A General Strategy for Differential Forensic Analysis

By

Simson Garfinkel, Alex Nelson and Joel Young

Presented At

The Digital Forensic Research Conference

DFRWS 2012 USA Washington, DC (Aug 6th - 8th)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

http://dfrws.org
A General Strategy for Differential Forensic Analysis

Simson L. Garfinkel, Alex Nelson, Joel Young
Naval Postgraduate School
August 7, 2012
https://domex.nps.edu/deep/

“The views expressed in this presentation are those of the author and do not reflect those of the Department of Defense or the US Government.”
Differential analysis determines the differences between A and B.

\[ A \rightarrow B \]

We report these differences as a set of changes (R) that turns A into B.
Even if A and B have the same common ancestor, we can still calculate the changes R.
Differential analysis is widely used in computer forensics

Reverse engineering and malware analysis
- A and B — registry entries, DLLs, EXEs
- R — changes the malware made

User Monitoring
- A and B — disk images
- R — residual data from visiting websites (cache, cookies, etc)

Network Capacity Planning
- A and B — monthly reports of bandwidth, sites visited, etc.
- R — growth from month-to-month
Our Contribution: Strategy identification and formalization

Strategy identification
• We have written numerous differential analysis programs.
• We realized they all used roughly the same strategy.
• Those that didn’t use the strategy had bugs!
• When we implemented the strategy completely, the bugs went away!

Strategy formalization
• A consistent terminology
• Application of this terminology to several scenarios
Our terminology for differential analysis

Image — A byte stream from any data-carrying device
  • e.g. disk images, memory images, cell phone images; may be physical or logical

Baseline Image (A)
  — The first image acquired at time $T_A$

Final Image (B)
  — The first image acquired at time $T_B$

Intermediary Images ($I_n$)
  — Zero or more images recorded between the Baseline and Final Images $T_A$

Common Baseline
  — A single image that is a common ancestor

Image Delta (B-A)
  — The differences between two images

Differencing Strategy
  — A strategy for reporting differences between two or more images
Our strategy is based on extracted features.

Feature $f$

— A piece of data or information that is explicitly extracted from the image...
  ... or otherwise dependent upon data within the image.

Feature in Image $(A,f)$

— Features typically are found in images.
  — $(A,f)$ is feature $f$ in image $A$

Feature Name $\text{NAME}(A,f)$

— Every feature may have zero, one or multiple names
  — If the feature is the contents of a file, the feature name might be the file name

Feature Location $\text{LOC}(f)$

— The location where the feature is found
  — For files, could be an inode, or sector #; features can have multiple locations
Features are extracted from images

Feature Extraction \( F(A) \)

- *The set of features extracted from an image*

Feature Set Delta \( F(B) - F(A) \)

- *The differences between the feature sets extracted from two images*

Transformation Set \( R \)

- *The specific set of operations that are applied to A to produce B*
- *For example, the diff “patch file”*
Prior work: differential analysis goes back 40 years!

Historical:
- `diff` (Thompson & Ritchie, 1975)
- Tripwire (Kim & Spafford, 1994)
  
  —Can largely be implemented with hashdeep (Kornblum) or ffs (Carrier) & diff

Forensics:
- EnCase and FTK — Manual differencing
- WiReD (NIST, 2009)
- Teleporter (Watkins, 2009)

Data Synchronization
- rsync — Direct examination of file system
- Unison — Examination of file system metadata snapshots

Revision Control Systems
- Centralized systems — RCS & Subversion
- Uncentralized — git, Darcs

Even timeline analysis is differential analysis
- CAT Detect (Marrington 2011) looks for R that is inconsistent with underlying OS.
Forensics practitioners use many forms of differential analysis.

Differential analysis is a primary tool for addressing data overload. Feature selection allows the analysis to focus on what’s important.

- **Malware Discovery and analysis**
  - *Identifies what the malware did*

- **Insider Threat Identification**
  - *Identifies abnormalities in time and space*

- **“Pattern of Life”**
  - *What a user does habitually*
  - *Computer used by multiple individuals*
  - *Multiple accounts used by a single person*
  - *Hijacked accounts*

- **Summarized reporting of what matters**
  - *Introduction of new features*
  - *Increase in count of an existing feature*
  - *Decrease in count of an existing feature*
  - *Removal of a feature from the image*
  - *Relocation of feature*
Every feature has content and metadata.
Change primitives transform A → B

Feature content — the feature’s byte sequence.

Feature metadata
- Location
- Name
- Timestamp(s) and other metadata

Image “A” and “B” are collections of features
- F(A) & F(B)
- R is a set of changes that transform F(A) → F(B)
A simple set of rules allows us to detect changes.

- If something did not exist and now it does, it was created
- If it did exist before and now it does not, it was deleted
- If it is in a new location, it was moved
- If more copies of it exist, it was copied
- If less copies of it exist, something got deleted
- Aliasing means names can be added or deleted

| Table 1: Change detection rules in English. |
### Abstract rules for transforming $A \rightarrow B$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Change Primitive for $A \rightarrow B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f \in F(A)$ and $f \in F(B)$</td>
<td>(no change)</td>
</tr>
<tr>
<td>$f \in F(A)$ and $f \notin F(B)$</td>
<td>DELETE $f$</td>
</tr>
<tr>
<td>$f \notin F(A)$ and $f \in F(B)$</td>
<td>CREATE $f$</td>
</tr>
</tbody>
</table>

| $|\text{LOC}(A, f)| = 1$ and $|\text{LOC}(B, f)| = 1$ and $\text{LOC}(A, f) \neq \text{LOC}(B, f)$ | MOVE $\text{LOC}(A, f) \rightarrow \text{LOC}(B, f)$ |
| $|\text{LOC}(A, f)| < |\text{LOC}(B, f)|$ | COPY $\text{LOC}(A, f) \rightarrow (\text{LOC}(B, f) \setminus \text{LOC}(A, f))$ |
| $|\text{LOC}(A, f)| > |\text{LOC}(B, f)|$ | DELETE $(\text{LOC}(A, f) \setminus \text{LOC}(B, f))$ |

| $|\text{NAME}(A, f)| = 1$ and $|\text{NAME}(B, f)| = 1$ and $\text{NAME}(A, f) \neq \text{NAME}(B, f)$ | RENAME $\text{NAME}(A, f) \rightarrow \text{NAME}(B, f)$ |

| ($|\text{NAME}(A, f)| \neq 1$ or $|\text{NAME}(B, f)| \neq 1$) and $n \notin \text{NAME}(A, f)$ and $n \in \text{NAME}(B, f)$ | ADDNAME $f, n$ |

| ($|\text{NAME}(A, f)| \neq 1$ or $|\text{NAME}(B, f)| \neq 1$) and $n \in \text{NAME}(A, f)$ and $n \notin \text{NAME}(B, f)$ | DELNAME $f, n$ |

Table 2: Abstract Rules for transforming $A \rightarrow B$ (A into B) based on observed changes to features ($f$), feature locations ($\text{LOC}(A, f)$), and feature names ($\text{NAME}(A, f)$). Although the RENAME primitive is not strictly needed (it can be implemented with a ADDNAME and a DELNAME), it is useful to distinguish the two operations.
These rules can also detect temporal inconsistencies.

If features have timestamps...

    and A and B are from the same system...
    and $T_B > T_A$

Then every feature in $F(B) - F(A)$ should have a timestamp after $T_A$.

Sources for temporal inconsistencies:

- Tampering of the system clock
- Copy programs (cp, copy) tampering destination mtime to match source
- Inconsistency in the way that time is updated
  - Inconsistent updates to Windows Registry hive last-update key
  - Windows rounding times to the hour
- Tool error
Forensic examiners *must* suppress extraneous information

Approaches for suppressing:

- Do not extract information that will not be reported
- Present *counts* rather than the actual features
- Organize features in a hierarchy
- Organize features in timelines
Tools we have written

idifference — differences between two disk images
  • Files added, removed, moved, changed
  • Timestamp modifications without file content changes

rdifference — differences between two Windows Registry hives
  • Deleted cells
  • Values with modified content or type
  • Keys with changed mtimes
    — Note: must handle Registry hives where multiple keys have the same name!

bulk_diff — Differences between two bulk_extractor reports
  • New email addresses, URLs, search terms, etc.
  • Allows one to rapidly infer “what happened” without examining files, browser cache, etc.

corpus_sync — uses change detection to sync NPS disk corpus

flow_diff — (under development) reports new services on pcap dumps
Case study: M57 illicit images

M57-Patents scenario: fictitious characters, working for fictitious company, committing crimes.

One persona, Jo, is a (simulated) pornographer: Kitty porn (JPEGs)

_How can we use differencing to quickly find suspected illicit pictures?_
Differencing reduces the amount of information that needs review.

<table>
<thead>
<tr>
<th></th>
<th>Nov. 12 → 20</th>
<th>Nov. 12 → 16</th>
<th>Nov. 16 → 19</th>
<th>Nov. 19 → 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files before</td>
<td>24,131</td>
<td>24,131</td>
<td>28,735</td>
<td>29,678</td>
</tr>
<tr>
<td>Files after</td>
<td>30,497</td>
<td>28,735</td>
<td>29,678</td>
<td>30,497</td>
</tr>
<tr>
<td>New files</td>
<td>8,546</td>
<td>5,140</td>
<td>1,157</td>
<td>2,773</td>
</tr>
<tr>
<td>Deleted files</td>
<td>1,900</td>
<td>200</td>
<td>98</td>
<td>1,814</td>
</tr>
<tr>
<td>Renamed files</td>
<td>463</td>
<td>449</td>
<td>566</td>
<td>703</td>
</tr>
<tr>
<td>Files w/ changed content</td>
<td>1,011</td>
<td>687</td>
<td>981</td>
<td>568</td>
</tr>
<tr>
<td>Files w/ changed metadata</td>
<td>3,581</td>
<td>1,906</td>
<td>4,275</td>
<td>1,784</td>
</tr>
<tr>
<td>Difference report lines with ‘.jpg’</td>
<td>603</td>
<td>33</td>
<td>146</td>
<td>643</td>
</tr>
</tbody>
</table>

Tools used:

- fiwalk, idifference
Conclusion:
All differencing tasks are fundamentally identical.

We have written *many* differencing tools.

- File system differencing
- Windows Registry differencing
- bulk_extractor output differencing
- Corpus synchronization

We realized that *all* of these tools implemented the same strategy.

- An “image” is a collection of “features.”
- Differencing determines the changes needed to change A → B
- This is the same as F(A) → F(B)
- Dividing the changes into categories eases reporting:
  - New features
  - Missing features
  - Features with changed names
  - Features with changed addresses

Questions?