Automated Target Definition Using Existing Evidence and Outlier Analysis

Brian D. Carrier
Eugene H. Spafford
CERIAS - Purdue University

DFRWS 2005
Searching Phases

1. Crime Scene Data Preprocessing

2. Target Definition
   • For what are we looking?

3. Crime Scene Data Processing
   • Process abstraction layers

4. Comparison
   • Are the crime scene data and the target similar?
Searching Phases

Phase 1: Crime Scene Data Pre-processing

Phase 2: Target Definition

Phase 3: Crime Scene Data Processing

Phase 4: Comparison

Crime Scene Data

Target Object

CS Data Object 1

CS Data Object 2

Target Object

CS Data Object
How Do We Define Target Objects?

• Develop a hypothesis
  – Experience and training
  – Existing evidence

• Determine what evidence would support or refute hypothesis

• Define the attributes in a target object
Existing Evidence Automation in Autopsy

1. Investigator identifies “evidence”
2. Tool makes suggestions for future searches
3. Tool saves approved searches
4. Investigator selects suggested searches
Suggested Targets

- Files in same parent directory
- Files with same temporal data
- Files with similar names
- Files with same application type
- Files with file name in content
- Files with similar content
Case Study Results

- Honeynet Forensic Challenge
- We find `/dev/ptyp` using experience
- Autopsy suggests similar times, name in content etc.
- Finds `/bin/ps` file and `/usr/man/.Ci directory`
New Topic: File Hiding Techniques

- Change MAC times
- Use similar names
- Pad data to maintain size or CRC
- Make “innocent looking” directory
- Special characters

Make file characteristics “fit in”
Spatial Outlier

- Outliers objects are “grossly different” from all other objects
- Spatial outliers are “grossly different” from local objects
- Hypothesis: Hidden files could be spatial outliers in their parent directory
Variant of Iterative Z Algorithm
Lu, Chen, & Kou - 2003 IEEE CDM

1. Compute average attribute value (g) for each directory
2. Compute distance (h_i) from each file to g
3. Compute mean (\( \mu \)) and std dev (\( \sigma \)) of \( h_i \)
4. Standardize each \( h_i \) :
   \[
   y_i = \left| \frac{h_i - \mu}{\sigma} \right|
   \]
   • If largest \( y_i \) is > \( \theta \) (2 or 3), it is an outlier
Single Attribute Results

<table>
<thead>
<tr>
<th>Attribute</th>
<th>/ partition</th>
<th>/usr/ partition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hits</td>
<td>Accuracy</td>
</tr>
<tr>
<td>App Type</td>
<td>1.01%</td>
<td>10.00%</td>
</tr>
<tr>
<td>Block</td>
<td>6.93%</td>
<td>6.22%</td>
</tr>
<tr>
<td>C-time</td>
<td>2.24%</td>
<td>26.92%</td>
</tr>
<tr>
<td>M-time</td>
<td>0.75%</td>
<td>5.77%</td>
</tr>
<tr>
<td>Size</td>
<td>4.78%</td>
<td>2.41%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hits</td>
</tr>
<tr>
<td>App Type</td>
<td>7.14%</td>
</tr>
<tr>
<td>Block</td>
<td>4.69%</td>
</tr>
<tr>
<td>C-time</td>
<td>1.51%</td>
</tr>
<tr>
<td>M-time</td>
<td>6.41%</td>
</tr>
<tr>
<td>Size</td>
<td>5.58%</td>
</tr>
</tbody>
</table>

/: 6,951 files and 50 (0.72%) involved

/usr/: 21,267 files and 323 (1.49%) involved
Results Summary

• Trojan exec on / not found, but was found on /usr/ because of starting block
• Many false positives were obvious: README files
• Generic Windows System:
  – 2.58% files identified (5.07% on honeypot)
  – 100% false positive rate
Multiple Attributes

1. Standardize all attributes
2. Compute average attribute for each directory (g) and distance for each file (hi)
3. Compute directory mean and variance-covariance

\[
\mu_s = \frac{1}{|NN(x_i)|} \sum_{x \in NN(x_i)} h(x)
\]

\[
\Sigma_s = \frac{1}{|NN(x_i)|} \sum_{x \in NN(x_i)} [h(x) - \mu_s][h(x) - \mu_s]^T
\]
Multiple Attributes (2)

4. Compute Mahalanobis distance for each file:

\[ a_i = (h(x_i) - \mu_s)^T \Sigma (h(x_i) - \mu_s) \]

5. Standardize distances and determine if largest is > 3

- Lu, Chen, Kou - 15th IEEE Conference on Tools with AI
Multiple Attribute Results

<table>
<thead>
<tr>
<th>Attribute</th>
<th>/ partition</th>
<th></th>
<th>/usr/ partition</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hits</td>
<td>Accuracy</td>
<td>Comp</td>
<td>Hits</td>
</tr>
<tr>
<td>C- and M-time</td>
<td>0.30%</td>
<td>85.71%</td>
<td>36.00%</td>
<td>0.56%</td>
</tr>
<tr>
<td>C-, M-time, and Size</td>
<td>0.22%</td>
<td>53.33%</td>
<td>16.00%</td>
<td>0.32%</td>
</tr>
<tr>
<td>C-, M-time, Size, and Block</td>
<td>0.22%</td>
<td>53.33%</td>
<td>16.00%</td>
<td>0.22%</td>
</tr>
<tr>
<td>C-, M-time, Size, Block and App Type</td>
<td>0.10%</td>
<td>57.14%</td>
<td>8.00%</td>
<td>0.15%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Combined</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hits</td>
<td>Accuracy</td>
<td>Comp</td>
</tr>
<tr>
<td>C- and M-time</td>
<td>0.50%</td>
<td>32.62%</td>
<td>12.33%</td>
</tr>
<tr>
<td>C-, M-time, and Size</td>
<td>0.29%</td>
<td>45.12%</td>
<td>9.92%</td>
</tr>
<tr>
<td>C-, M-time, Size, and Block</td>
<td>0.22%</td>
<td>37.10%</td>
<td>6.17%</td>
</tr>
<tr>
<td>C-, M-time, Size, Block and App Type</td>
<td>0.13%</td>
<td>42.11%</td>
<td>4.29%</td>
</tr>
</tbody>
</table>

Clean Windows:
C&M: 0.63%
C,M,&S: 0.54%
Hidden Directories

• Calculate average attribute value for files in a directory
• Compare with other directories at same “level”
• Detect outliers using “iterative Z”
### Directory Outlier Results

- `/` had 89 and 0 hidden directories
- `/usr/` had 1,088 and 1 hidden directory
- Was found when $\theta = 2$
- 0.78% on Windows System

<table>
<thead>
<tr>
<th>Attribute</th>
<th>/ partition</th>
<th>/usr/ partition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hits</td>
<td>Accuracy</td>
</tr>
<tr>
<td>App Type</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>C-time</td>
<td>2.25%</td>
<td>0.00%</td>
</tr>
<tr>
<td>M-time</td>
<td>1.12%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Size</td>
<td>1.12%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hits</td>
</tr>
<tr>
<td>App Type</td>
<td>6.88%</td>
</tr>
<tr>
<td>C-time</td>
<td>1.70%</td>
</tr>
<tr>
<td>M-time</td>
<td>1.36%</td>
</tr>
<tr>
<td>Size</td>
<td>3.57%</td>
</tr>
</tbody>
</table>
Conclusions

• Implemented a tool to suggest and save searches
• Implemented outlier analysis algorithms to find hidden & new files

• What is human error rate?
• Honeypot is not an ideal case study (little user activity before and after incident)
Future Work

• Investigate human error rate
• Identify which techniques work better with different types of incidents
• Incorporate data mining with other analysis techniques - keyword searching, hashes etc.
Brian Carrier

carrier@cerias.purdue.edu

www.cerias.purdue.edu