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?
- Ethereal 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

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
☐ 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
☐ 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
☐ 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
☐ 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
☐ 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
☐ 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
☐ 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
☐ 17	2006-01-14	12:39:57	zion	sshd[5306]	Illegal user asmith from 68.38.148.149	5306	68.38.148.149	asmith
☐ 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
☐ 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)

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

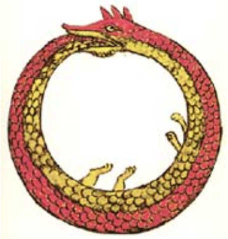
sshd:

    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

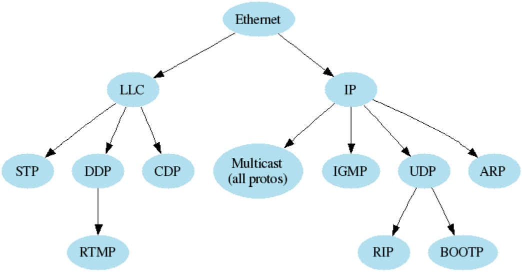
1. Log browsing moves
   - Pipes and tables
   - Trees are better than pipes and tables!
2. Data organization
   - Trying to define the browsing problem
   - Entropy
   - Measuring co-dependence
   - Mutual Information
   - The tree building algorithm
3. Examples

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)

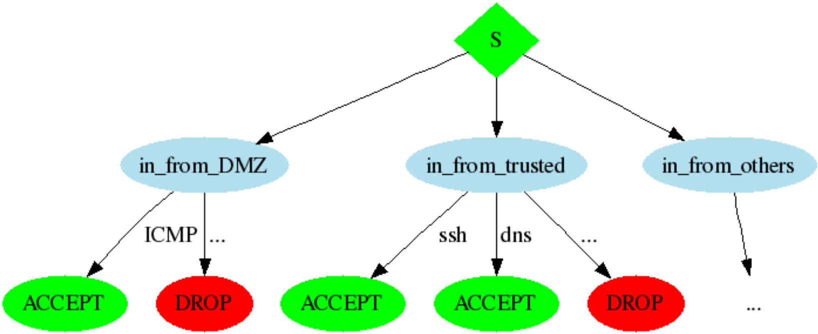
Ethernet	100.00%
Logical-Link Control	70.83%
Spanning Tree Protocol	50.00%
Datagram Delivery Protocol	18.75%
Routing Table Maintenance Protocol	18.75%
Cisco Discovery Protocol	2.08%
Internet Protocol	12.50%
Protocol Independent Multicast	4.17%
Internet Group Management Protocol	2.08%
User Datagram Protocol	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:

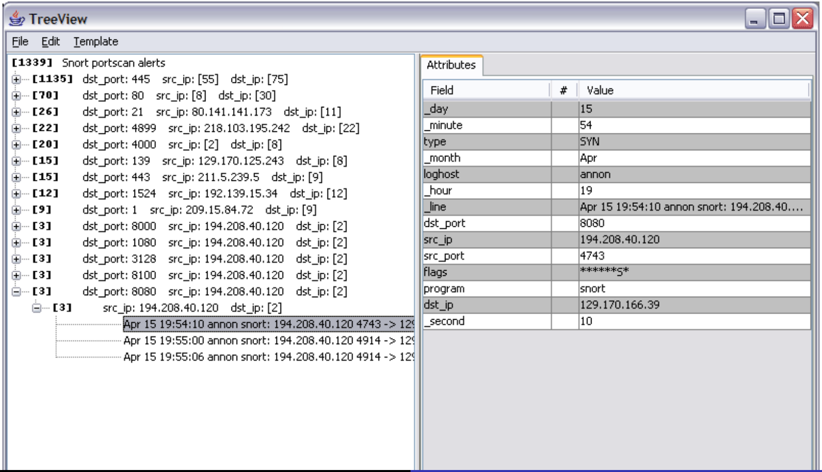
- Protocol hierarchies (e.g., Wireshark)
- Firewall decision trees (e.g., iptables chains)



Use tree views to show logs!

Pipes, SQL queries → branches / paths

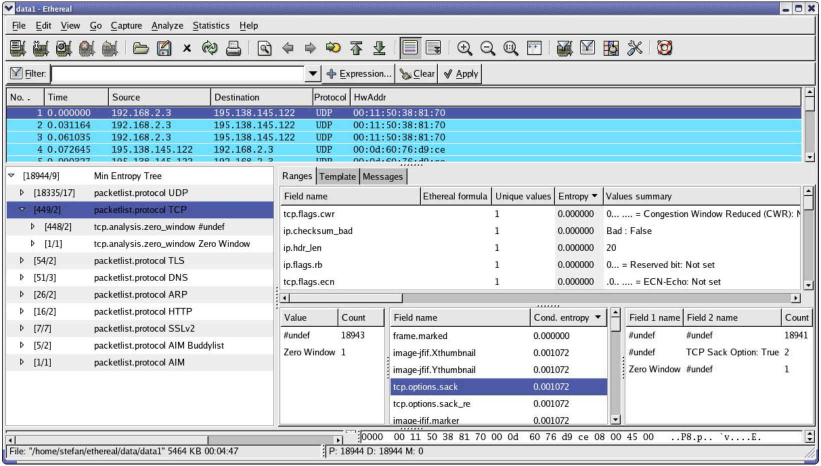
Groups ↔ nodes (sorted by count / weight), records ↔ leaves.



Use tree views to show logs!

Pipes, SQL queries → branches / paths

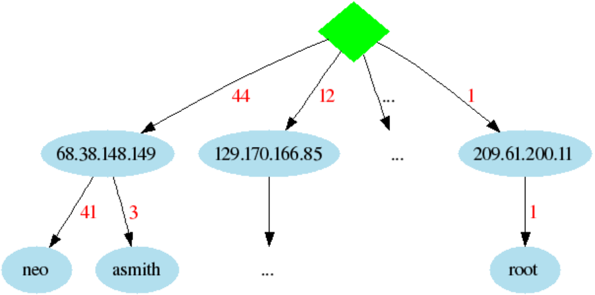
Groups ↔ nodes (sorted by count / weight), records ↔ leaves.



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.



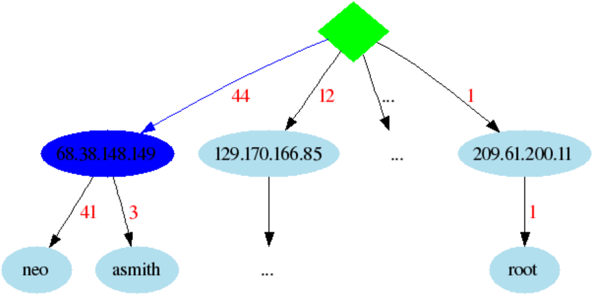
cnt	ip
44	68.38.148.149
12	129.10.75.101
2	129.170.166.85
1	66.183.80.107
1	209.61.200.11

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.



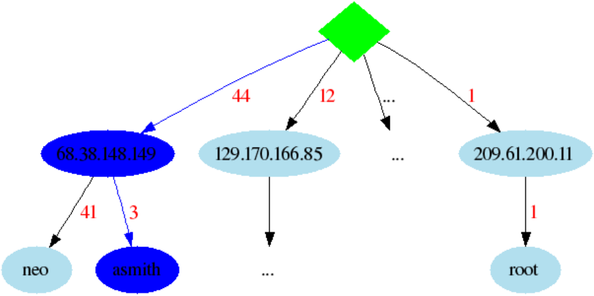
cnt	ip
44	68.38.148.149
12	129.10.75.101
2	129.170.166.85
1	66.183.80.107
1	209.61.200.11

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

Use tree views to show logs!

Pipes, SQL queries → branches / paths

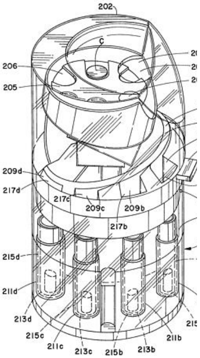
Groups ↔ nodes (sorted by count / weight), records ↔ leaves.
Queries pick out a leaf or a node in the tree.

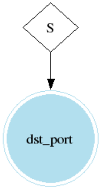


cnt	ip
☐ 44	68.38.148.149
☐ 12	129.10.75.101
☐ 2	129.170.166.85
☐ 1	66.183.80.107
☐ 1	209.61.200.11

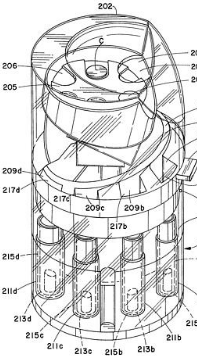
grep 68.38.148.149 /var/log/secure | **grep** asmith | **grep ...**

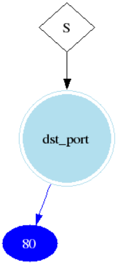
A “coin sorter” for records/packets



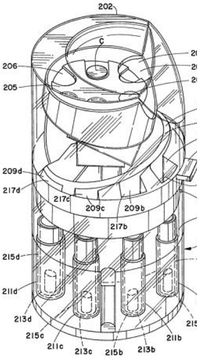


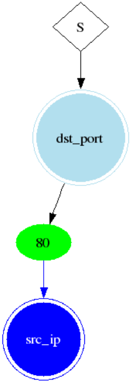
A “coin sorter” for records/packets



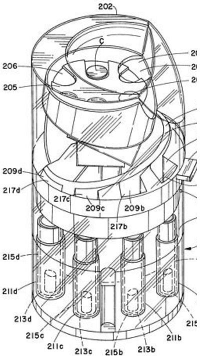


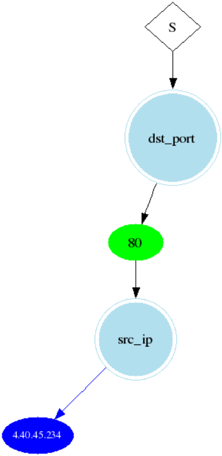
A “coin sorter” for records/packets



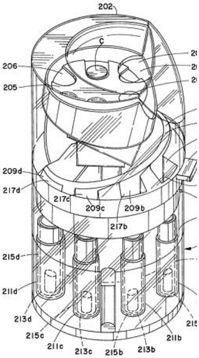


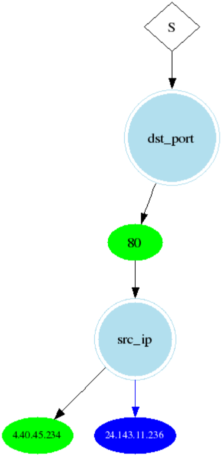
A “coin sorter” for records/packets



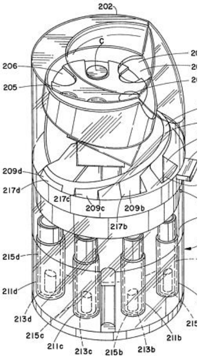


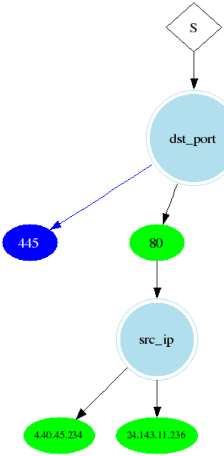
A “coin sorter” for records/packets



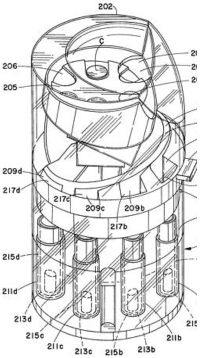


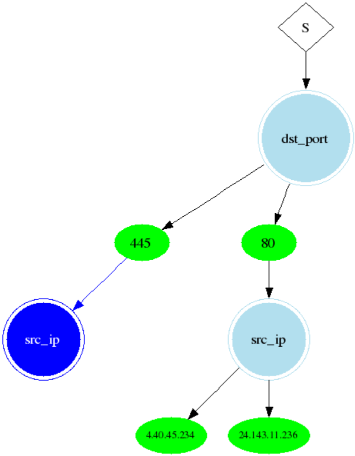
A “coin sorter” for records/packets



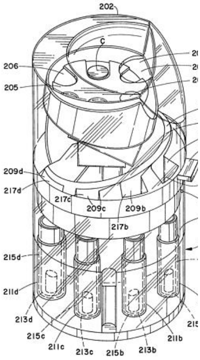


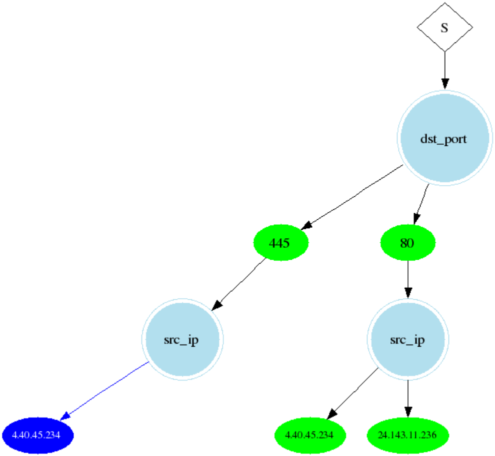
A “coin sorter” for records/packets



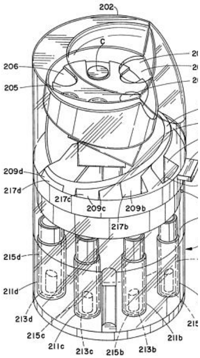


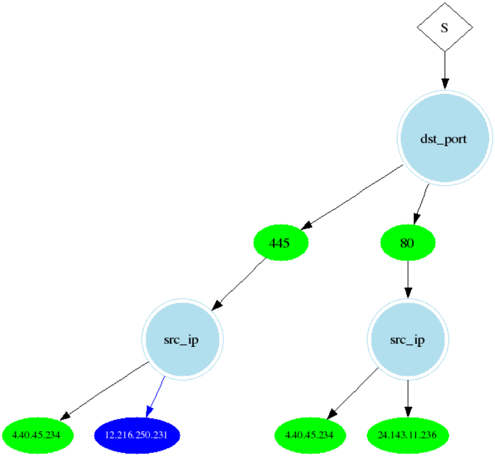
A “coin sorter” for records/packets



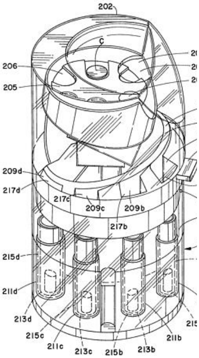


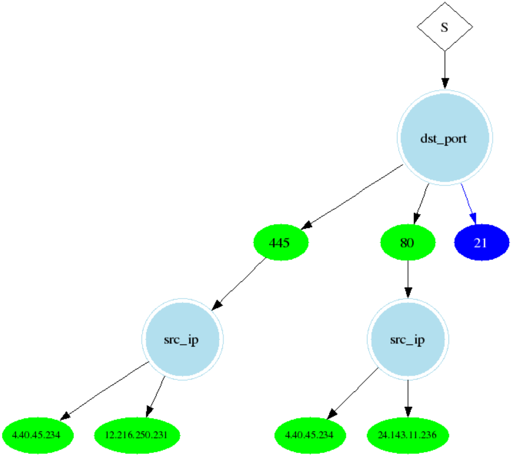
A “coin sorter” for records/packets



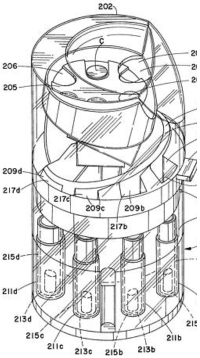


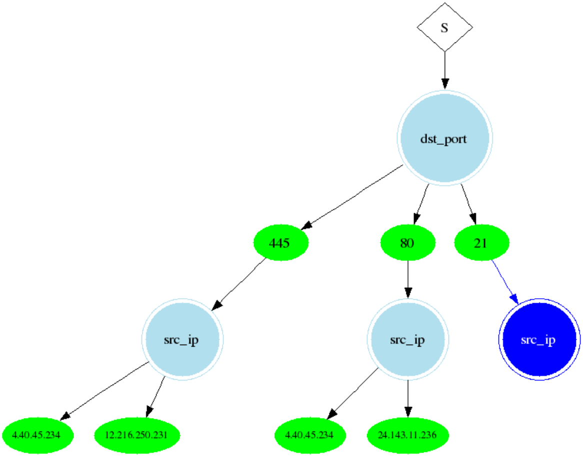
A “coin sorter” for records/packets



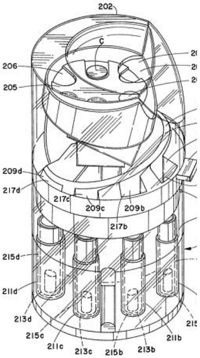


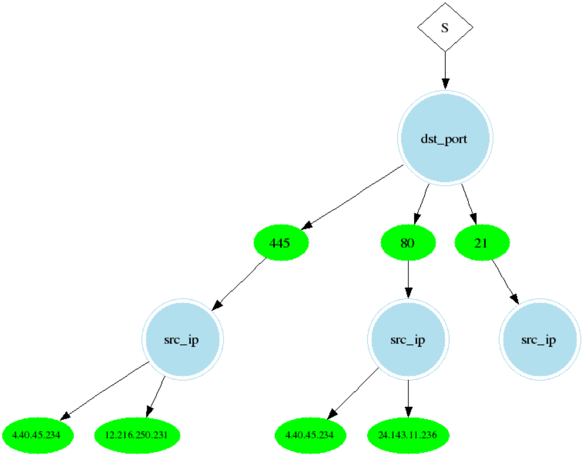
A “coin sorter” for records/packets



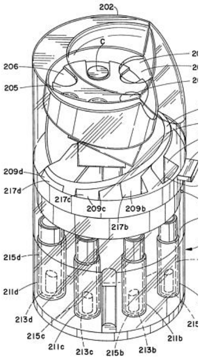


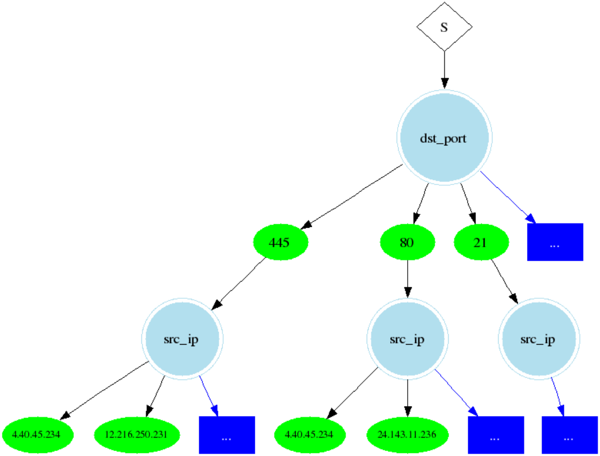
A “coin sorter” for records/packets



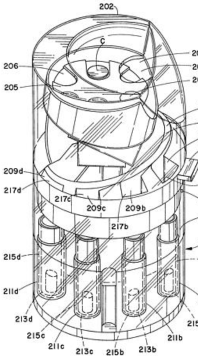


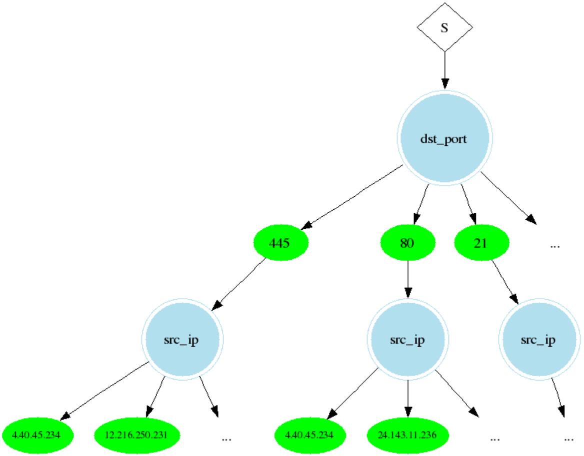
A “coin sorter” for records/packets



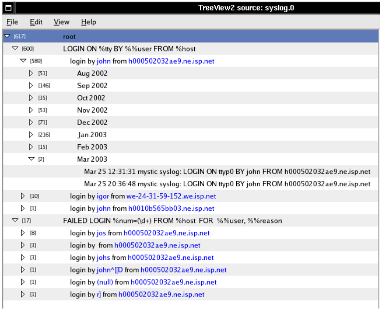


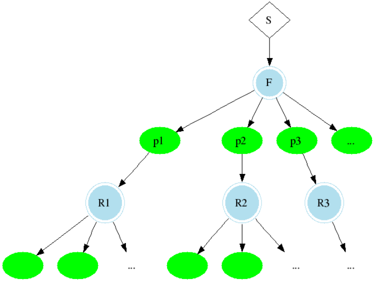
A “coin sorter” for records/packets

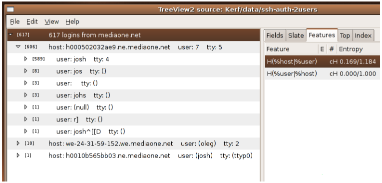




Classify → Save → Apply



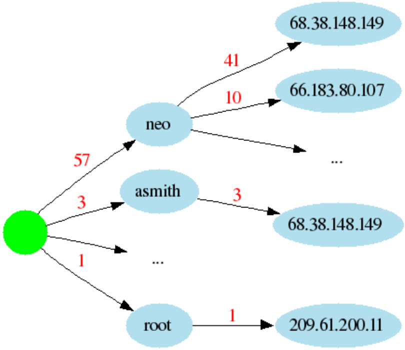




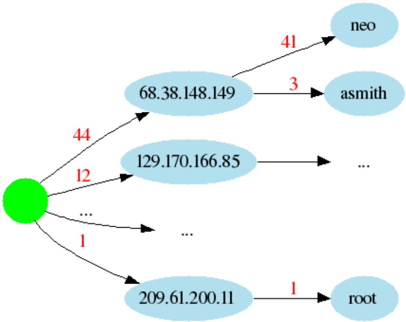
1. Build a classification tree from a dataset
2. Save template
3. Reuse on another dataset

Which tree to choose?

user → ip?



ip → user?



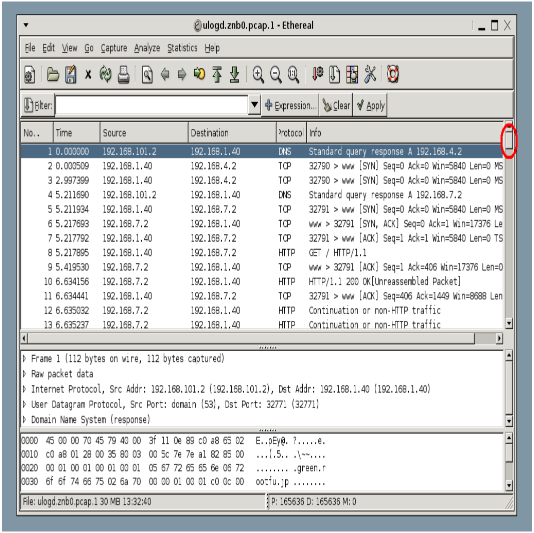
Goal: best grouping

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

Outline

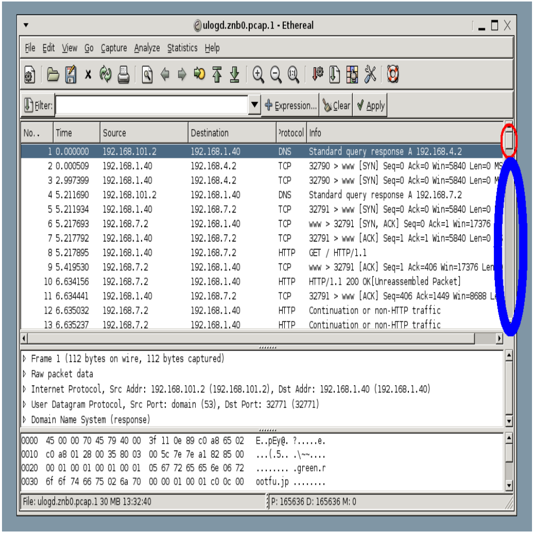
1. Log browsing moves
   - Pipes and tables
   - Trees are better than pipes and tables!
2. Data organization
   - Trying to define the browsing problem
   - Entropy
   - Measuring co-dependence
   - Mutual Information
   - The tree building algorithm
3. Examples

Trying to define the browsing problem

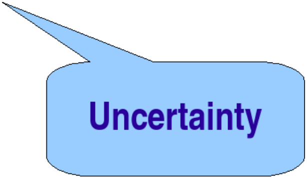


- The lines you need are only 20 **PgDn**s away:
- ...each one surrounded by a page of chaff...
- ...in a twisty maze of messages, all alike...
- ...but slightly different, in ways you don't expect.

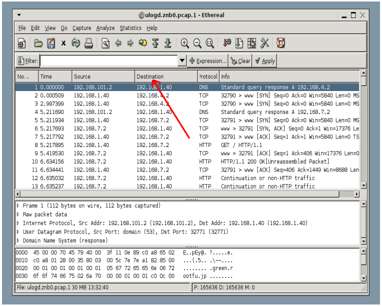
Trying to define the browsing problem



- The lines you need are only 20 **PgDn**s 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.



Old tricks



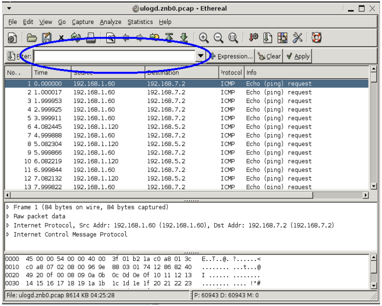
Sorting, grouping & filtering:

- 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



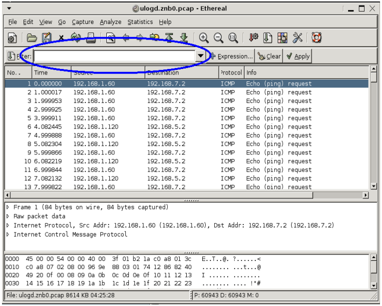
Sorting, grouping & filtering:

- 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)?

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

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAAA...

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

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAAA...

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

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAAA...

The problem:

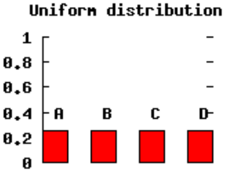
Must deal with uncertainty vs. redundancy of the data.

Measure it!

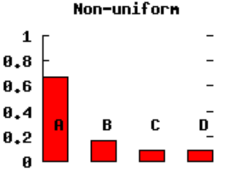
There is a **measure** of uncertainty/redundancy: entropy.

Entropy intuitions

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



ABBBACDBAADBBDCAAACBCDACBBADAD CBBBAABADA... ⇒ 2 bits/symbol



BAADBAAAAAABAAACABBABAAAAAAAAA BAAAADBAAC... ⇒ 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 ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD ZEWRTZYNSADXESYJRQY 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 TUCOOWE 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 $\longrightarrow$ less uncertainty $\longrightarrow$ 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 $\longrightarrow$ less uncertainty $\longrightarrow$ smaller entropy.

Automating old tricks (1)

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

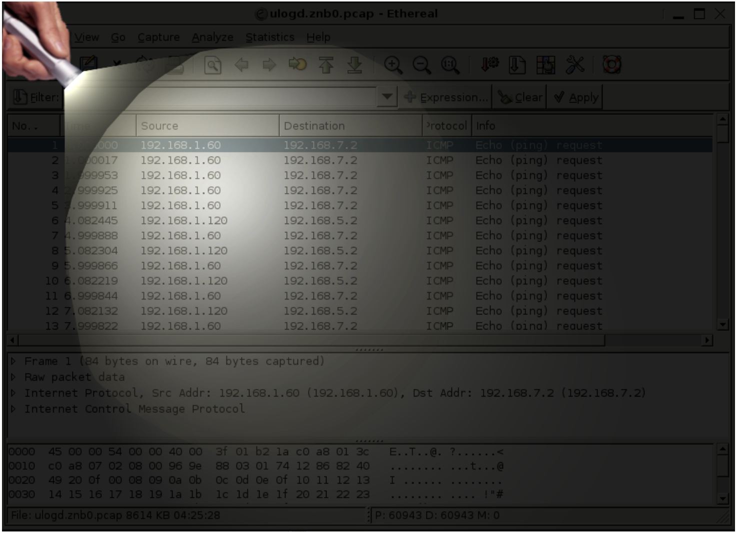
- What if there are many columns and batches of data?
- Which column to start with? How to rank them?

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 $\longrightarrow$ less uncertainty $\longrightarrow$ smaller entropy.

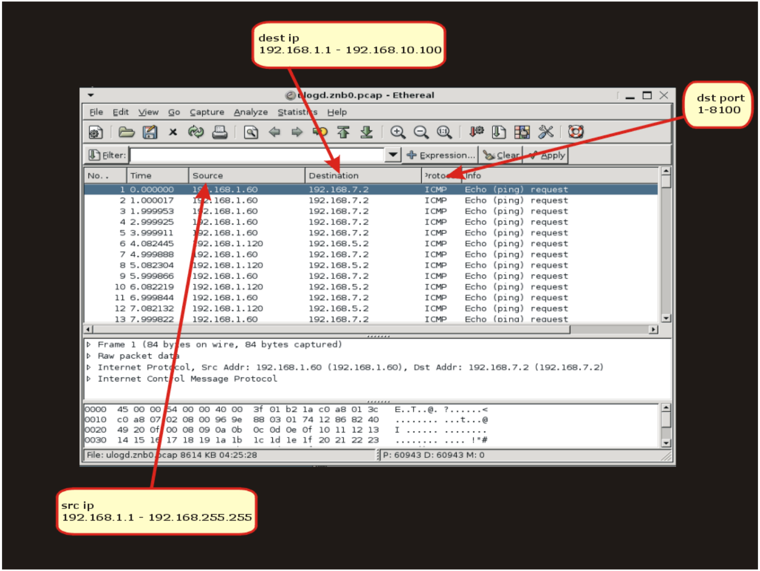
Trivial observations, visualized



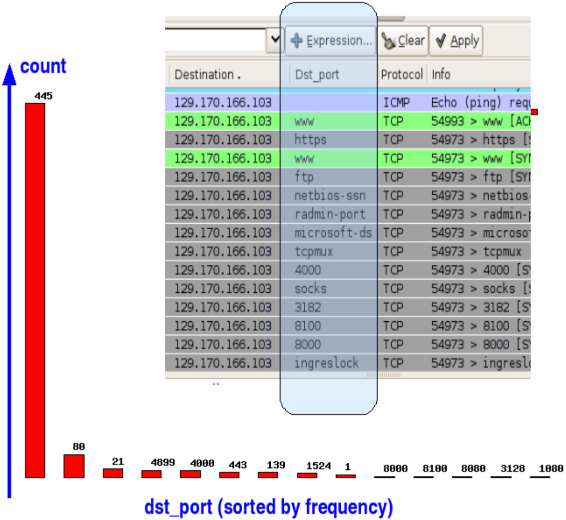
Outline

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

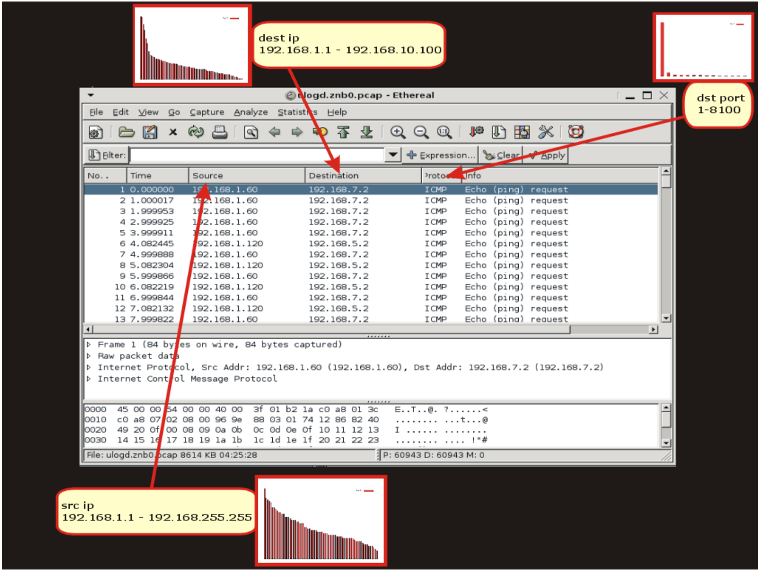
Start simple: Ranges



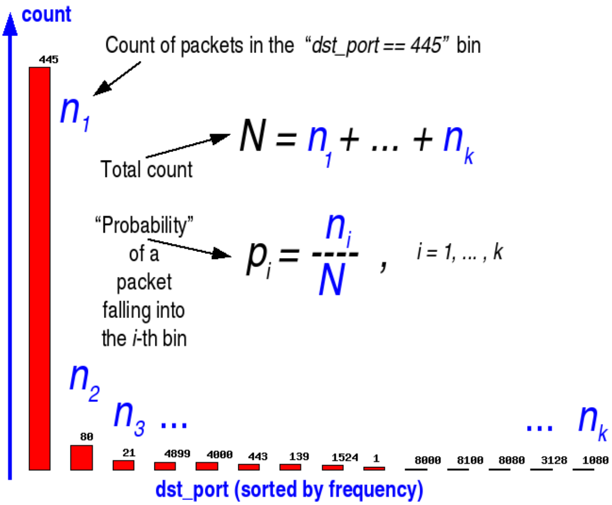
A frequency histogram



Start simple: Histograms



Probability distribution



Definition of entropy

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

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 $i = 1, \ldots, k$.

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

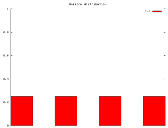
- You are to receive one of N objects, equally likely to be chosen.
- What is the measure of your uncertainty?

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

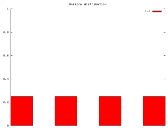
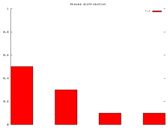
1 

Interpretation

Entropy is a measure of uncertainty about the value of X

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

Entropy on a histogram

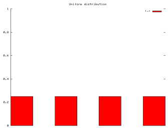
1. 
2. 

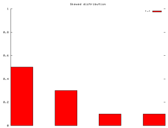
Interpretation

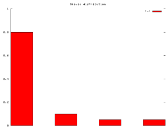
Entropy is a measure of uncertainty about the value of X

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

Entropy on a histogram

1 

2 

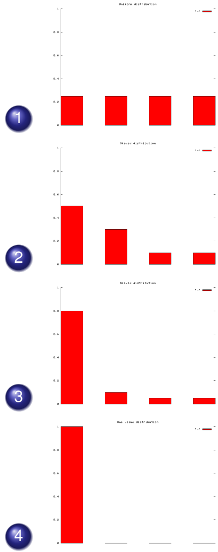
3 

Interpretation

Entropy is a measure of uncertainty about the value of X

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

Entropy on a histogram

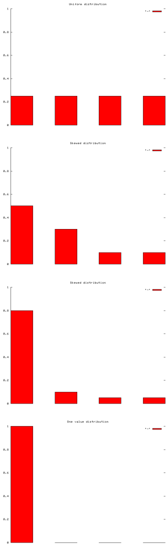


Interpretation

Entropy is a measure of uncertainty about the value of X

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

Entropy on a histogram



1 2 3 4

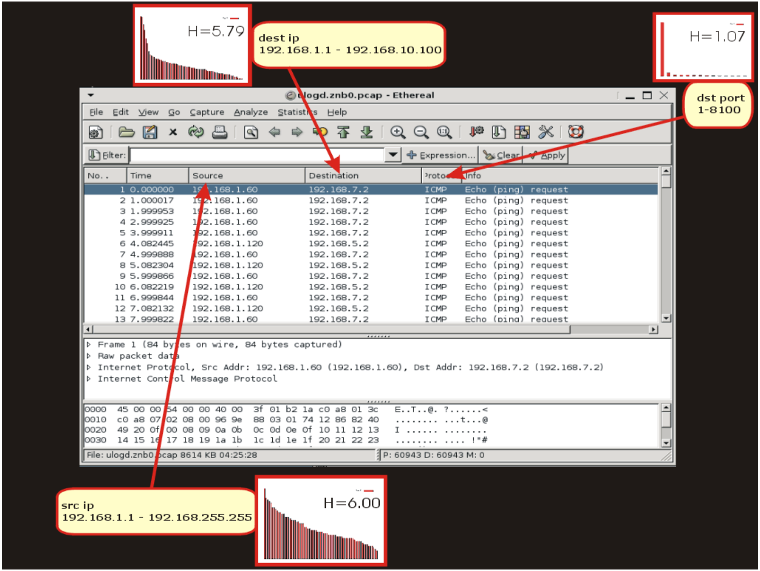
Interpretation

Entropy is a measure of uncertainty about the value of X

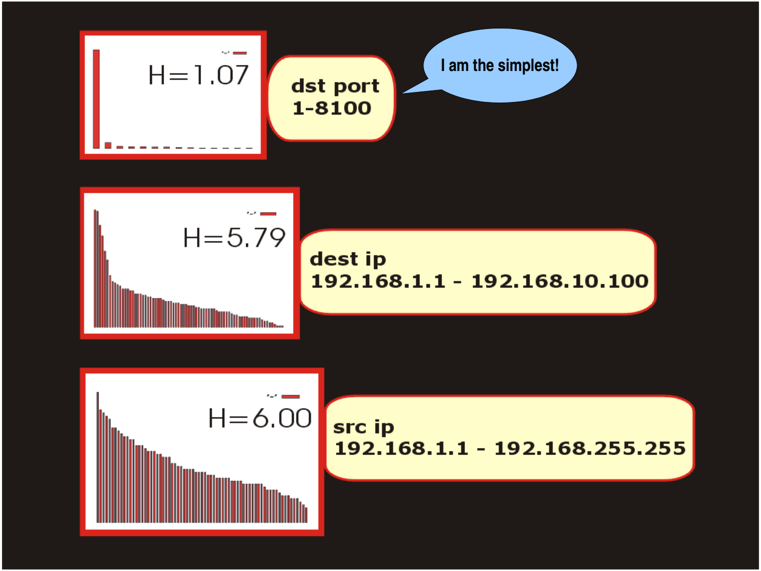
1. $X = (.25 \quad .25 \quad .25 \quad .25) : H(X) = 2$ (bits)
2. $X = (.5 \quad .3 \quad .1 \quad .1) : \quad H(X) = 1.685$
3. $X = (.8 \quad .1 \quad .05 \quad .05) : H(X) = 1.022$
4. $X = (1 \quad 0 \quad 0 \quad 0) : \quad H(X) = 0$

For only one value, the entropy is 0.
When all N values have the same frequency, the entropy is maximal, $\log_2 N$.

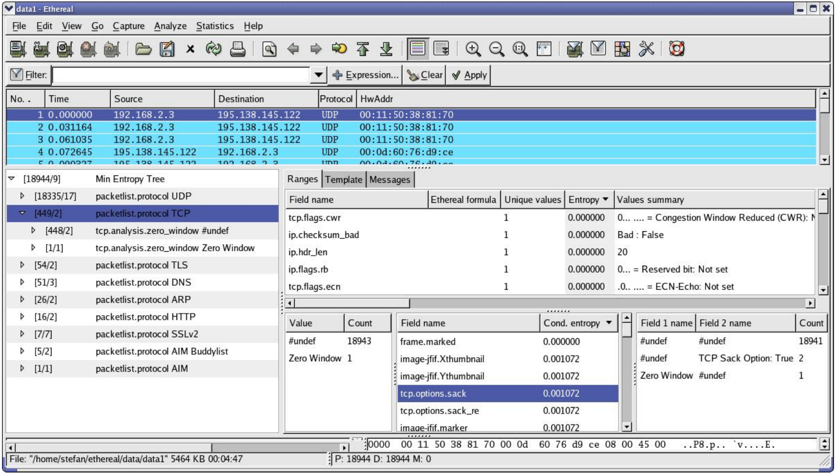
Compare histograms



Start with the simplest



A tree grows in Ethereal



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
```

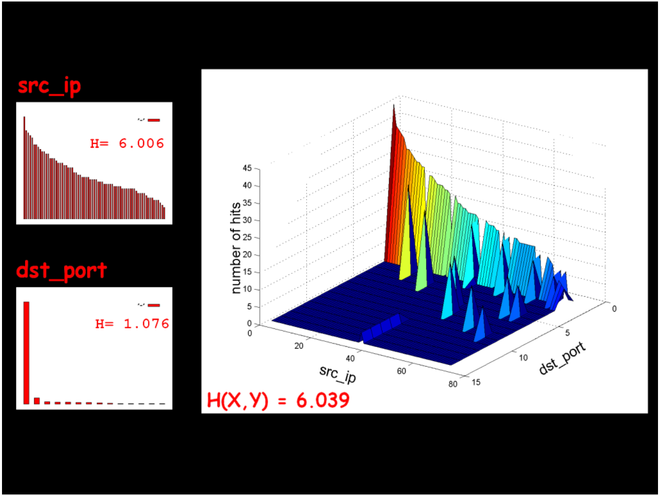
- 2 rooms had by far the largest number of objects picked up.
- Major source of money in the game was: robbers!
  - Stationary camp, safe area, close to cities, easy kill...

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



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}	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$

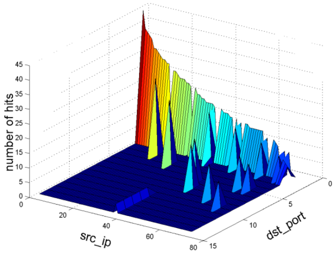
$$p(x_i, y_j) = \frac{n_{ij}}{N}, \quad (N = \sum_{i,j} n_{ij})$$

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.



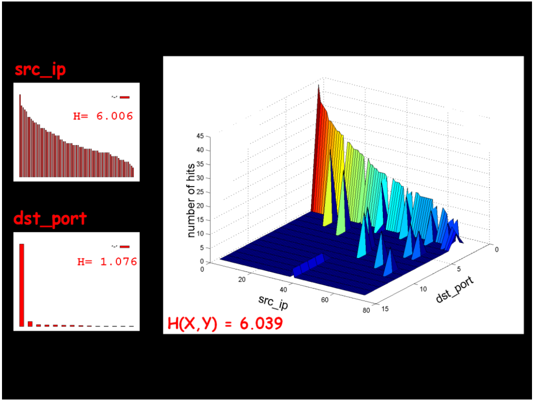
$$p(x_i, y_j) = \frac{n_{ij}}{N}, \quad (N = \sum_{i,j} n_{ij})$$

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:

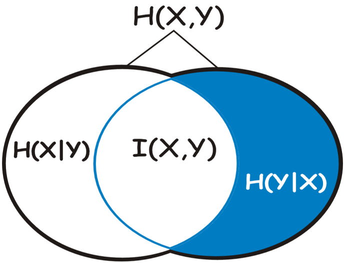
- Knowing X tells us nothing about Y
- No matter what x we fix, *the histogram of Y's values co-occurring with that x* will be the same shape
- $H(X, Y) = H(X) + H(Y)$

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

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.

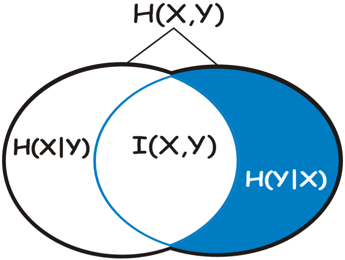
Definition

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

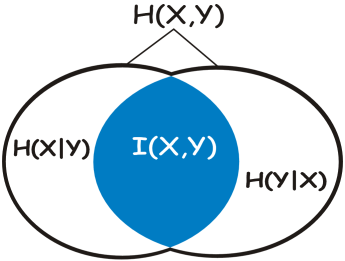


Definition

Conditional entropy of Y given X

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

Uncertainty about Y left once we know X.



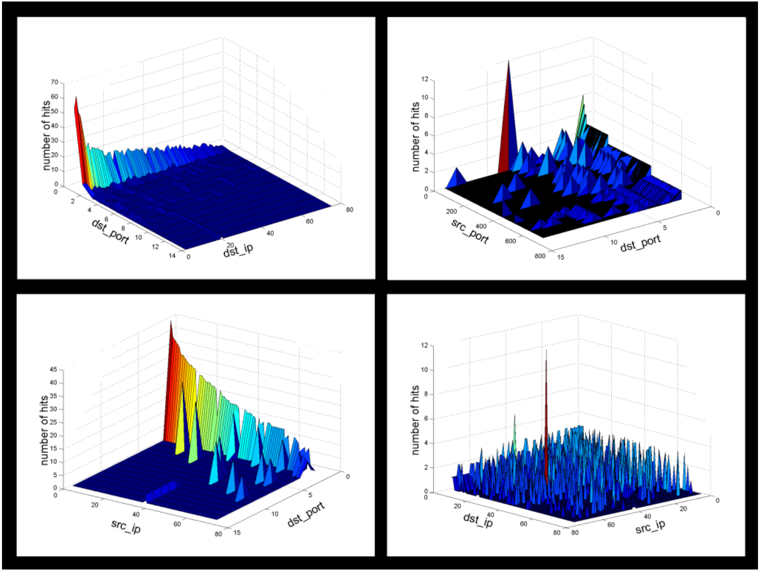
Definition

Mutual information of two variables X and Y

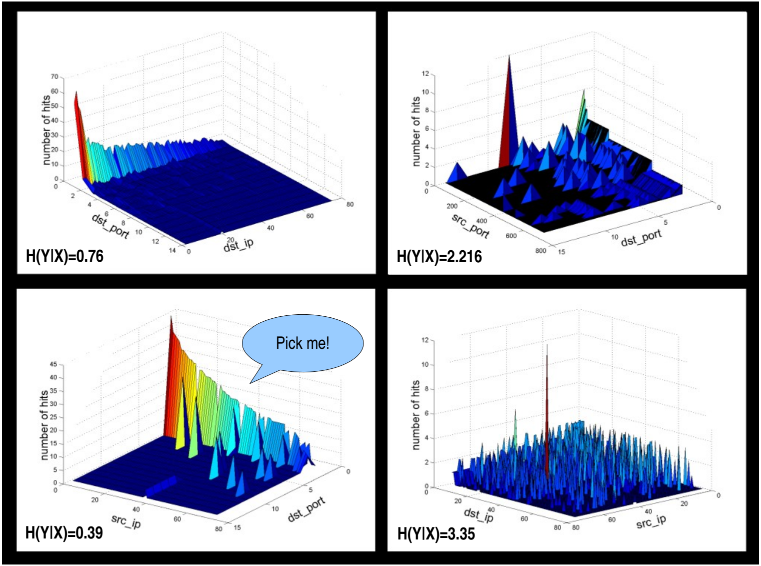
$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

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

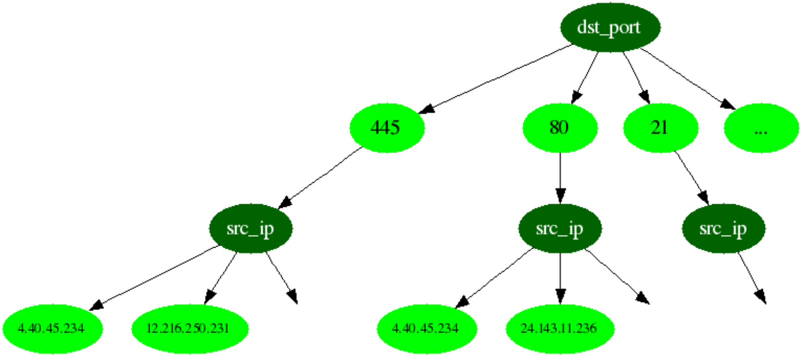
Histograms 3d: Feature pairs, Port scan



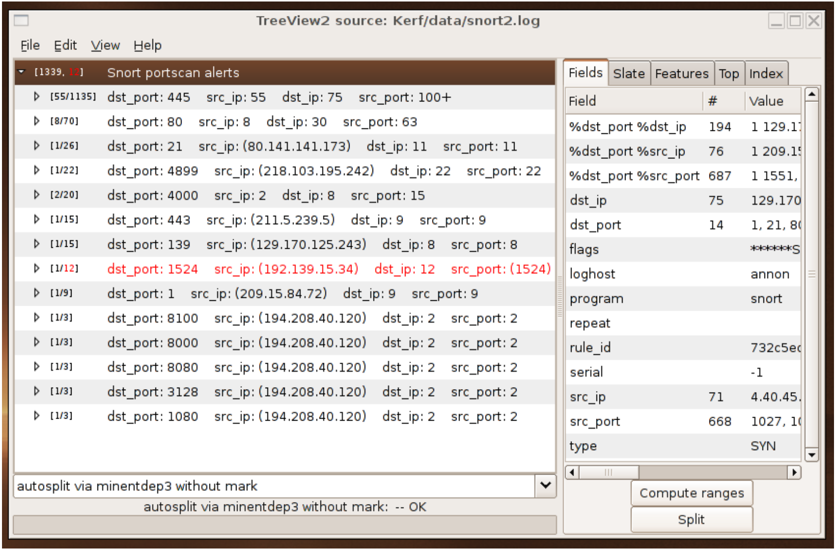
Histograms 3d: Feature pairs, Port scan



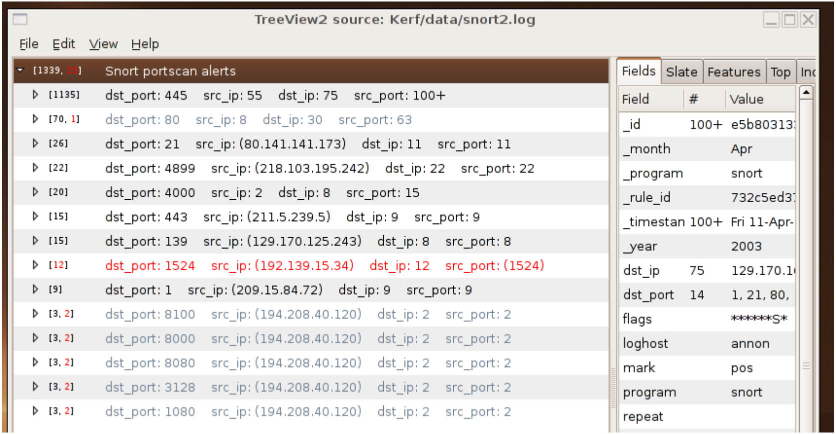
Snort port scan alerts



Snort port scan alerts



Snort port scan alerts

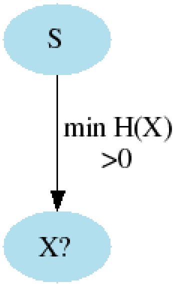


Outline

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

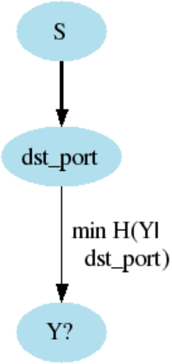
Building a data view

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)



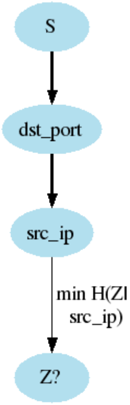
Building a data view

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)



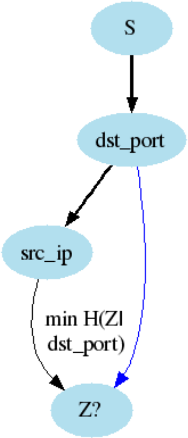
Building a data view

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)



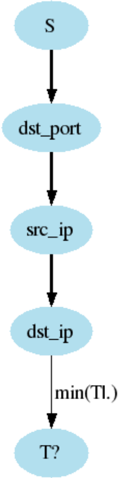
Building a data view

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)

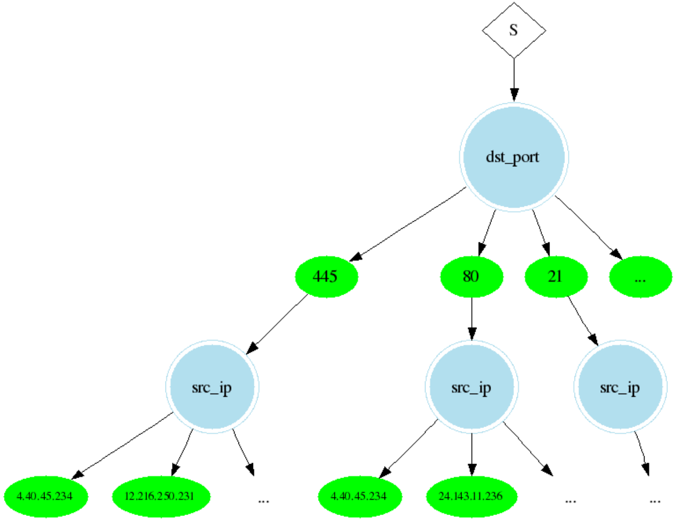


Building a data view

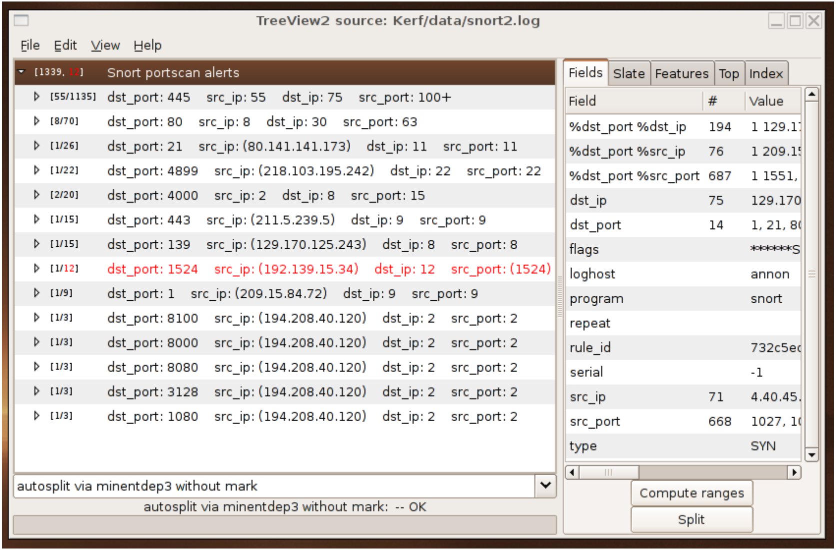
1. Pick the feature with lowest non-zero entropy (“simplest histogram”)
2. Split all records on its distinct values
3. Order other features by the strength of their dependence with with the first feature (conditional entropy or mutual information)
4. Use this order to label groups
5. Repeat with next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



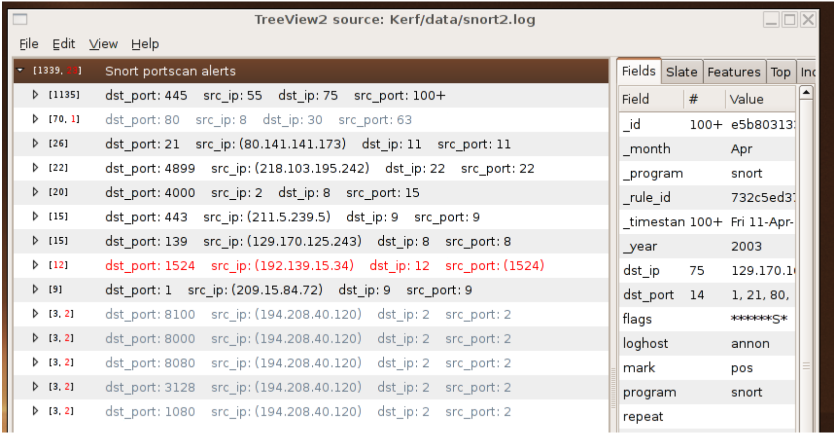
Snort port scan alerts



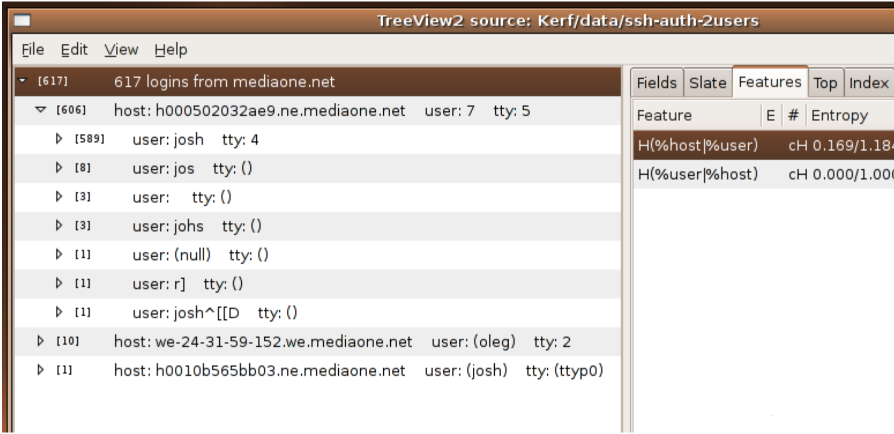
Snort port scan alerts



Snort port scan alerts

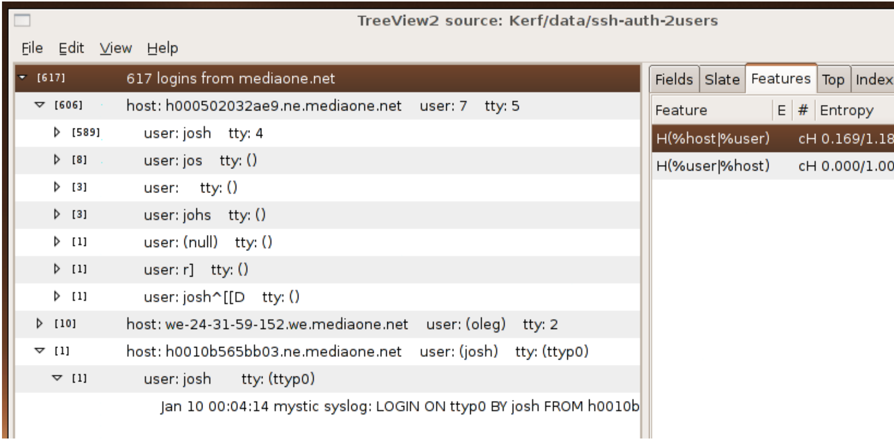


Quick pair summary



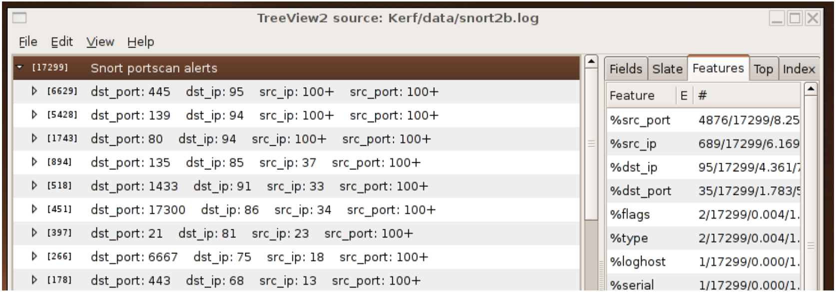
One ISP, 617 lines, 2 users, one tends to mistype.
11 lines of screen space.

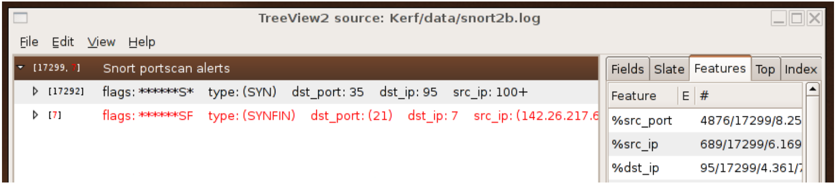
Quick pair summary



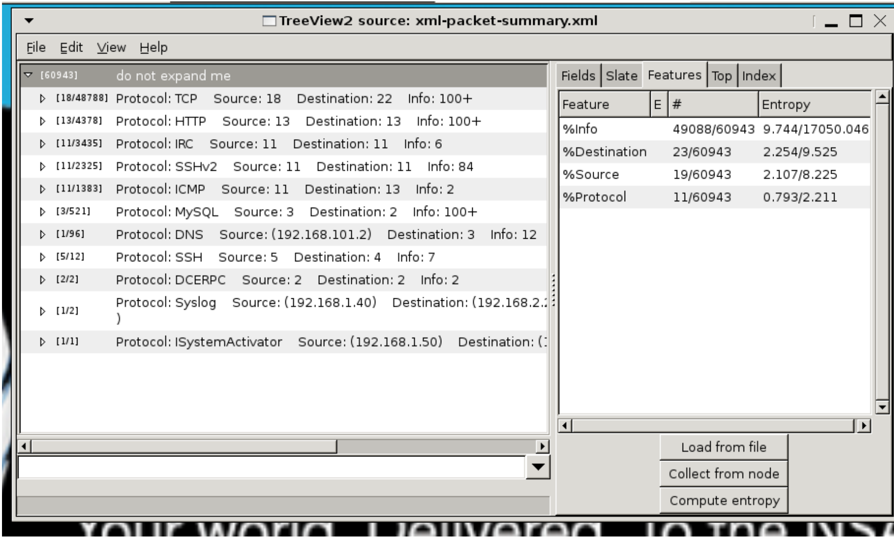
One ISP, 617 lines, 2 users, one tends to mistype.
11 lines of screen space.

Novelty changes the order

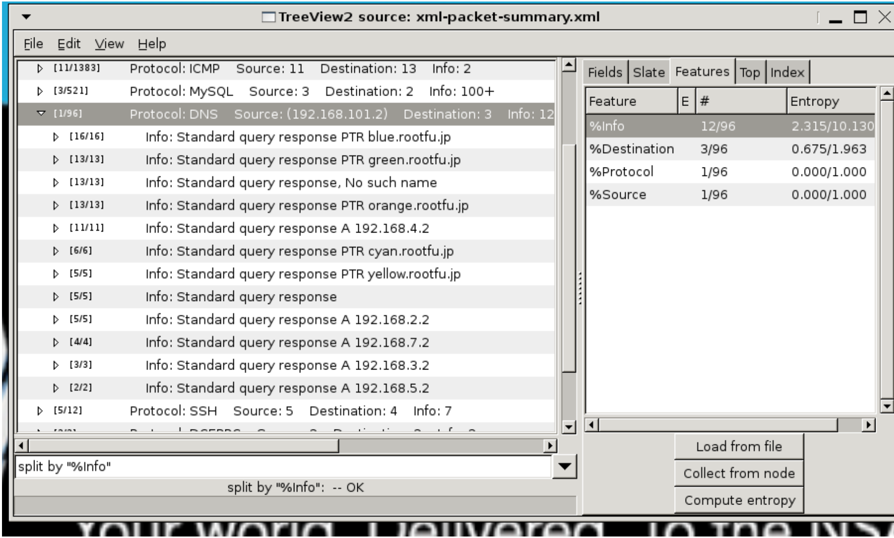




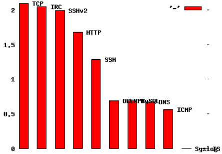
Looking at Root-Fu captures

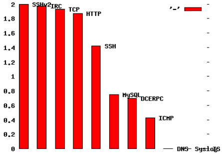


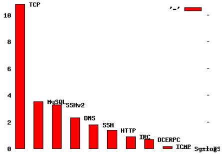
Looking at Root-Fu captures



Comparing 2nd order uncertainties

1


2


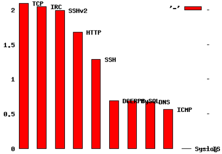
3


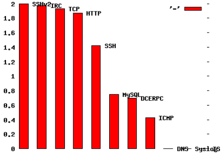
Compare uncertainties in each Protocol group:

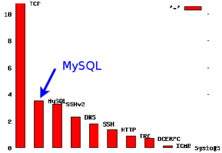
1. Destination: $H = 2.9999$
2. Source: $H = 2.8368$
3. Info: $H = 2.4957$

"Start with the simpler view"

Comparing 2nd order uncertainties

1 

2 

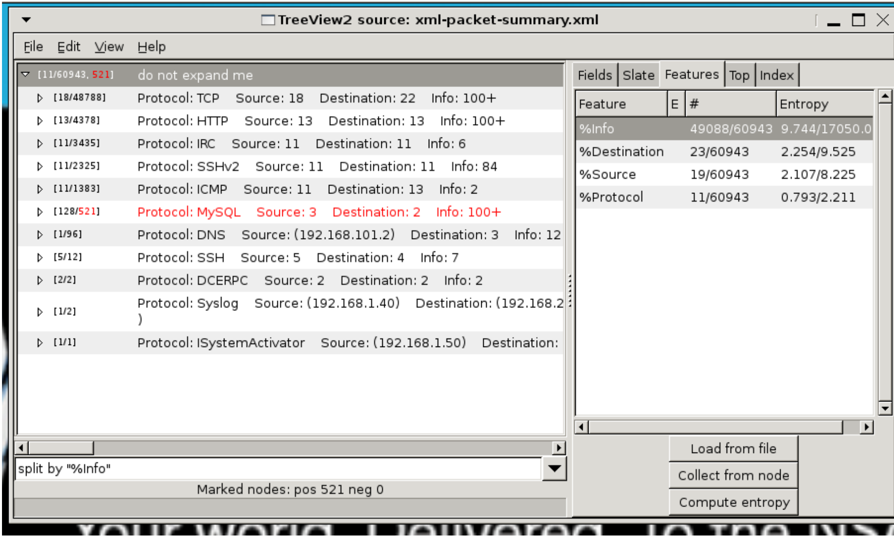
3 

Compare uncertainties in each Protocol group:

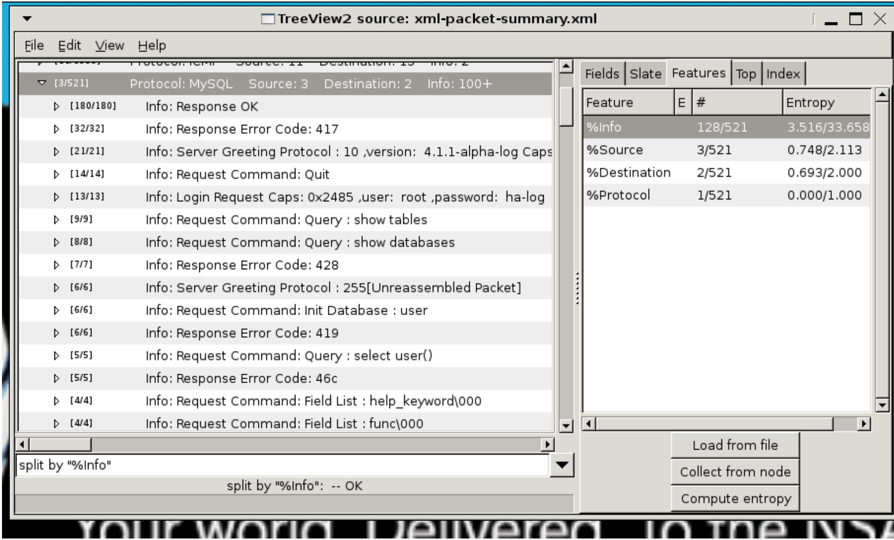
1. Destination: $H = 2.9999$
2. Source: $H = 2.8368$
3. Info: $H = 2.4957$

"Start with the simpler view"

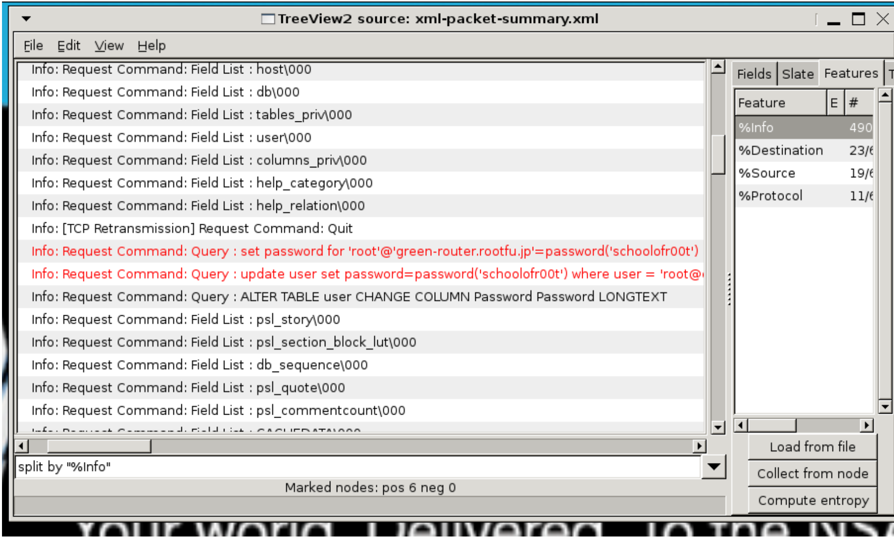
Looking at Root-Fu captures



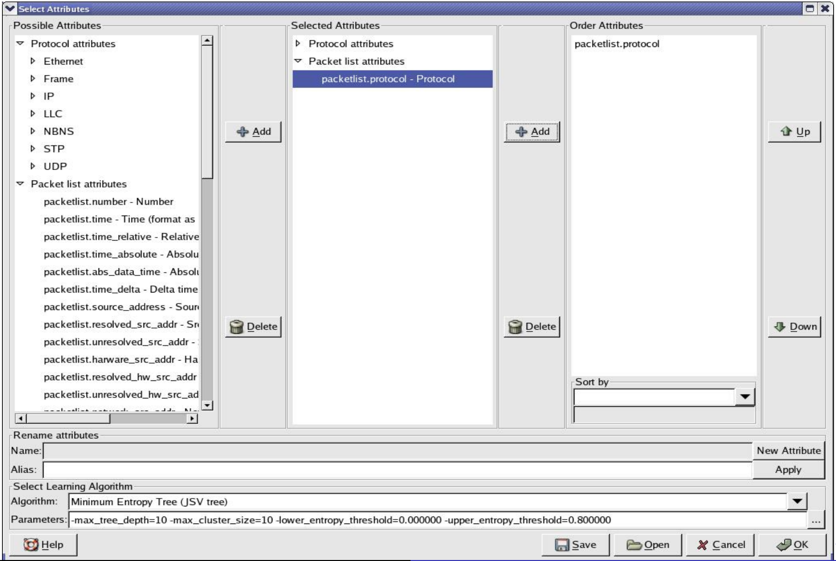
Looking at Root-Fu captures



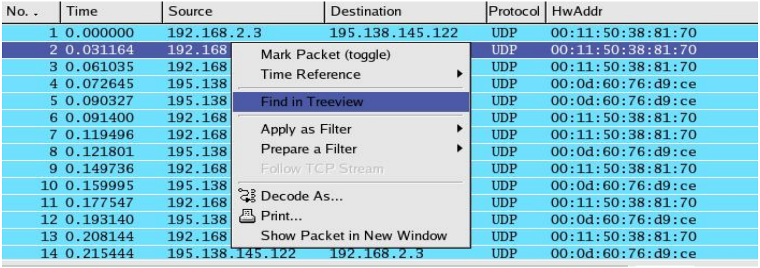
Looking at Root-Fu captures

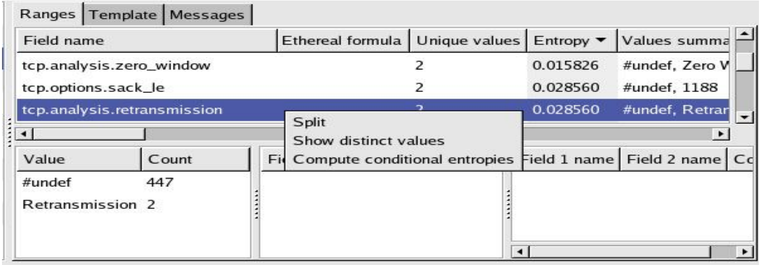


Screenshots (1)

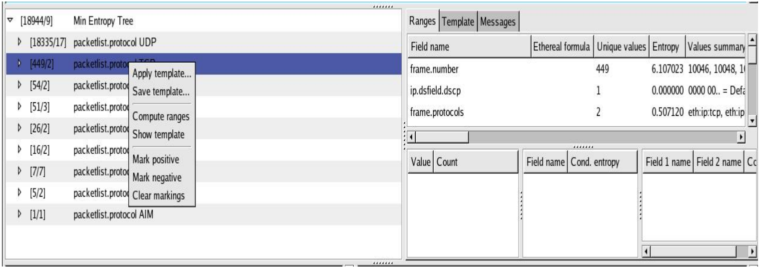


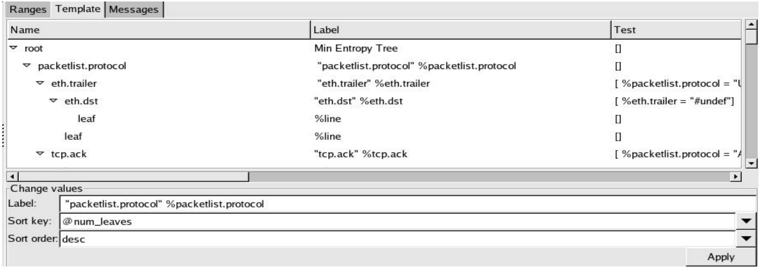
Screenshots (2)





Screenshots (3)





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

$H(X), H(X|Y), I(X;Y), \ldots$: 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

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

Credits & source code

Credits

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

Code

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

http://kerf.cs.dartmouth.edu

Thanks!