Entropy-based data organization tricks for browsing logs and packet captures

Sergey Bratus

Department of Computer Science
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
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 etc.
3. NOT on-line IDS/IPS stuff:
   - Learning the “normal” values, patterns.
   - Statistical training $\rightarrow$ black box “oracle”.
   - Once trained, hard to understand or tweak.
4. These tricks are for off-line log browsing (“analysis”).

Entropy & friends:
What can they do for us in everyday log browsing?
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

Sergey Bratus
Entropy tricks for browsing logs and packet captures
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 answer is:

<table>
<thead>
<tr>
<th>Count</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>68.38.148.149</td>
</tr>
<tr>
<td>12</td>
<td>129.10.75.101</td>
</tr>
<tr>
<td>2</td>
<td>129.170.166.85</td>
</tr>
<tr>
<td>1</td>
<td>66.183.80.107</td>
</tr>
<tr>
<td>1</td>
<td>209.61.200.11</td>
</tr>
</tbody>
</table>

Successful logins via ssh using password by IP address
...where is my WHERE clause?

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

var.log.secure

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

(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`
Log browsing moves
Data organization
Examples

“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 ...

Sergey Bratus
Entropy tricks for browsing logs and packet captures
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)
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 $\rightarrow$ branches / paths

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

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

![Diagram](image-url)
Use tree views to show logs!

Pipes, SQL queries $\rightarrow$ branches / paths

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

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

<table>
<thead>
<tr>
<th>cnt</th>
<th>ip</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>68.38.148.149</td>
</tr>
<tr>
<td>12</td>
<td>129.10.75.101</td>
</tr>
<tr>
<td>2</td>
<td>129.170.166.85</td>
</tr>
<tr>
<td>1</td>
<td>66.183.80.107</td>
</tr>
<tr>
<td>1</td>
<td>209.61.200.11</td>
</tr>
</tbody>
</table>
A “coin sorter” for records/packets

Log browsing moves  Data organization  Examples

Entropy tricks for browsing logs and packet captures
1. Build a classification tree from a dataset
2. Save template
3. Reuse on another dataset

Entropy tricks for browsing logs and packet captures
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

Sergey Bratus

Entropy tricks for browsing logs and packet captures
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.
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.
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.
- Zero in on “interesting stuff”, must fold away and ignore the rest.

The problem:

Must deal with uncertainty about the rest of the log.

Measure it!

There is a measure of uncertainty: entropy.
“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 $\rightarrow$ less uncertainty $\rightarrow$ 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.
“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 $\rightarrow$ less uncertainty $\rightarrow$ smaller entropy.
**Trivial observations, visualized**

---

**Examples**

**Entropy tricks for browsing logs and packet captures**

---

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000000</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>2</td>
<td>000017</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>3</td>
<td>000053</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>4</td>
<td>000095</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>5</td>
<td>000091</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>6</td>
<td>000844</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>7</td>
<td>000888</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>8</td>
<td>000830</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>9</td>
<td>000866</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>10</td>
<td>000819</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>11</td>
<td>000844</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>12</td>
<td>000832</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>13</td>
<td>000822</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
</tbody>
</table>

---

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

---
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
# Log browsing moves

## Data organization

### Examples

Start simple: Ranges

```
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>2</td>
<td>1.000017</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>3</td>
<td>1.999953</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>4</td>
<td>2.999925</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>5</td>
<td>3.999911</td>
<td>192.168.1.60</td>
<td>192.168.7.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>6</td>
<td>4.082445</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>7</td>
<td>4.999888</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>8</td>
<td>5.082894</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>9</td>
<td>5.999966</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>10</td>
<td>6.082219</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>11</td>
<td>6.999844</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>12</td>
<td>7.082132</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
<tr>
<td>13</td>
<td>7.999822</td>
<td>192.168.1.120</td>
<td>192.168.5.2</td>
<td>ICMP</td>
<td>Echo (ping) request</td>
</tr>
</tbody>
</table>

src ip: 192.168.1.1 - 192.168.255.255

dest ip: 192.168.1.1 - 192.168.10.100

dst port: 1-8100

---

Sergey Bratus  Entropy tricks for browsing logs and packet captures
A frequency histogram

dst_port (sorted by frequency)
Start simple: Histograms
Probability distribution

Count of packets in the “dst_port == 445” bin

Total count

“Probability” of a packet falling into the i-th bin

\[ p_i = \frac{n_i}{N}, \quad i = 1, \ldots, k \]

\( N = n_1 + \ldots + n_k \)
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

Interpretation

Entropy is a measure of uncertainty about the value of \( X \)

1. \( X = (0.25, 0.25, 0.25, 0.25) \) : \( H(X) = 2 \) (bits)
2. \( X = (0.5, 0.3, 0.1, 0.1) \) : \( H(X) = 1.685 \)
3. \( X = (0.8, 0.1, 0.05, 0.05) \) : \( H(X) = 1.022 \)
4. \( X = (1, 0, 0, 0) \) : \( H(X) = 0 \)
Interpretation

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

1. $X = (0.25, 0.25, 0.25, 0.25)$: $H(X) = 2$ (bits)
2. $X = (0.5, 0.3, 0.1, 0.1)$: $H(X) = 1.685$
3. $X = (0.8, 0.1, 0.05, 0.05)$: $H(X) = 1.022$
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. $X = (0.25, 0.25, 0.25, 0.25)$: $H(X) = 2$ (bits)

2. $X = (0.5, 0.3, 0.1, 0.1)$: $H(X) = 1.685$

3. $X = (0.8, 0.1, 0.05, 0.05)$: $H(X) = 1.022$

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. $X = (0.25, 0.25, 0.25, 0.25) : H(X) = 2$ (bits)
2. $X = (0.5, 0.3, 0.1, 0.1) : H(X) = 1.685$
3. $X = (0.8, 0.1, 0.05, 0.05) : H(X) = 1.022$
4. $X = (1, 0, 0, 0) : H(X) = 0$
Entropy is a measure of **uncertainty** about the value of $X$.

1. $X = (.25 \; .25 \; .25 \; .25) : H(X) = 2 \text{ (bits)}$
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, $\log_2 N$. 

Sergey Bratus

Entropy tricks for browsing logs and packet captures
Compare histograms

H = 5.79

dest ip 192.168.1.1 - 192.168.100

H = 1.07

dst port 1-8100

H = 6.00

src ip 192.168.1.1 - 192.168.255.255
Start with the simplest

**H = 1.07**

**dst port 1-8100**

**H = 5.79**

**dest ip 192.168.1.1 - 192.168.10.100**

**H = 6.00**

**src ip 192.168.1.1 - 192.168.255.255**

I am the simplest!
A tree grows in Ethereal

Sergey Bratus

Entropy tricks for browsing logs and packet captures
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
"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 $\rightarrow$ stronger correlation between features $\rightarrow$ smaller conditional entropy.
“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 $\implies$ stronger correlation between features $\implies$ 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 $\rightarrow$ stronger correlation between features $\rightarrow$ smaller conditional entropy.
Histograms 3d: Feature pairs

熵技巧用于浏览日志和包捕获

.src_ip

H = 6.006

.dst_port

H = 1.076

H(X,Y) = 6.039
Measure of mutual dependence

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

\[ H(X, Y) \text{ and } H(X) \]

\[ H(X) + H(Y) \text{ and } H(X, Y) \]

\( H(X, Y) = 6.039 \)
Take $N$ records with two variables $X$ and $Y$ and estimate the probabilities of seeing a pair of values

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

where $n_{ij}$ is the count of a pair $(x_i, y_j)$.
Joint Entropy

Take \( N \) records with two variables \( X \) and \( Y \) and estimate the probabilities of seeing a pair of values

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

where \( n_{ij} \) is the count of a pair \((x_i, y_j)\).

Joint Entropy

\[
H(X, Y) = \sum_{ij} p(x_i, y_j) \cdot \log_2 \left( \frac{1}{p(x_i, y_j)} \right)
\]
Joint Entropy

Take $N$ records with two variables $X$ and $Y$ and estimate the probabilities of seeing a pair of values

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

where $n_{ij}$ is the count of a pair $(x_i, y_j)$.

Joint Entropy

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

Always true:

$$H(X) + H(Y) \geq H(X, Y)$$
Joint Entropy

Take $N$ records with two variables $X$ and $Y$ and estimate the probabilities of seeing a pair of values

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

where $n_{ij}$ is the count of a pair $(x_i, y_j)$.

Joint Entropy

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

Independence

$H(X, Y) = H(X) + H(Y)$ if and only if $X$ and $Y$ are independent.
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 $y$s 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 $y$s 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 $y$s 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 $y$s 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 $Y$’s 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 $y$s co-occurring with a fixed $x$ have *different shapes*
- $H(X, Y) < H(X) + H(Y)$
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
Histories 3d: Feature pairs, Port scan

H(Y|X)=0.76
H(Y|X)=2.216
H(Y|X)=0.39
H(Y|X)=3.35
Pick me!

Sergey Bratus
Entropy tricks for browsing logs and packet captures
Snort port scan alerts
Snort port scan alerts

TreeView2 source: Kerf/data/snort2.log

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

- **flags**: ******S
- **loghost**: anon
- **program**: snort
- **repeat**: 
- **rule_id**: 732c5ed
- **serial**: -1
- **src_ip**: 71  4.40.45.
- **src_port**: 668  1027, 10
- **type**: SYN

autosplit via minentdep3 without mark

autosplit via minentdep3 without mark: -- OK
Snort port scan alerts

```
_file_ _id_ _month_ _program_ _rule_id_ _timestamp_ _year_ _dst_ip_ _dst_port_ _flags_ _loghost_ _mark_ _program_ _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 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 the first feature (conditional entropy or mutual information)
4. Use this order to label groups
5. Repeat with next feature in (1)

S

\[
\min \ H(Y|dst\_port)
\]

Y?
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 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 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 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

Entropy tricks for browsing logs and packet captures
Snort port scan alerts

TreeView2 source: Kerf/data/snort2.log

- Snort port scan alerts
  - %dst_port %dst_ip
  - %dst_port %src_ip
  - %dst_port %src_port
  - dst_ip
  - dst_port
  - flags
  - loghost
  - program
  - repeat
  - rule_id
  - serial
  - src_ip
  - src_port
  - type

- Entropy tricks for browsing logs and packet captures
Snort port scan alerts

### Snort Port Scan Alerts

<table>
<thead>
<tr>
<th>ID</th>
<th>dst_port</th>
<th>src_ip</th>
<th>dst_ip</th>
<th>src_port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1135</td>
<td>445</td>
<td>55</td>
<td>75</td>
<td>100+</td>
</tr>
<tr>
<td>70</td>
<td>80</td>
<td>8</td>
<td>30</td>
<td>63</td>
</tr>
<tr>
<td>26</td>
<td>21</td>
<td>(80.141.141.173)</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>22</td>
<td>4899</td>
<td>(218.103.195.242)</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>4000</td>
<td>2</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>443</td>
<td>(211.5.239.5)</td>
<td>9</td>
<td>src_port: 9</td>
</tr>
<tr>
<td>15</td>
<td>139</td>
<td>(129.170.125.243)</td>
<td>8</td>
<td>src_port: 8</td>
</tr>
<tr>
<td>12</td>
<td>1524</td>
<td>(192.139.15.34)</td>
<td>12</td>
<td>(1524)</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>(209.15.84.72)</td>
<td>9</td>
<td>src_port: 9</td>
</tr>
<tr>
<td>3, 2</td>
<td>8100</td>
<td>(194.208.40.120)</td>
<td>2</td>
<td>src_port: 2</td>
</tr>
<tr>
<td>3, 2</td>
<td>8000</td>
<td>(194.208.40.120)</td>
<td>2</td>
<td>src_port: 2</td>
</tr>
<tr>
<td>3, 2</td>
<td>8080</td>
<td>(194.208.40.120)</td>
<td>2</td>
<td>src_port: 2</td>
</tr>
<tr>
<td>3, 2</td>
<td>3128</td>
<td>(194.208.40.120)</td>
<td>2</td>
<td>src_port: 2</td>
</tr>
<tr>
<td>3, 2</td>
<td>1080</td>
<td>(194.208.40.120)</td>
<td>2</td>
<td>src_port: 2</td>
</tr>
</tbody>
</table>
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

Sergey Bratus

Entropy tricks for browsing logs and packet captures
Looking at Root-Fu captures

Entropy tricks for browsing logs and packet captures
Looking at Root-Fu captures
Comparing 2nd order uncertainties

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

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

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Source</th>
<th>Destination</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>18</td>
<td>22</td>
<td>100+</td>
</tr>
<tr>
<td>HTTP</td>
<td>13</td>
<td>13</td>
<td>100+</td>
</tr>
<tr>
<td>IRC</td>
<td>11</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>SSHv2</td>
<td>11</td>
<td>11</td>
<td>84</td>
</tr>
<tr>
<td>ICMP</td>
<td>11</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>MySQL</td>
<td>3</td>
<td>2</td>
<td>100+</td>
</tr>
<tr>
<td>DNS</td>
<td>(192.168.101.2)</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>SSH</td>
<td>5</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>DCERPC</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Syslog</td>
<td>(192.168.1.40)</td>
<td>(192.168.2)</td>
<td>2</td>
</tr>
<tr>
<td>ISysSystemActivator</td>
<td>(192.168.1.50)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Split by "%Info"

Marked nodes: pos 521 neg 0

Entropy tricks for browsing logs and packet captures
Looking at Root-Fu captures

Sergey Bratus

Entropy tricks for browsing logs and packet captures
Looking at Root-Fu captures
Screenshots (1)

Entropy tricks for browsing logs and packet captures
Entropy tricks for browsing logs and packet captures
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 & Christophe Diot, 2004
- Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation, Yu Gu, Andrew McCallum & Don Towsley, 2005
- ...
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

\[ H(X), H(X|Y), I(X; Y), \ldots : \text{parts of a complete analysis kit!} \]
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
\[ H(X), H(X|Y), I(X; Y), \ldots : \text{parts of a complete analysis kit!} \]
For source code (GPL), documentation, and technical reports:

http://kerf.cs.dartmouth.edu