have the same frequency, the entropy is maximal, $\log_2 N$.

Compare histograms

The screenshot shows the Wireshark interface with a packet capture of ICMP Echo (ping) requests. Three histograms are overlaid on the interface, each with a red arrow pointing to a specific field in the packet list table.

- Top-left histogram:** Labeled $H=5.79$. A red arrow points from this histogram to the 'Destination' column of the packet list table.
- Top-right histogram:** Labeled $H=1.07$. A red arrow points from this histogram to the 'Protocol' column of the packet list table.
- Bottom histogram:** Labeled $H=6.00$. A red arrow points from this histogram to the 'Source' column of the packet list table.

The packet list table shows the following data:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
2	1.000017	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
3	1.999953	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
4	2.999925	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
5	3.999911	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
6	4.082445	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
7	4.999688	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
8	5.082304	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
9	5.999866	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
10	6.082219	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
11	6.999844	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request
12	7.082132	192.168.1.120	192.168.5.2	ICMP	Echo (ping) request
13	7.999622	192.168.1.60	192.168.7.2	ICMP	Echo (ping) request

The packet details pane shows the following information:

```

Frame 1 (84 bytes on wire (84 bytes captured)
Raw packet data
Internet Protocol, Src Addr: 192.168.1.60 (192.168.1.60), Dst Addr: 192.168.7.2 (192.168.7.2)
Internet Control Message Protocol
  
```

The packet bytes pane shows the following hex data:

```

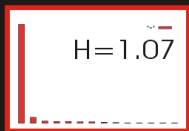
0000 45 00 00 34 00 00 40 00 3f 01 b2 1a c0 a8 01 3c  E..T..@. 7.....<
0010 c0 a8 07 02 08 00 96 9e 88 03 01 74 12 86 82 40  .....t...@
0020 49 20 0f 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13  I .....
0030 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23  .....!#
  
```

The file information pane shows the following information:

```

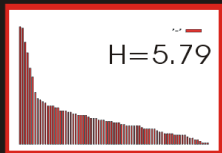
File: ulogd.znb0.pcap 8614 KB 04:25:28
P: 60943 D: 60943 M: 0
  
```

Start with the simplest

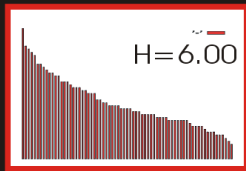


dst port
1-8100

I am the simplest!



dest ip
192.168.1.1 - 192.168.10.100



src ip
192.168.1.1 - 192.168.255.255

A tree grows in Ethereal

The screenshot shows the Ethereal network protocol analyzer interface. The main window displays a list of captured packets. The selected packet (No. 4) is a TCP packet from 192.168.2.3 to 195.138.145.122. The interface is split into several panes:

- Packet List:** Shows a list of captured packets with columns for No., Time, Source, Destination, Protocol, and HwAddr.
- Min Entropy Tree:** A tree view showing the hierarchical structure of the selected packet's fields. The tree is expanded to show the 'tcp.analysis.zero_window' field.
- Ranges:** A table showing the unique values and entropy for various fields in the packet.

Field name	Ethereal formula	Unique values	Entropy	Values summary
tcp.flags.cwr		1	0.000000	0.. = Congestion Window Reduced (CWR): N
ip.checksum_bad		1	0.000000	Bad : False
ip.hdr_len		1	0.000000	20
ip.flags.rb		1	0.000000	0.. = Reserved bit: Not set
tcp.flags.ecn		1	0.000000	0.. = ECN-Echo: Not set
- Table:** A table showing the conditional entropy for various fields.

Value	Count	Field name	Cond. entropy
#undef	18943	frame.marked	0.000000
Zero Window 1		image-jfff.Xthumbnail	0.001072
		image-jfff.Ythumbnail	0.001072
		tcp.options.sack	0.001072
		tcp.options.sack_re	0.001072
		image-iff.marker	0.001072
- Bottom Panel:** Shows the file path and the packet details for the selected packet: File: "/home/stefan/ethereal/data/data1" 5464 KB 00:04:47 P: 18944 D: 18944 M: 0

Outline

- 1 Log browsing moves
 - Pipes and tables
 - Trees are better than pipes and tables!
- 2 Data organization
 - Trying to define the browsing problem
 - Entropy
 - **Measuring co-dependence**
 - Mutual Information
 - The tree building algorithm
- 3 Examples

Automating old tricks (2)

“Look for correlations. If two fields are strongly correlated on average, but for some values the correlation breaks, look at those more closely”.

- Which pair of fields to start with?
- How to rank correlations?

Too many to try by hand, even with a good graphing tool like R or Matlab.

Suggestion:

- 1 Try and rank pairs before looking, and look at the simpler correlations first.
- 2 Simplicity \longrightarrow stronger correlation between features \longrightarrow smaller **conditional entropy**.

Automating old tricks (2)

“Look for correlations. If two fields are strongly correlated on average, but for some values the correlation breaks, look at those more closely”.

- Which pair of fields to start with?
- How to rank correlations?

Too many to try by hand, even with a good graphing tool like R or Matlab.

Suggestion:

- 1 Try and rank pairs before looking, and look at the simpler correlations first.
- 2 Simplicity \longrightarrow stronger correlation between features \longrightarrow smaller **conditional entropy**.

Automating old tricks (2)

“Look for correlations. If two fields are strongly correlated on average, but for some values the correlation breaks, look at those more closely”.

- Which pair of fields to start with?
- How to rank correlations?

Too many to try by hand, even with a good graphing tool like R or Matlab.

Suggestion:

- 1 Try and rank pairs before looking, and look at the simpler correlations first.
- 2 Simplicity \longrightarrow stronger correlation between features \longrightarrow smaller **conditional entropy**.

Examples (1)

Example

Source IP of user logins:

- Almost everyone comes in from a couple of machines
- One user comes in from all over the place. *Problem?*

Example

Small network, SRC_IP \sim TTL

- On average, src_ip predicts ttl.
- What if a host sends packets with all sorts of ttl?
 - *A user just discovered traceroute?*
 - *What if that machine is a printer or appliance?*

Examples (2)

MUD: Multi-user text adventure (like WoW in ASCII text, only better PvP)

Example

```
%user gets %obj [%objnum] in room %room
```

- 2 rooms had by far the largest number of objects picked up.
- Major source of money in the game was: **robbers!**
 - Stationary camp, safe area, close to cities, easy kill...

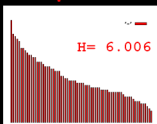
Example

Cheating: player killing by agreement for experience

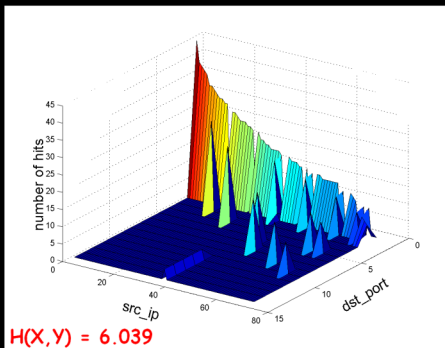
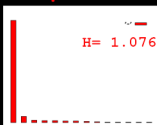
- A kills B repeatedly, often in the same room. *Why?*
- A gets experience, warpoints, levels. B is used as a throw-away character, owner of B gets favors.

Histograms 3d: Feature pairs

src_ip



dst_port



Joint Entropy

For fields X and Y , count # times n_{ij} a pair (x_i, y_j) . is seen together in the same record.

	y_1	y_2	\dots
x_1	n_{11}	n_{12}	\dots
x_2	n_{21}	n_{22}	\dots
\vdots	\vdots	\vdots	\ddots

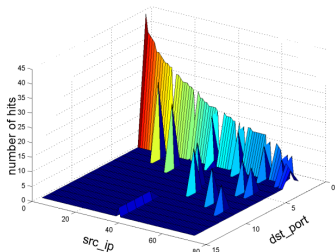
$$p(x_i, y_j) = \frac{n_{ij}}{N}, \quad (N = \sum_{i,j} n_{ij})$$

Joint Entropy

$$H(X, Y) = \sum_{ij} p(x_i, y_j) \cdot \log_2 \frac{1}{p(x_i, y_j)}$$

Joint Entropy

For fields X and Y , count # times n_{ij} a pair (x_i, y_j) . is seen together in the same record.



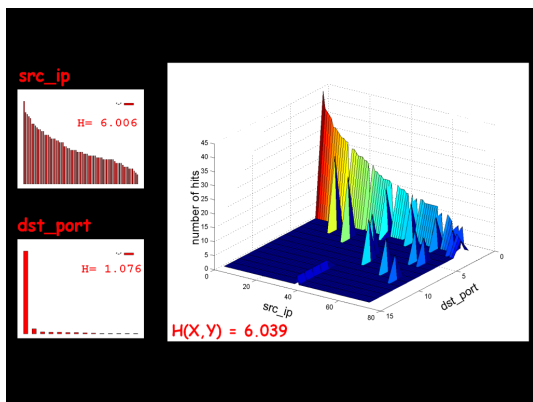
$$p(x_i, y_j) = \frac{n_{ij}}{N}, \quad (N = \sum_{i,j} n_{ij})$$

Joint Entropy

$$H(X, Y) = \sum_{ij} p(x_i, y_j) \cdot \log_2 \frac{1}{p(x_i, y_j)}$$

Measure of mutual dependence

- How much knowing X tells about Y (on average)?
- How strong is the connection?



Compare:

$H(X, Y)$ and $H(X)$

Compare:

$H(X) + H(Y)$ and $H(X, Y)$

Dependence

Independent variables X and Y :

- Knowing X tells us nothing about Y
- No matter what x we fix, *the histogram of Y 's values co-occurring with that x will be the same shape*
- $H(X, Y) = H(X) + H(Y)$

Dependent X and Y :

- Knowing X tells us something about Y (and vice versa)
- Histograms of ys co-occurring with a fixed x have *different shapes*
- $H(X, Y) < H(X) + H(Y)$

Dependence

Independent variables X and Y :

- Knowing X tells us nothing about Y
- No matter what x we fix, *the histogram of Y 's values co-occurring with that x will be the same shape*
- $H(X, Y) = H(X) + H(Y)$

Dependent X and Y :

- Knowing X tells us something about Y (and vice versa)
- Histograms of ys co-occurring with a fixed x have *different shapes*
- $H(X, Y) < H(X) + H(Y)$

Dependence

Independent variables X and Y :

- Knowing X tells us nothing about Y
- No matter what x we fix, *the histogram of Y 's values co-occurring with that x will be the same shape*
- $H(X, Y) = H(X) + H(Y)$

Dependent X and Y :

- Knowing X tells us something about Y (and vice versa)
- Histograms of ys co-occurring with a fixed x have *different shapes*
- $H(X, Y) < H(X) + H(Y)$

Dependence

Independent variables X and Y :

- Knowing X tells us nothing about Y
- No matter what x we fix, *the histogram of Y 's values co-occurring with that x will be the same shape*
- $H(X, Y) = H(X) + H(Y)$

Dependent X and Y :

- Knowing X tells us something about Y (and vice versa)
- Histograms of ys co-occurring with a fixed x have *different shapes*
- $H(X, Y) < H(X) + H(Y)$

Dependence

Independent variables X and Y :

- Knowing X tells us nothing about Y
- No matter what x we fix, *the histogram of Y 's values co-occurring with that x will be the same shape*
- $H(X, Y) = H(X) + H(Y)$

Dependent X and Y :

- Knowing X tells us something about Y (and vice versa)
- Histograms of ys co-occurring with a fixed x have **different shapes**
- $H(X, Y) < H(X) + H(Y)$

Dependence

Independent variables X and Y :

- Knowing X tells us nothing about Y
- No matter what x we fix, *the histogram of Y 's values co-occurring with that x* will be **the same shape**
- $H(X, Y) = H(X) + H(Y)$

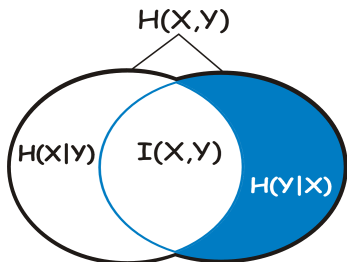
Dependent X and Y :

- Knowing X tells us something about Y (and vice versa)
- Histograms of ys co-occurring with a fixed x have **different shapes**
- $H(X, Y) < H(X) + H(Y)$

Outline

- 1 Log browsing moves
 - Pipes and tables
 - Trees are better than pipes and tables!
- 2 Data organization
 - Trying to define the browsing problem
 - Entropy
 - Measuring co-dependence
 - **Mutual Information**
 - The tree building algorithm
- 3 Examples

Mutual Information



Definition

Conditional entropy of Y given X

$$H(Y|X) = H(X, Y) - H(X)$$

Uncertainty about Y left once we know X .

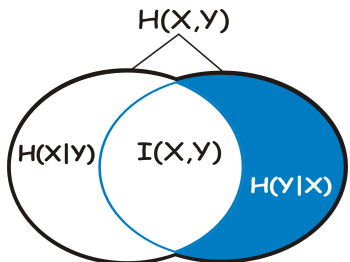
Definition

Mutual information of two variables X and Y

$$I(X; Y) = H(X) + H(Y) - H(X, Y)$$

Reduction in uncertainty about X once we know Y and vice versa.

Mutual Information

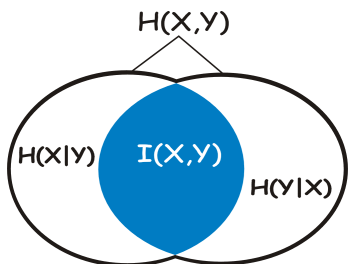


Definition

Conditional entropy of Y given X

$$H(Y|X) = H(X, Y) - H(X)$$

Uncertainty about Y left once we know X .



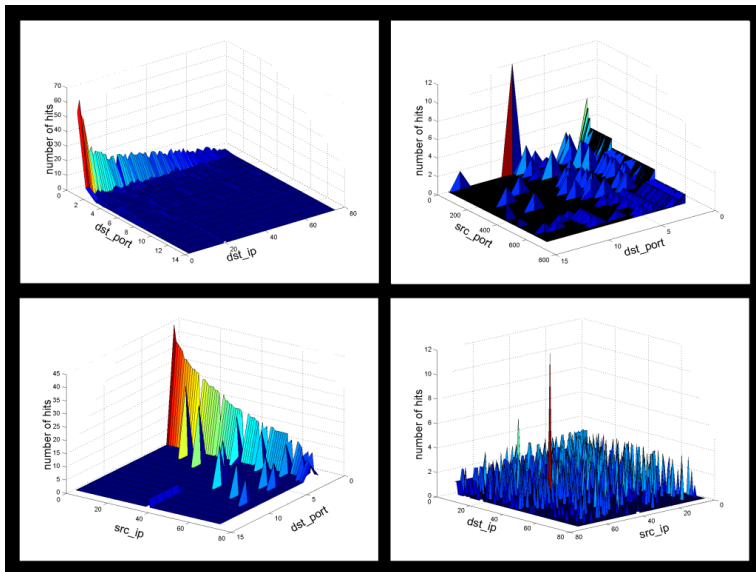
Definition

Mutual information of two variables X and Y

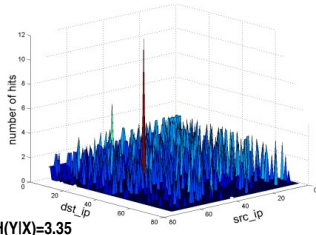
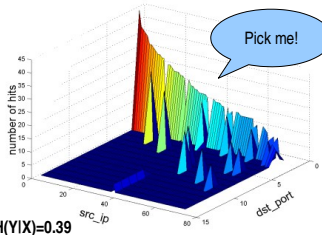
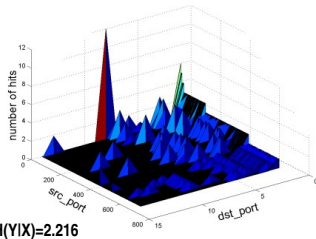
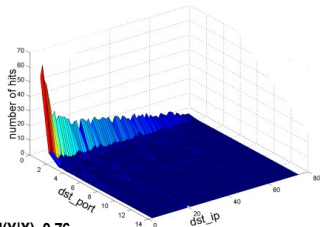
$$I(X; Y) = H(X) + H(Y) - H(X, Y)$$

Reduction in uncertainty about X once we know Y and vice versa.

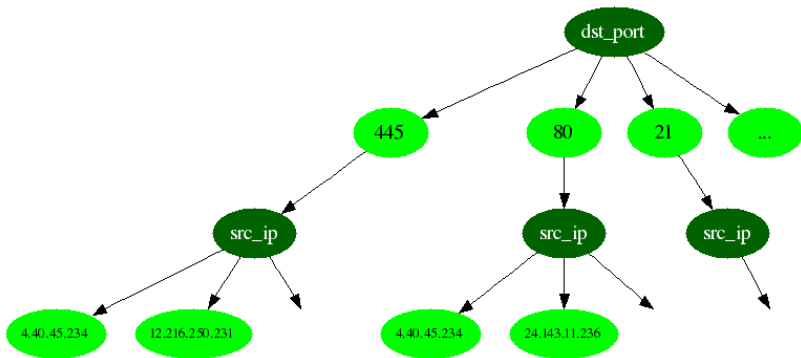
Histograms 3d: Feature pairs, Port scan



Histograms 3d: Feature pairs, Port scan



Snort port scan alerts



Snort port scan alerts

TreeView2 source: Kerf/data/snort2.log

File Edit View Help

▼ [1339, 1] Snort portscan alerts

- ▶ (55/1135) dst_port: 445 src_ip: 55 dst_ip: 75 src_port: 100+
- ▶ (8/70) dst_port: 80 src_ip: 8 dst_ip: 30 src_port: 63
- ▶ (1/26) dst_port: 21 src_ip: (80.141.141.173) dst_ip: 11 src_port: 11
- ▶ (1/22) dst_port: 4899 src_ip: (218.103.195.242) dst_ip: 22 src_port: 22
- ▶ (2/20) dst_port: 4000 src_ip: 2 dst_ip: 8 src_port: 15
- ▶ (1/15) dst_port: 443 src_ip: (211.5.239.5) dst_ip: 9 src_port: 9
- ▶ (1/15) dst_port: 139 src_ip: (129.170.125.243) dst_ip: 8 src_port: 8
- ▶ (1/12) **dst_port: 1524 src_ip: (192.139.15.34) dst_ip: 12 src_port: (1524)**
- ▶ (1/9) dst_port: 1 src_ip: (209.15.84.72) dst_ip: 9 src_port: 9
- ▶ (1/3) dst_port: 8100 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ (1/3) dst_port: 8000 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ (1/3) dst_port: 8080 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ (1/3) dst_port: 3128 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ (1/3) dst_port: 1080 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2

autosplit via minentdep3 without mark ▼

autosplit via minentdep3 without mark: -- OK

Field	#	Value
%dst_port %dst_ip	194	1 129.170.125.243
%dst_port %src_ip	76	1 209.15.84.72
%dst_port %src_port	687	1 1551, 1524
dst_ip	75	129.170.125.243
dst_port	14	1, 21, 80, 1080, 1524, 4000, 443, 4899
flags		*****S
loghost		annon
program		snort
repeat		
rule_id		732c5ec
serial		-1
src_ip	71	4.40.45.1
src_port	668	1027, 1524
type		SYN

Compute ranges

Split

Snort port scan alerts

TreeView2 source: Kerf/data/snort2.log

File Edit View Help

▼ (1339, 2) Snort portscan alerts

- ▶ [1135] dst_port: 445 src_ip: 55 dst_ip: 75 src_port: 100+
- ▶ [70, 1] dst_port: 80 src_ip: 8 dst_ip: 30 src_port: 63
- ▶ [26] dst_port: 21 src_ip: (80.141.141.173) dst_ip: 11 src_port: 11
- ▶ [22] dst_port: 4899 src_ip: (218.103.195.242) dst_ip: 22 src_port: 22
- ▶ [20] dst_port: 4000 src_ip: 2 dst_ip: 8 src_port: 15
- ▶ [15] dst_port: 443 src_ip: (211.5.239.5) dst_ip: 9 src_port: 9
- ▶ [15] dst_port: 139 src_ip: (129.170.125.243) dst_ip: 8 src_port: 8
- ▶ [12] **dst_port: 1524 src_ip: (192.139.15.34) dst_ip: 12 src_port: (1524)**
- ▶ [9] dst_port: 1 src_ip: (209.15.84.72) dst_ip: 9 src_port: 9
- ▶ [3, 2] dst_port: 8100 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ [3, 2] dst_port: 8000 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ [3, 2] dst_port: 8080 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ [3, 2] dst_port: 3128 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ [3, 2] dst_port: 1080 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2

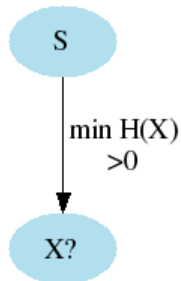
Field	#	Value
_id	100+	e5b80313:
_month		Apr
_program		snort
_rule_id		732c5ed3:
_timestan	100+	Fri 11-Apr-
_year		2003
dst_ip	75	129.170.10
dst_port	14	1, 21, 80,
flags		*****S*
loghost		annon
mark		pos
program		snort
repeat		

Outline

- 1 Log browsing moves
 - Pipes and tables
 - Trees are better than pipes and tables!
- 2 Data organization
 - Trying to define the browsing problem
 - Entropy
 - Measuring co-dependence
 - Mutual Information
 - The tree building algorithm
- 3 Examples

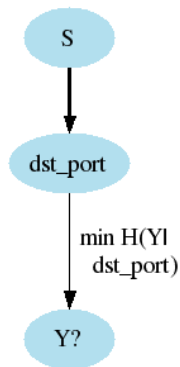
Building a data view

- 1 Pick the feature with lowest non-zero entropy (“simplest histogram”)
- 2 Split all records on its distinct values
- 3 Order other features by the strength of their dependence with with the first feature (conditional entropy or mutual information)
- 4 Use this order to label groups
- 5 Repeat with next feature in (1)



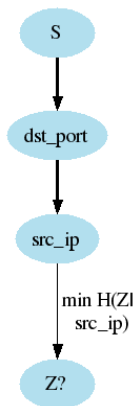
Building a data view

- 1 Pick the feature with lowest non-zero entropy ("simplest histogram")
- 2 Split all records on its distinct values
- 3 Order other features by the strength of their dependence with with the first feature (conditional entropy or mutual information)
- 4 Use this order to label groups
- 5 Repeat with next feature in (1)



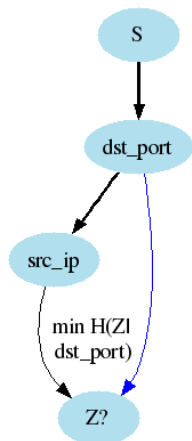
Building a data view

- 1 Pick the feature with lowest non-zero entropy (“simplest histogram”)
- 2 Split all records on its distinct values
- 3 Order other features by the strength of their dependence with with the first feature (conditional entropy or mutual information)
- 4 Use this order to label groups
- 5 Repeat with next feature in (1)



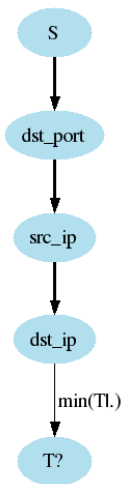
Building a data view

- 1 Pick the feature with lowest non-zero entropy ("simplest histogram")
- 2 Split all records on its distinct values
- 3 Order other features by the strength of their dependence with with the first feature (conditional entropy or mutual information)
- 4 Use this order to label groups
- 5 Repeat with next feature in (1)

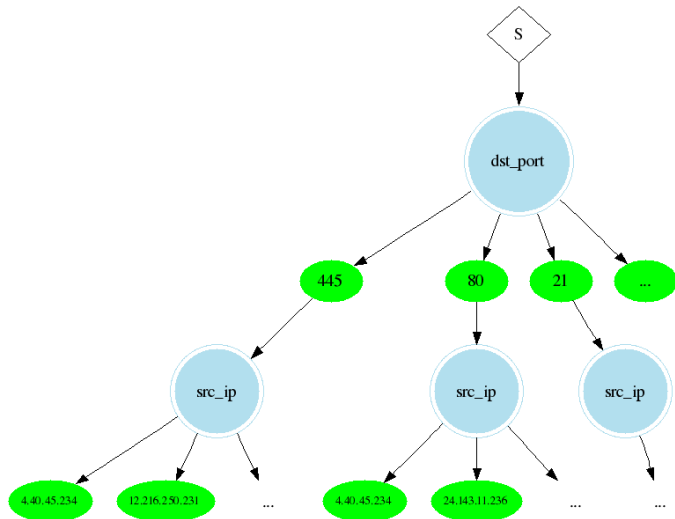


Building a data view

- 1 Pick the feature with lowest non-zero entropy (“simplest histogram”)
- 2 Split all records on its distinct values
- 3 Order other features by the strength of their dependence with with the first feature (conditional entropy or mutual information)
- 4 Use this order to label groups
- 5 Repeat with next feature in (1)



Snort port scan alerts



Snort port scan alerts

TreeView2 source: Kerf/data/snort2.log

File Edit View Help

▼ [1339, 1] Snort portscan alerts

- ▶ (55/1135) dst_port: 445 src_ip: 55 dst_ip: 75 src_port: 100+
- ▶ (8/70) dst_port: 80 src_ip: 8 dst_ip: 30 src_port: 63
- ▶ (1/26) dst_port: 21 src_ip: (80.141.141.173) dst_ip: 11 src_port: 11
- ▶ (1/22) dst_port: 4899 src_ip: (218.103.195.242) dst_ip: 22 src_port: 22
- ▶ (2/20) dst_port: 4000 src_ip: 2 dst_ip: 8 src_port: 15
- ▶ (1/15) dst_port: 443 src_ip: (211.5.239.5) dst_ip: 9 src_port: 9
- ▶ (1/15) dst_port: 139 src_ip: (129.170.125.243) dst_ip: 8 src_port: 8
- ▶ (1/12) **dst_port: 1524 src_ip: (192.139.15.34) dst_ip: 12 src_port: (1524)**
- ▶ (1/9) dst_port: 1 src_ip: (209.15.84.72) dst_ip: 9 src_port: 9
- ▶ (1/3) dst_port: 8100 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ (1/3) dst_port: 8000 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ (1/3) dst_port: 8080 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ (1/3) dst_port: 3128 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ (1/3) dst_port: 1080 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2

autosplit via minentdep3 without mark

autosplit via minentdep3 without mark: -- OK

Field	#	Value
%dst_port %dst_ip	194	1 129.170.125.243
%dst_port %src_ip	76	1 209.15.84.72
%dst_port %src_port	687	1 1551, 1524
dst_ip	75	129.170.125.243
dst_port	14	1, 21, 80, 1080, 1524, 4000, 443, 4899
flags		*****S
loghost		annon
program		snort
repeat		
rule_id		732c5ec
serial		-1
src_ip	71	4.40.45.1
src_port	668	1027, 1524
type		SYN

Compute ranges

Split

Snort port scan alerts

TreeView2 source: Kerf/data/snort2.log

File Edit View Help

▼ (1339, 2) Snort portscan alerts

- ▶ [1135] dst_port: 445 src_ip: 55 dst_ip: 75 src_port: 100+
- ▶ [70, 1] dst_port: 80 src_ip: 8 dst_ip: 30 src_port: 63
- ▶ [26] dst_port: 21 src_ip: (80.141.141.173) dst_ip: 11 src_port: 11
- ▶ [22] dst_port: 4899 src_ip: (218.103.195.242) dst_ip: 22 src_port: 22
- ▶ [20] dst_port: 4000 src_ip: 2 dst_ip: 8 src_port: 15
- ▶ [15] dst_port: 443 src_ip: (211.5.239.5) dst_ip: 9 src_port: 9
- ▶ [15] dst_port: 139 src_ip: (129.170.125.243) dst_ip: 8 src_port: 8
- ▶ [12] **dst_port: 1524 src_ip: (192.139.15.34) dst_ip: 12 src_port: (1524)**
- ▶ [9] dst_port: 1 src_ip: (209.15.84.72) dst_ip: 9 src_port: 9
- ▶ [3, 2] dst_port: 8100 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ [3, 2] dst_port: 8000 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ [3, 2] dst_port: 8080 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ [3, 2] dst_port: 3128 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2
- ▶ [3, 2] dst_port: 1080 src_ip: (194.208.40.120) dst_ip: 2 src_port: 2

Field	#	Value
_id	100+	e5b80313:
_month		Apr
_program		snort
_rule_id		732c5ed3:
_timestan	100+	Fri 11-Apr-
_year		2003
dst_ip	75	129.170.10
dst_port	14	1, 21, 80,
flags		*****S*
loghost		annon
mark		pos
program		snort
repeat		

Quick pair summary

TreeView2 source: Kerf/data/ssh-auth-2users

File Edit View Help

- [617] 617 logins from mediaone.net
 - [606] host: h000502032ae9.ne.mediaone.net user: 7 tty: 5
 - [589] user: josh tty: 4
 - [8] user: jos tty: (
 - [3] user: tty: (
 - [3] user: johs tty: (
 - [1] user: (null) tty: (
 - [1] user: r] tty: (
 - [1] user: josh^[[D tty: (
 - [10] host: we-24-31-59-152.we.mediaone.net user: (oleg) tty: 2
 - [1] host: h0010b565bb03.ne.mediaone.net user: (josh) tty: (tty0)

Feature	E	#	Entropy
H(%host %user)	cH	0.169/1.18	
H(%user %host)	cH	0.000/1.00	

One ISP, 617 lines, 2 users, one tends to mistype.
11 lines of screen space.

Quick pair summary

TreeView2 source: Kerf/data/ssh-auth-2users

File Edit View Help

617 logins from mediaone.net

- 606 host: h000502032ae9.ne.mediaone.net user: 7 tty: 5
 - 589 user: josh tty: 4
 - 8 user: jos tty: ()
 - 3 user: tty: ()
 - 3 user: josh tty: ()
 - 11 user: (null) tty: ()
 - 11 user: r] tty: ()
 - 11 user: josh^[[D tty: ()
 - 10 host: we-24-31-59-152.we.mediaone.net user: (oleg) tty: 2
 - 1 host: h0010b565bb03.ne.mediaone.net user: (josh) tty: (tty0)
 - 1 user: josh tty: (tty0)

Jan 10 00:04:14 mystic syslog: LOGIN ON tty0 BY josh FROM h0010b

Feature	E	#	Entropy
H(%host%user)	cH	0.169/1.18	
H(%user%host)	cH	0.000/1.00	

One ISP, 617 lines, 2 users, one tends to mistype.
11 lines of screen space.

Novelty changes the order

TreeView2 source: Kerf/data/snort2b.log

File Edit View Help

[17299] Snort portscan alerts

- [6629] dst_port: 445 dst_ip: 95 src_ip: 100+ src_port: 100+
- [5428] dst_port: 139 dst_ip: 94 src_ip: 100+ src_port: 100+
- [1743] dst_port: 80 dst_ip: 94 src_ip: 100+ src_port: 100+
- [894] dst_port: 135 dst_ip: 85 src_ip: 37 src_port: 100+
- [518] dst_port: 1433 dst_ip: 91 src_ip: 33 src_port: 100+
- [451] dst_port: 17300 dst_ip: 86 src_ip: 34 src_port: 100+
- [397] dst_port: 21 dst_ip: 81 src_ip: 23 src_port: 100+
- [266] dst_port: 6667 dst_ip: 75 src_ip: 18 src_port: 100+
- [178] dst_port: 443 dst_ip: 68 src_ip: 13 src_port: 100+

Fields	Slate	Features	Top	Index
Feature	E	#		
%src_port		4876/17299/8.25		
%src_ip		689/17299/6.169		
%dst_ip		95/17299/4.361/7		
%dst_port		35/17299/1.783/5		
%flags		2/17299/0.004/1.		
%type		2/17299/0.004/1.		
%loghost		1/17299/0.000/1.		
%serial		1/17299/0.000/1.		

TreeView2 source: Kerf/data/snort2b.log

File Edit View Help

[17299,] Snort portscan alerts

- [17292] flags: *****S* type: (SYN) dst_port: 35 dst_ip: 95 src_ip: 100+
- [7] flags: *****SF type: (SYNFIN) dst_port: (21) dst_ip: 7 src_ip: (142.26.217.6)

Fields	Slate	Features	Top	Index
Feature	E	#		
%src_port		4876/17299/8.25		
%src_ip		689/17299/6.169		
%dst_ip		95/17299/4.361/7		

Looking at Root-Fu captures

TreeView2 source: xml-packet-summary.xml

File Edit View Help

[60943] do not expand me

- ▶ [18/48788] Protocol: TCP Source: 18 Destination: 22 Info: 100+
- ▶ [13/4378] Protocol: HTTP Source: 13 Destination: 13 Info: 100+
- ▶ [11/3435] Protocol: IRC Source: 11 Destination: 11 Info: 6
- ▶ [11/2325] Protocol: SSHv2 Source: 11 Destination: 11 Info: 84
- ▶ [11/1383] Protocol: ICMP Source: 11 Destination: 13 Info: 2
- ▶ [3/521] Protocol: MySQL Source: 3 Destination: 2 Info: 100+
- ▶ [1/96] Protocol: DNS Source: (192.168.101.2) Destination: 3 Info: 12
- ▶ [5/12] Protocol: SSH Source: 5 Destination: 4 Info: 7
- ▶ [2/2] Protocol: DCERPC Source: 2 Destination: 2 Info: 2
- ▶ [1/2] Protocol: Syslog Source: (192.168.1.40) Destination: (192.168.2.1)
- ▶ [1/1] Protocol: ISystemActivator Source: (192.168.1.50) Destination: (:

Feature	E	#	Entropy
%Info		49088/60943	9.744/17050.046
%Destination		23/60943	2.254/9.525
%Source		19/60943	2.107/8.225
%Protocol		11/60943	0.793/2.211

Load from file

Collect from node

Compute entropy

Looking at Root-Fu captures

TreeView2 source: xml-packet-summary.xml

File Edit View Help

▶ [11/1383]	Protocol: ICMP	Source: 11	Destination: 13	Info: 2
▶ [3/521]	Protocol: MySQL	Source: 3	Destination: 2	Info: 100+
▼ [1/96]	Protocol: DNS	Source: (192.168.101.2)	Destination: 3	Info: 12
▶ [16/16]	Info: Standard query response PTR blue.rootfu.jp			
▶ [13/13]	Info: Standard query response PTR green.rootfu.jp			
▶ [13/13]	Info: Standard query response, No such name			
▶ [13/13]	Info: Standard query response PTR orange.rootfu.jp			
▶ [11/11]	Info: Standard query response A 192.168.4.2			
▶ [6/6]	Info: Standard query response PTR cyan.rootfu.jp			
▶ [5/5]	Info: Standard query response PTR yellow.rootfu.jp			
▶ [5/5]	Info: Standard query response			
▶ [5/5]	Info: Standard query response A 192.168.2.2			
▶ [4/4]	Info: Standard query response A 192.168.7.2			
▶ [3/3]	Info: Standard query response A 192.168.3.2			
▶ [2/2]	Info: Standard query response A 192.168.5.2			
▶ [5/12]	Protocol: SSH	Source: 5	Destination: 4	Info: 7

split by "%Info"

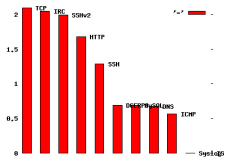
split by "%Info": -- OK

Feature	E	#	Entropy
%Info		12/96	2.315/10.130
%Destination		3/96	0.675/1.963
%Protocol		1/96	0.000/1.000
%Source		1/96	0.000/1.000

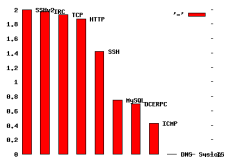
Load from file
Collect from node
Compute entropy

Comparing 2nd order uncertainties

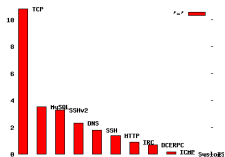
1



2



3



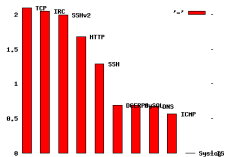
Compare uncertainties in each Protocol group:

- 1 Destination: $H = 2.9999$
- 2 Source: $H = 2.8368$
- 3 Info: $H = 2.4957$

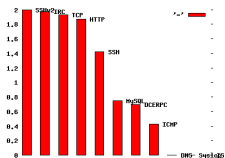
“Start with the simpler view”

Comparing 2nd order uncertainties

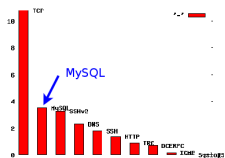
1



2



3



Compare uncertainties in each Protocol group:

- 1 Destination: $H = 2.9999$
- 2 Source: $H = 2.8368$
- 3 Info: $H = 2.4957$

“Start with the simpler view”

Looking at Root-Fu captures

TreeView2 source: xml-packet-summary.xml

File Edit View Help

do not expand me

- ▶ [18/48788] Protocol: TCP Source: 18 Destination: 22 Info: 100+
- ▶ [13/4378] Protocol: HTTP Source: 13 Destination: 13 Info: 100+
- ▶ [11/3435] Protocol: IRC Source: 11 Destination: 11 Info: 6
- ▶ [11/2325] Protocol: SSHv2 Source: 11 Destination: 11 Info: 84
- ▶ [11/1383] Protocol: ICMP Source: 11 Destination: 13 Info: 2
- ▶ [128/521] **Protocol: MySQL Source: 3 Destination: 2 Info: 100+**
- ▶ [1/96] Protocol: DNS Source: (192.168.101.2) Destination: 3 Info: 12
- ▶ [5/12] Protocol: SSH Source: 5 Destination: 4 Info: 7
- ▶ [2/2] Protocol: DCERPC Source: 2 Destination: 2 Info: 2
- ▶ [1/2] Protocol: Syslog Source: (192.168.1.40) Destination: (192.168.2)
- ▶ [1/1] Protocol: SystemActivator Source: (192.168.1.50) Destination:

split by "%Info"

Marked nodes: pos 521 neg 0

Feature	E	#	Entropy
%Info		49088/60943	9.744/17050.0
%Destination		23/60943	2.254/9.525
%Source		19/60943	2.107/8.225
%Protocol		11/60943	0.793/2.211

Load from file

Collect from node

Compute entropy

Looking at Root-Fu captures

TreeView2 source: xml-packet-summary.xml

File Edit View Help

▼ [3/521] Protocol: MySQL Source: 3 Destination: 2 Info: 100+

- ▶ [180/180] Info: Response OK
- ▶ [32/32] Info: Response Error Code: 417
- ▶ [21/21] Info: Server Greeting Protocol : 10 ,version: 4.1.1-alpha-log Caps
- ▶ [14/14] Info: Request Command: Quit
- ▶ [13/13] Info: Login Request Caps: 0x2485 ,user: root ,password: ha-log
- ▶ [9/9] Info: Request Command: Query : show tables
- ▶ [8/8] Info: Request Command: Query : show databases
- ▶ [7/7] Info: Response Error Code: 428
- ▶ [6/6] Info: Server Greeting Protocol : 255[Unreassembled Packet]
- ▶ [6/6] Info: Request Command: Init Database : user
- ▶ [6/6] Info: Response Error Code: 419
- ▶ [5/5] Info: Request Command: Query : select user()
- ▶ [5/5] Info: Response Error Code: 46c
- ▶ [4/4] Info: Request Command: Field List : help_keyword\000
- ▶ [4/4] Info: Request Command: Field List : func\000

split by "%Info"

split by "%Info": -- OK

Feature	E	#	Entropy
%Info		128/521	3.516/33.658
%Source		3/521	0.748/2.113
%Destination		2/521	0.693/2.000
%Protocol		1/521	0.000/1.000

Load from file
Collect from node
Compute entropy

Looking at Root-Fu captures

TreeView2 source: xml-packet-summary.xml

File Edit View Help

Info: Request Command: Field List : host\000
 Info: Request Command: Field List : db\000
 Info: Request Command: Field List : tables_priv\000
 Info: Request Command: Field List : user\000
 Info: Request Command: Field List : columns_priv\000
 Info: Request Command: Field List : help_category\000
 Info: Request Command: Field List : help_relation\000
 Info: [TCP Retransmission] Request Command: Quit
Info: Request Command: Query : set password for 'root'@'green-router.rootfu.jp'=password('schoolofr00t')
Info: Request Command: Query : update user set password=password('schoolofr00t') where user = 'root'@
 Info: Request Command: Query : ALTER TABLE user CHANGE COLUMN Password Password LONGTEXT
 Info: Request Command: Field List : psl_story\000
 Info: Request Command: Field List : psl_section_block_lut\000
 Info: Request Command: Field List : db_sequence\000
 Info: Request Command: Field List : psl_quote\000
 Info: Request Command: Field List : psl_commentcount\000
 Info: Request Command: Field List : CASURDATA\000

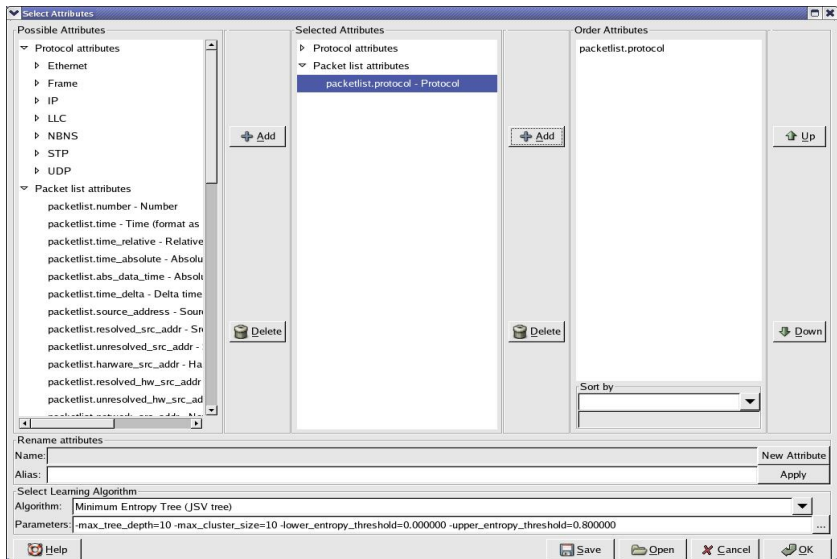
split by "%Info"

Marked nodes: pos 6 neg 0

Feature	E	#
%Info		490
%Destination	23/6	
%Source	19/6	
%Protocol	11/6	

Load from file
 Collect from node
 Compute entropy

Screenshots (1)



Screenshots (2)

No. .	Time	Source	Destination	Protocol	HwAddr
1	0.000000	192.168.2.3	195.138.145.122	UDP	00:11:50:38:81:70
2	0.031164	192.168		UDP	00:11:50:38:81:70
3	0.061035	192.168		UDP	00:11:50:38:81:70
4	0.072645	195.138		UDP	00:0d:60:76:d9:ce
5	0.090327	195.138		UDP	00:0d:60:76:d9:ce
6	0.091400	192.168		UDP	00:11:50:38:81:70
7	0.119496	192.168		UDP	00:11:50:38:81:70
8	0.121801	195.138		UDP	00:0d:60:76:d9:ce
9	0.149736	192.168		UDP	00:11:50:38:81:70
10	0.159995	195.138		UDP	00:0d:60:76:d9:ce
11	0.177547	192.168		UDP	00:11:50:38:81:70
12	0.193140	195.138		UDP	00:0d:60:76:d9:ce
13	0.208144	192.168		UDP	00:11:50:38:81:70
14	0.215444	195.138.145.122	192.168.2.3	UDP	00:0d:60:76:d9:ce

Field name	Ethereal formula	Unique values	Entropy	Values summary
tcp.analysis.zero_window		2	0.015826	#undef, Zero V
tcp.options.sack_le		2	0.028560	#undef, 1188
tcp.analysis.retransmission		2	0.028560	#undef, Retrar

Value	Count	Field 1 name	Field 2 name	Co
#undef	447			
Retransmission 2				

Screenshots (3)

The screenshot shows the 'Min Entropy Tree' interface. On the left, a tree view displays the hierarchy: [18944/9] Min Entropy Tree > [18335/17] packetlist.protocol UDP > [449/2] packetlist.protocol. A context menu is open over the selected node, listing options: Apply template..., Save template..., Compute ranges, Show template, Mark positive, Mark negative, and Clear markings. On the right, the 'Ranges' tab is active, showing a table with the following data:

Field name	Ethereal formula	Unique values	Entropy	Values summary
frame.number		449	6.107023	10046, 10048, 1
ip.dsfield.dscp		1	0.000000	0000 00.. = Def
frame.protocols		2	0.507120	eth:ip,tcp, eth:ip

Below the table, there is a section for 'Value Count' with columns for Field name, Cond. entropy, Field 1 name, Field 2 name, and Count.

The screenshot shows the 'Ranges' tab with a tree view on the left and a 'Change values' dialog box at the bottom. The tree view shows the hierarchy: root > packetlist.protocol > eth.trailer > eth.dst > leaf > leaf > tcp.ack. The 'Change values' dialog box has the following fields:

- Label: "packetlist.protocol" %packetlist.protocol
- Sort key: @num_leaves
- Sort order: desc

An 'Apply' button is located at the bottom right of the dialog box.

Research links

Research on using entropy and related measures for network anomaly detection:

- Information-Theoretic Measures for Anomaly Detection, Wenke Lee & Dong Xiang, 2001
- Characterization of network-wide anomalies in traffic flows, Anukool Lakhina, Mark Crovella & Christiphe Diot, 2004
- Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation, Yu Gu, Andrew McCallum & Don Towsley, 2005
- ...

Summary

Information theory provides useful heuristics for:

- summarizing log data in medium size batches,
- choosing data views that show off interesting features of a particular batch,
- finding good starting points for analysis.

Helpful even with simplest data organization tricks.

In one sentence

$H(X)$, $H(X|Y)$, $I(X; Y)$, ... : parts of a complete analysis kit!

Summary

Information theory provides useful heuristics for:

- summarizing log data in medium size batches,
- choosing data views that show off interesting features of a particular batch,
- finding good starting points for analysis.

Helpful even with simplest data organization tricks.

In one sentence

$H(X)$, $H(X|Y)$, $I(X; Y)$, ... : parts of a complete analysis kit!

Credits & source code

Credits

<i>Kerf project:</i>	Javed Aslam, David Kotz, Daniela Rus, Ron Peterson
<i>Coding:</i>	Cory Cornelius, Stefan Savev
<i>Data & discussions:</i>	George Bakos, Greg Conti, Jason Spence, and many others.
<i>Sponsors:</i>	see website

Code

For source code (GPL), documentation, and technical reports:

<http://kerf.cs.dartmouth.edu>

Thanks!