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)

A General Strategy for Differential Forensic Analysis

Simson L. Garfinkel, Alex Nelson, Joel Young
Naval Postgraduate School

August 7, 2012

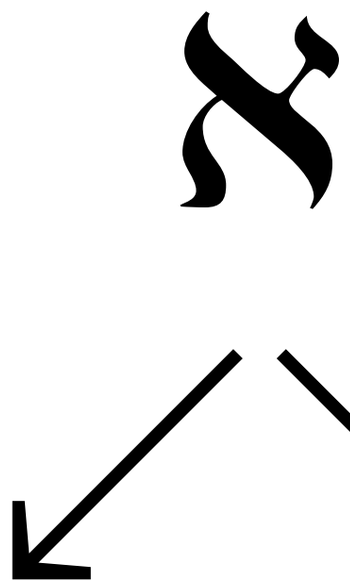
https://domex.nps.edu/deep/

Differential analysis determines the differences between A and B.

$$A \xrightarrow{R} B$$

We report these differences as a set of changes (R) that turns A into B.

B does not necessarily come from A.



$$A \xrightarrow[R]{} B$$

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

$$A \xrightarrow[R]{} B$$

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*

$$A \xrightarrow[R]{} B$$

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*

$$\begin{tikzcd} & \aleph \arrow[dl] \arrow[dr] & \\ A \arrow[rr, "R"'] & & B \end{tikzcd}$$

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 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 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!

$$A \underset{R}{\longrightarrow} B$$

Historical:

- diff (Thompson & Ritchie, 1975)
- Tripwire (Kim & Spafford, 1994)
  - *Can largely be implemented with hashdeep (Kornblum) or fls (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

$$A \xrightarrow[R]{} B$$

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 → B

Rule	Change Primitive for $A \underset{R}{\rightarrow} B$
$f \in F(A)$ and $f \in F(B)$	(no change)
$f \in F(A)$ and $f \notin F(B)$	DELETE f
$f \notin F(A)$ and $f \in F(B)$	CREATE f
$\lvert\mathrm{LOC}(A, f)\rvert = 1$ and $\lvert\mathrm{LOC}(B, f)\rvert = 1$ and $\mathrm{LOC}(A, f) \neq \mathrm{LOC}(B, f)$	MOVE $\mathrm{LOC}(A, f) \rightarrow \mathrm{LOC}(B, f)$
$\lvert\mathrm{LOC}(A, f)\rvert < \lvert\mathrm{LOC}(B, f)\rvert$	COPY $\mathrm{LOC}(A, f) \rightarrow (\mathrm{LOC}(B, f) \setminus \mathrm{LOC}(A, f))$
$\lvert\mathrm{LOC}(A, f)\rvert > \lvert\mathrm{LOC}(B, f)\rvert$	DELETE $(\mathrm{LOC}(A, f) \setminus \mathrm{LOC}(B, f))$
$\lvert\mathrm{NAME}(A, f)\rvert = 1$ and $\lvert\mathrm{NAME}(B, f)\rvert = 1$ and $\mathrm{NAME}(A, f) \neq \mathrm{NAME}(B, f)$	RENAME $\mathrm{NAME}(A, f) \rightarrow \mathrm{NAME}(B, f)$
$(\lvert\mathrm{NAME}(A, f)\rvert \neq 1$or $\lvert\mathrm{NAME}(B, f)\rvert \neq 1)$ and $n \notin \mathrm{NAME}(A, f)$ and $n \in \mathrm{NAME}(B, f)$	ADDNAME f, n
$(\lvert\mathrm{NAME}(A, f)\rvert \neq 1$or $\lvert\mathrm{NAME}(B, f)\rvert \neq 1)$ and $n \in \mathrm{NAME}(A, f)$ and $n \notin \mathrm{NAME}(B, f)$	DELNAME f, n

Table 2: Abstract Rules for transforming $A \rightarrow B$ (A into B) based on observed changes to features (f), feature locations ($\mathrm{LOC}(A, f)$), and feature names ($\mathrm{NAME}(A, f)$). 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?

Difference statistics - M57 illicit image machine (“Jo”)

Differencing reduces the amount of information that needs review.

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

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?