



A Combination of  
Advanced Carver and Intelligent Parser  
Teru Yamazaki  
Cyber Defense Institute, Inc.

The 9th Annual Open Source Digital Forensics Conference



# Teru Yamazaki

Forensic Investigator, Instructor, and Researcher

- [Twitter] @4n6ist
- [Blog] <https://www.kazamiya.net/>
- [Programming] C/C++/C#/EnScript
- Free Tools / Open Source Tools
  - fte
  - NSRLJP
  - HFS Journal Parser EnScript
  - KaniVola
  - CDIR
  - **bulk\_extractor-rec**
  - **usn\_analytics**

# Contents

1

Background

2

Advanced Carver

Extracting More Potential Evidence

3

Intelligent Parser

Producing More Valuable Information

4

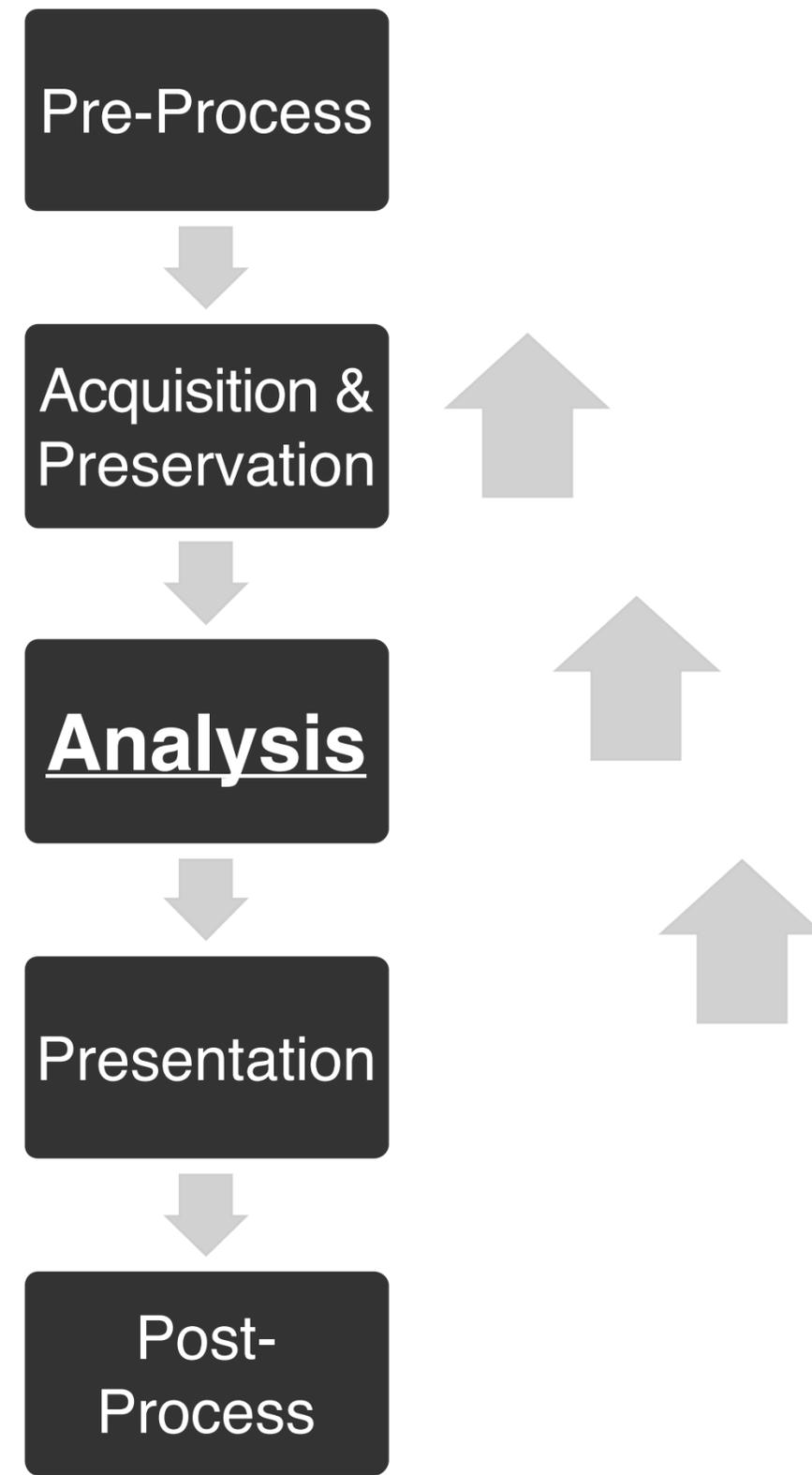
Conclusion

3



# 1. Background

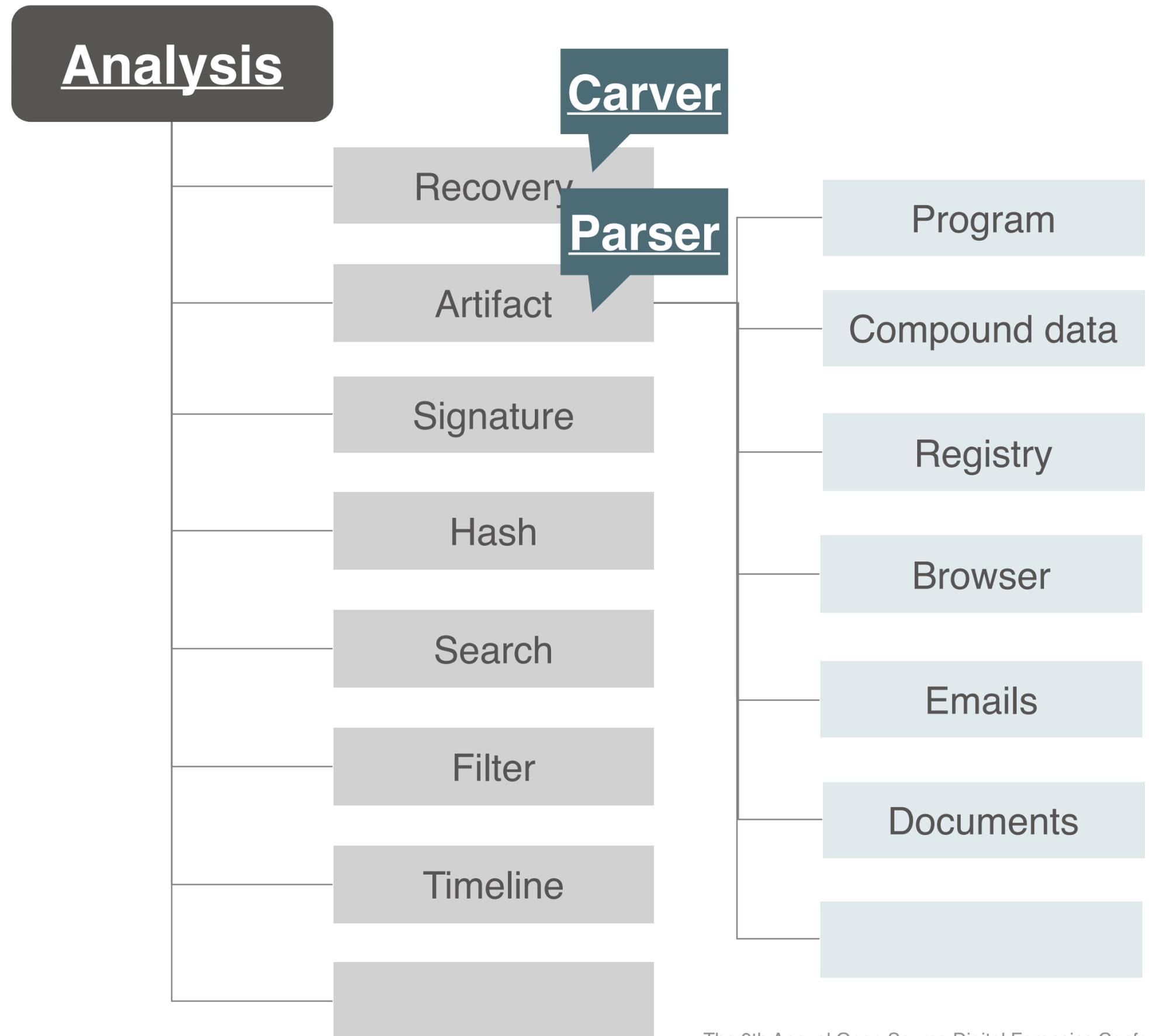
# Generic Computer Forensic Investigation Model [1]



COMMON PHASES OF COMPUTER FORENSICS INVESTIGATION MODELS

<http://airccse.org/journal/jcsit/0611csit02.pdf>

# Drilling Down on Analysis Phase



# Why Carver and Parser?

- Carver
  - At present, most forensic tools support carving out a **file**
  - In the past, Jeff Hamm talked about records carving [2]
  - The idea inspired me to develop record carving scanners
- Parser
  - Typical parser produces a huge amount of records, depending on the artifacts
  - I would like to get rid of unnecessary records without information loss
  - I would like to produce valuable information in one artifact itself

[2] Carve for Records Not Files

<https://digital-forensics.sans.org/summit-archives/2012/carve-for-record-not-files.pdf>

# A combination of Carver and Parser



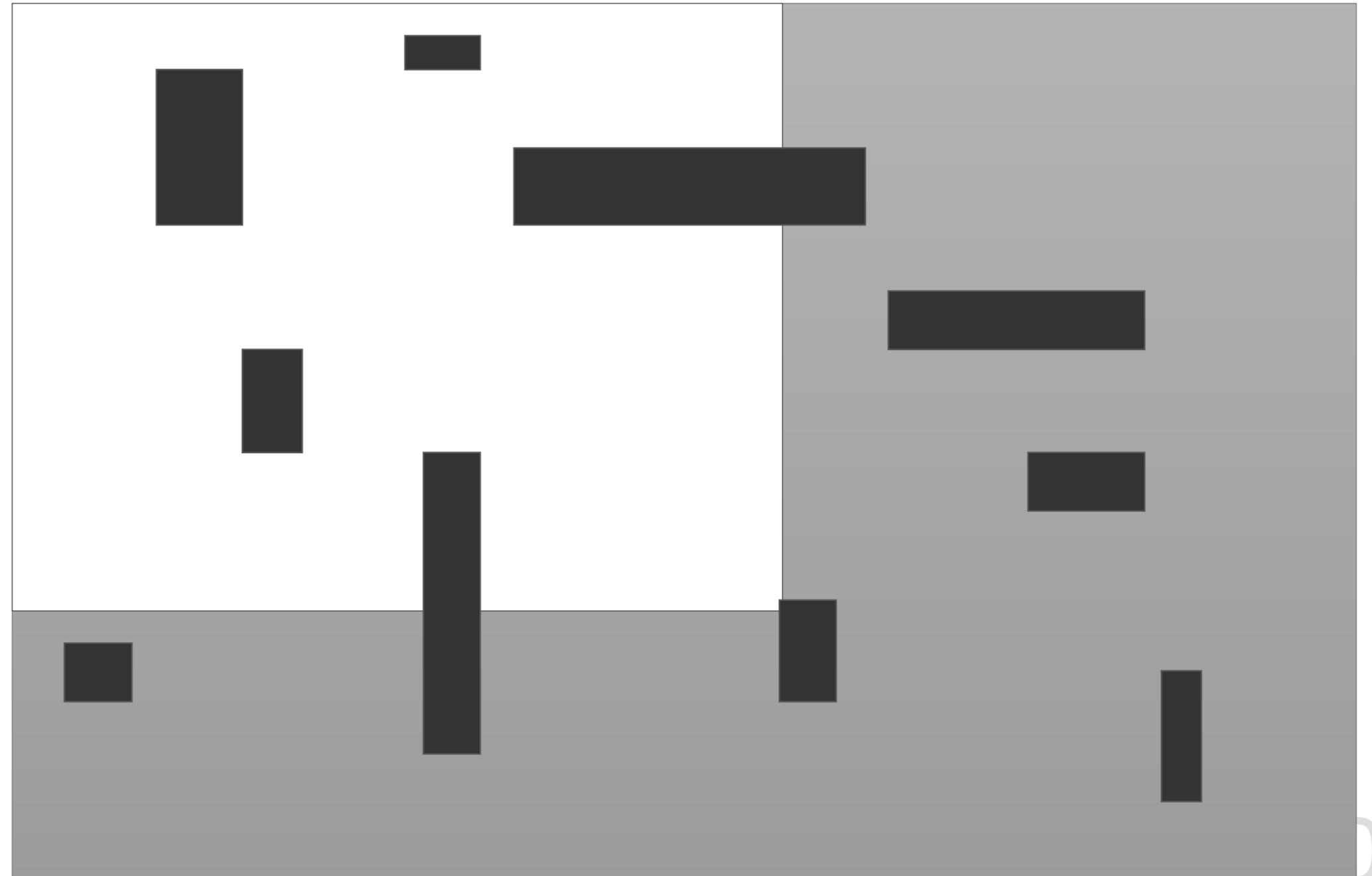
extracts more potential evidence and produces valuable information, giving you better results !



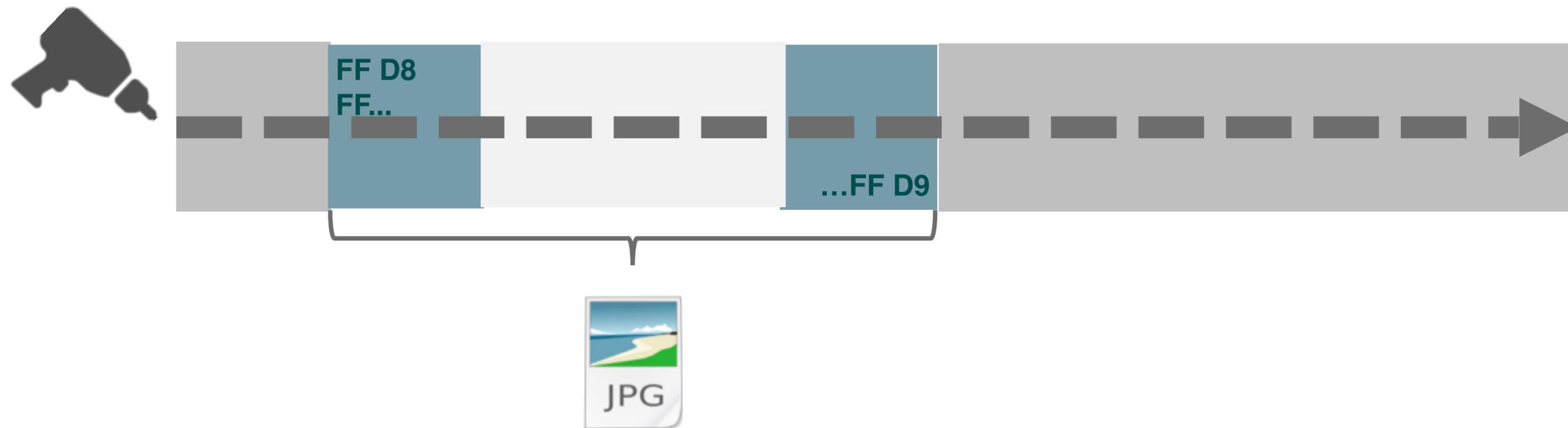
## 2. Advanced Carver

# Carving Big Picture

How do you find meaningful data  
in entire storage?



# File Carving



“File Carving, or sometimes simply Carving, is the practice of searching an input for files or other kinds of objects based on content, rather than on metadata” [3]

[3] File Carving

[http://www.forensicswiki.org/wiki/File\\_Carving](http://www.forensicswiki.org/wiki/File_Carving)

# Challenges of File Carving

- Range estimation

“Not all file types have a uniquely identifiable final data block and may require tools to guess where the end of the file is located.” [4]

- Fragmentation

“If a complete source file is present in the search arena, but the file is fragmented then the carving tool needs to be capable of identifying all file fragments and assembling the fragments in the correct order. This is not an easy task and may not be possible in many cases.” [4]

- Partially overwritten files

“If a source file is incomplete within the search arena then it may be possible to assemble the first or last part of a file from the available data, but this may not be possible in many cases.” [4]

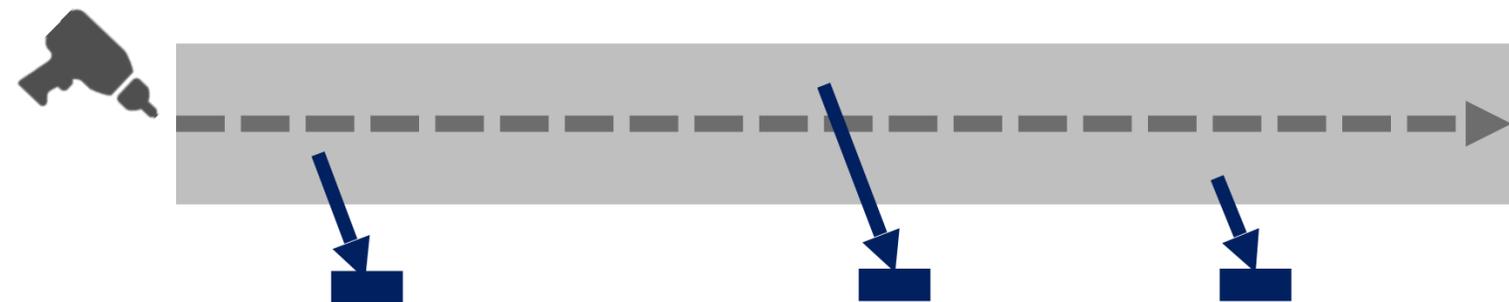
[4] Forensic File Carving Tool Specification Version 1.0

<https://www.nist.gov/sites/default/files/documents/2017/05/09/fc-req-public-draft-01-of-ver-01.pdf>

The 9th Annual Open Source Digital Forensics Conference

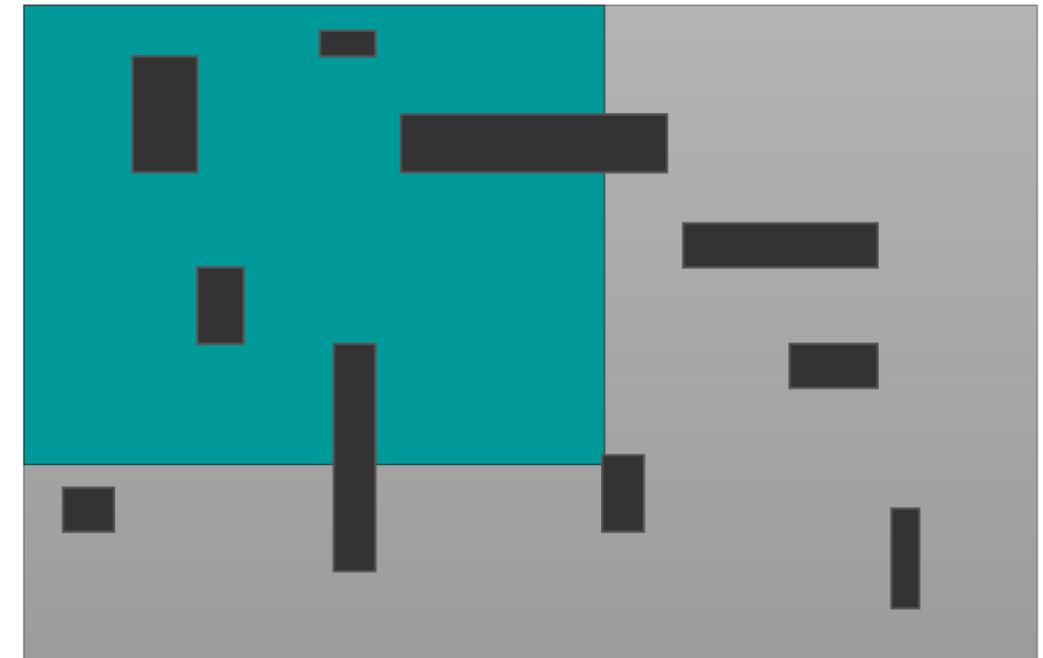
# From files to pieces

- Many file types have unique signatures
- A lot of pieces such as chunks, blocks, records, and nodes also have unique signatures
- I refer to such pieces as **records** in this talk
- Record Carving can be one of the solutions for carving challenges



# Searching Not Only Unallocated Space but Entire Space

- We should include allocated area when carving records because there are a lot of pieces of records in their space
- Compound files
- VSC snapshots
- NTFS Initialized space
- RAM
- Hibernation space
- Swap space



- Many types of input
- Multi platforms
- Buffer handling
- Recursive process
- Fast processing
- Plug-in architecture and Open Source

# Bulk Extractor[5]

- Carving Infrastructure -

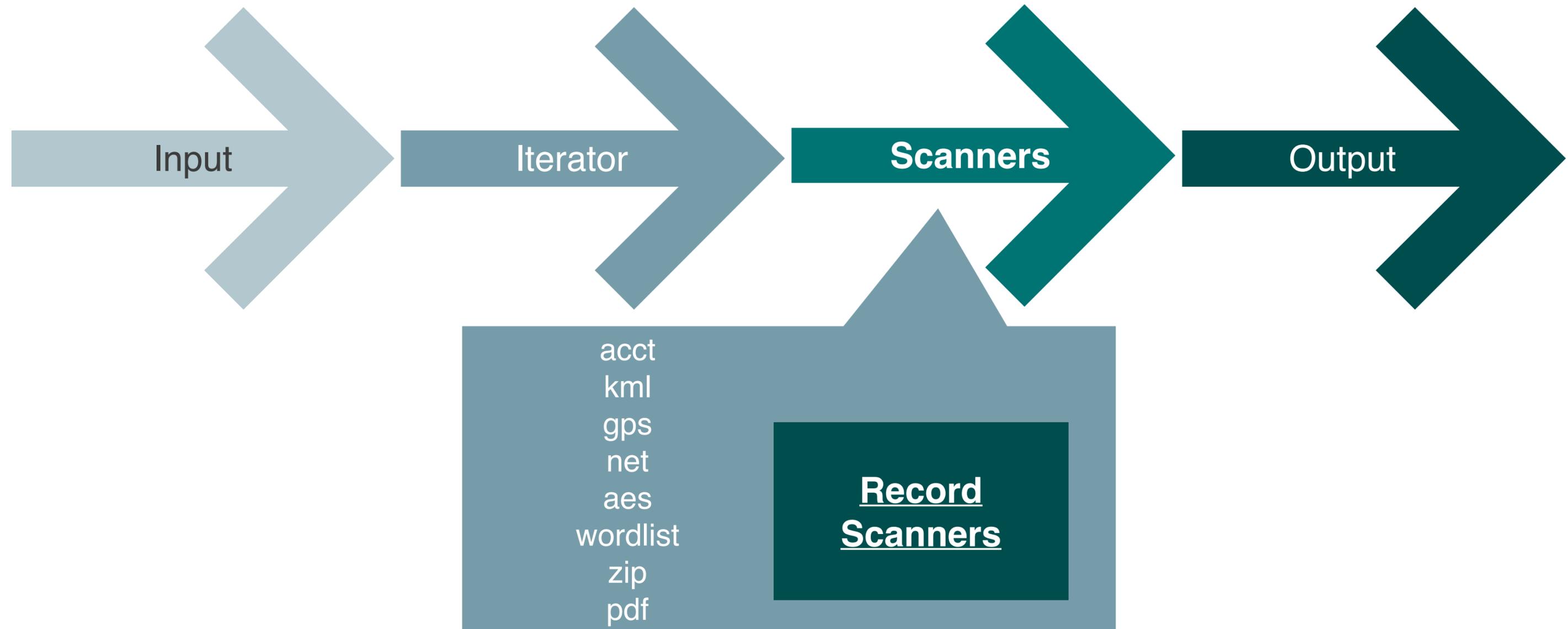


acct  
kml  
gps  
net  
aes  
wordlist  
zip  
pdf

- Many types of input
- Multi platforms
- Buffer handling
- Recursive process
- Fast processing
- Plug-in architecture and Open Source

# Bulk Extractor with Record Carving

[https://www.kazamiya.net/en/bulk\\_extractor-rec](https://www.kazamiya.net/en/bulk_extractor-rec)



# Steps of Record Scanners Development

1. Install Fedora and required packages
2. Get bulk\_extractor's repository
3. **Create a scanner file named *plugin\_name.cpp***
4. Update Makefile.am, bulk\_extractor\_scanners.cpp, and bulk\_extractor\_noscanners.cpp

# How to Implement Record Scanners

1. Understand data format
2. **Create core rules**
3. Determine a process flow
4. Write code
5. Repeat trial and error

# Create core rules

- To reduce noise and find more records, we must create robust signature from a specification and actual records
  - Magic bytes  
ideal for a lot of unique patterns
  - Offset  
may be useful
  - Date  
useful if it indicates a limited range
  - Integer  
useful if it indicates a limited range (i.e. positive number, minimum value, and maximum value)
  - Strings  
useful if these are assumed ASCII (i.e. 0x00-0x7F and ends with 0x00)

# Record Carving Scanners

- ntfsindx
- ntfslogfile
- ntfsmft
- **ntfsusn**
- **utmp**
- **evtx**

# USN\_RECORD\_V2 Structure [6]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	<b><i>RecordLength</i></b>			<b><i>Major Version</i></b>		<b><i>Minor Version</i></b>		<b><i>FileReferenceNumber</i></b>								
10h	ParentFileReferenceNumber								USN							
20h	<b><i>TimeStamp</i></b>								<b><i>Reason</i></b>				<b><i>SourceInfo</i></b>			
30h	SecurityId			FileAttributes				FileName Length		FileName Offset		FileName ...				

[6] USN\_RECORD\_V2 structure

[https://msdn.microsoft.com/ja-jp/library/windows/desktop/aa365722\(v=vs.85\).aspx](https://msdn.microsoft.com/ja-jp/library/windows/desktop/aa365722(v=vs.85).aspx)

Note: Currently, USN\_RECORD\_V3 and USN\_RECORD\_V4 are disabled by default

# Signature for USN Record

Based on actual record

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	<b><i>64-600 and multiple of 8</i></b>				<u>02</u>	<u>00</u>	<u>00</u>	<u>00</u>	<b><i>FileReferenceNumber</i></b>							
10h	ParentFileReferenceNumber								USN							
20h	<b><i>TimeStamp</i></b>								<b><i>Reason</i></b>				<b><i>SourceInfo</i></b>			
30h	SecurityId				FileAttributes				<b>2-512</b>	<u>3C</u>	<u>00</u>	FileName ...				

Unicode and length (1-256)

# utmp record format (Linux)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	<i>ut_type</i>				<i>ut_pid</i>				<i>ut_line</i>							
10h	<i>ut_line</i>															
20h	<i>ut_line</i>								<i>ut_id</i>				<i>ut_user (32 bytes)</i>			
...	...															
140h	<i>ut_host (256 bytes)</i>												<i>ut_exit</i>			
150h	<i>ut_session</i>				<i>tv_sec</i>				<i>tv_usec</i>				<i>ut_addr_v6</i>			
160h	<i>ut_addr_v6</i>												<i>unused</i>			
170h	<i>unused</i>															

[7] utmp(5) – Linux manual page

<http://man7.org/linux/man-pages/man5/utmp.5.html>

# Signature for utmp Record (Linux)

Based on actual record

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	<b>1-8</b>	<u>00</u>	<u>00</u>	<u>00</u>	<i>ut_pid</i>				<b>ASCII</b>							
10h	<b>ASCII</b>															
20h	<b>ASCII</b>								<i>ut_id</i>				<b>ASCII</b>			
...																
140h	<b>ASCII</b>								<b>UNIX Epoch time</b>				<b>1,000,000 means 1 second</b>			
150h	<i>ut_session</i>				<b>A positive number</b>				<b>0-9999999</b>				<i>ut_addr_v6</i>			
160h	<i>ut_addr_v6</i>												<u>00</u>	<u>00</u>	<u>00</u>	<u>00</u>
170h													<u>00 00 ... 00</u>			

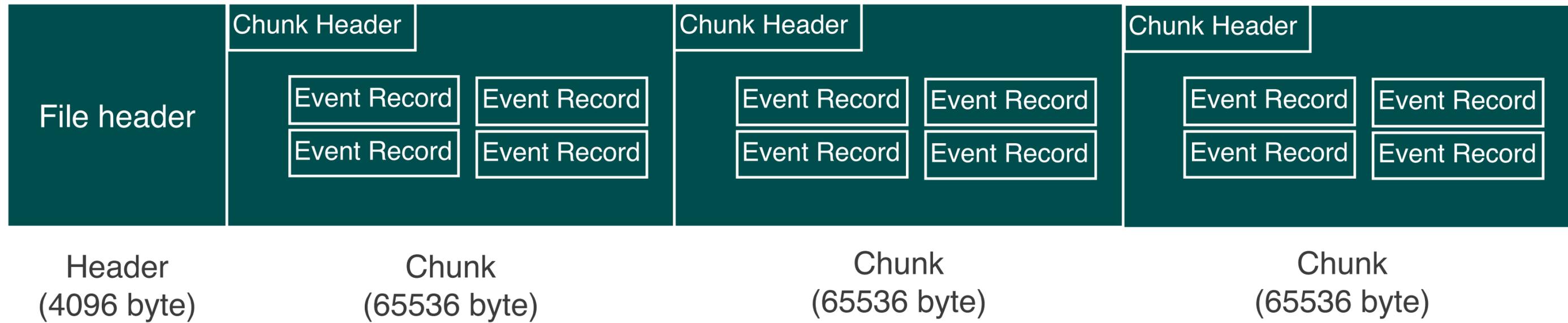
Based on actual record

UNIX Epoch time

1,000,000 means 1 second

Based on actual record

# Big Picture - EVTX -



# EVTX file header format [8]



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	<i>Signature</i>								<i>First chunk number</i>							
10h	Last chunk number								Next record identifier							
20h	<b><i>Header size</i></b>			<b><i>Minor version</i></b>		<b><i>Major version</i></b>		<i>Header block size</i>		<i>Number of chunks</i>		<b><i>Unknown</i></b>				
30h																
40h																
50h	Unknown															
60h	<b><i>Unknown</i></b>															
70h	Unknown															
	<b><i>Unknown</i></b>															
	Unknown								File flags				Checksum			
	<b><i>Unknown (Empty)</i></b>															

[8] Windows XML Event Log (EVTX) format

[https://github.com/libyal/libevtx/blob/master/documentation/Windows%20XML%20Event%20Log%20\(EVTX\).asciidoc](https://github.com/libyal/libevtx/blob/master/documentation/Windows%20XML%20Event%20Log%20(EVTX).asciidoc)



# EVTX Chunk header format [8]

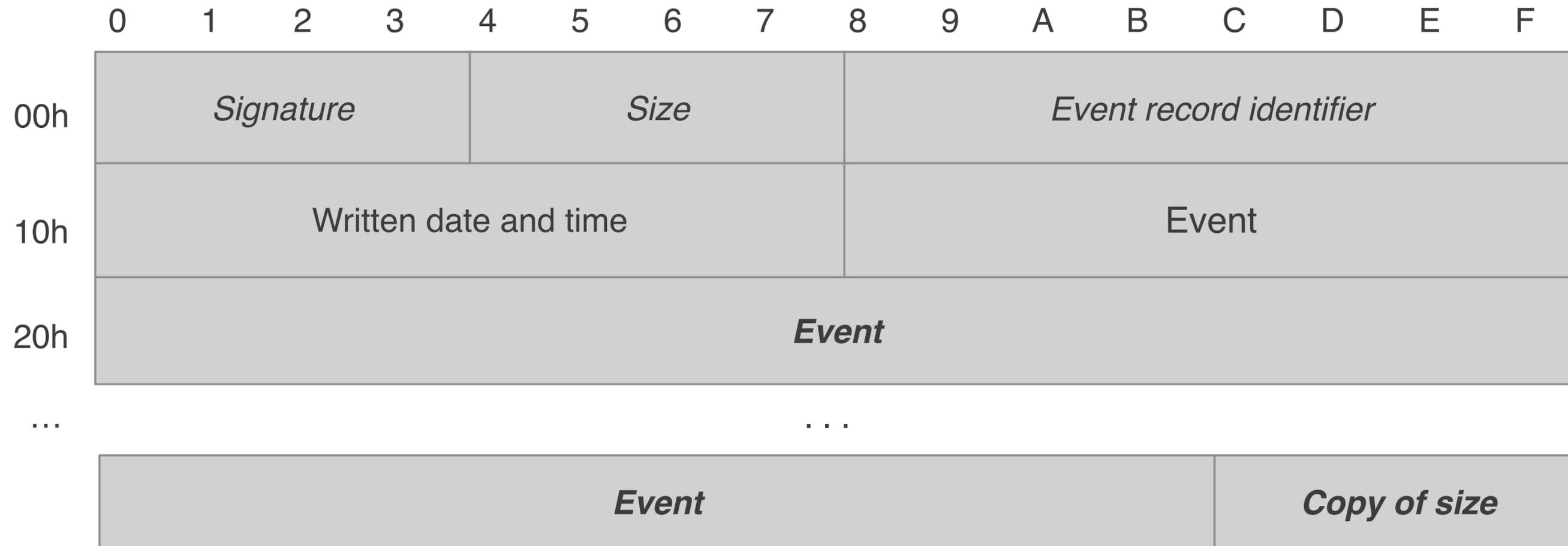
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	<i>Signature</i>								<i>First event record number</i>							
10h	Last event record number								First event record identifier							
20h	<b><i>Last event record identifier</i></b>								<i>Header size or offset</i>				<b><i>Last event record offset</i></b>			
30h	Free space offset				Event record checksum				Unknown							
40h	<b><i>Unknown</i></b>															
50h	Unknown															
60h	<b><i>Unknown</i></b>															
70h	Unknown								Unknown				Checksum			
	<b><i>Common string / Template</i></b>															

[8] Windows XML Event Log (EVTX) format

[https://github.com/libyal/libevtx/blob/master/documentation/Windows%20XML%20Event%20Log%20\(EVTX\).asciidoc](https://github.com/libyal/libevtx/blob/master/documentation/Windows%20XML%20Event%20Log%20(EVTX).asciidoc)



# EVTX Event record format [8]



[8] Windows XML Event Log (EVTX) format

[https://github.com/libyal/libevtx/blob/master/documentation/Windows%20XML%20Event%20Log%20\(EVTX\).asciidoc](https://github.com/libyal/libevtx/blob/master/documentation/Windows%20XML%20Event%20Log%20(EVTX).asciidoc)

# What Format to Focus On

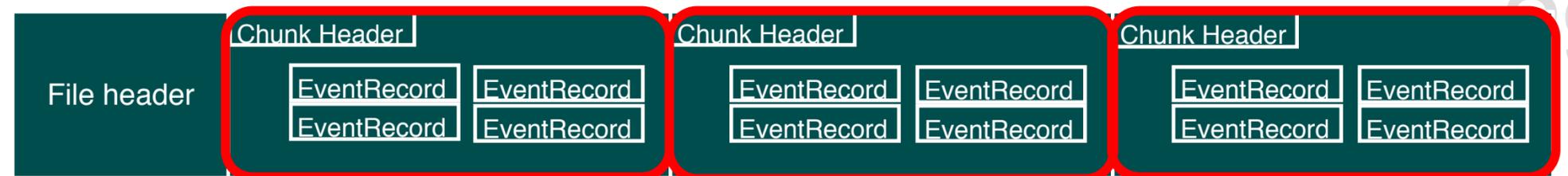
- **EVTX header** is just a header
- **EVTX chunk** keeps multiple event records
- **EVTX event record** can be carved out, but may be incomplete

[Important part]

- **A valid EVTX file can be generated from EVTX chunk header**  
(It enables us to reconstruct a file header from a chunk header)



So we focus on **EVTX chunk** carving



# Signature for EVTX Chunk

Unique characters

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	<u>E</u>	<u>I</u>	<u>f</u>	<u>C</u>	<u>h</u>	<u>n</u>	<u>k</u>	<u>00</u>	<i>First event record number</i>							
10h	<i>Last event record number</i>								<i>First event record identifier</i>							
20h	<i>Last event record identifier</i>								<u>128</u>		<i>Last event record</i>					
30h	<i>Free space offset</i>				<i>Event record checksum</i>				<i>Unknown</i>							
40h	<i>Unknown</i>															
50h	<i>Unknown</i>															
60h	<i>Unknown</i>															
70h	<i>Unknown</i>								<i>Unknown</i>				<i>Checksum</i>			
	<i>Common string / Template</i>															

It is easy to carve out because chunk size is 65,536 bytes

# Generating EVTX header

Some values take over from a chunk

	0	1	2	3	4	5	6	7	8	9	A	B	C
00h	<u>E</u>	<u>I</u>	<u>f</u>	<u>E</u>	<u>i</u>	<u>I</u>	<u>e</u>	<u>00</u>	<i>First chunk number</i>				
10h	<b>Last chunk number</b>							<b>Next record identifier</b>					
20h	<u><i>Header size</i></u>				<u><i>Minor version</i></u>		<u><i>Major version</i></u>		<i>Header block size</i>	<i>Number of chunks</i>		<u><i>Unknown</i></u>	
30h													
40h	<u>Unknown</u>												
50h	<u>Unknown</u>												
60h	<u>Unknown</u>												
70h	<u>Unknown</u>												
	<u>Unknown</u>								<u>File flags</u>			<u>Checksum</u>	

All other values we can create and set appropriate information

# Run record carving scanners

(For Windows)

```
> bulk_extractor -x all -e hiberfile -e ntfsindx -e ntfslogfile  
-e ntfsmft -e ntfsusn -e evtv -o output_dir input_deviceimage_file
```

(For Linux)

```
> bulk_extractor -x all -e gzip -e utmp -o output_dir input_deviceimage_file
```

DEMO



### 3. Intelligent Parser

# Parsing Big Picture

How do you find valuable information  
from raw data?

Artifact (Raw Data)

Convert



Filter



Search



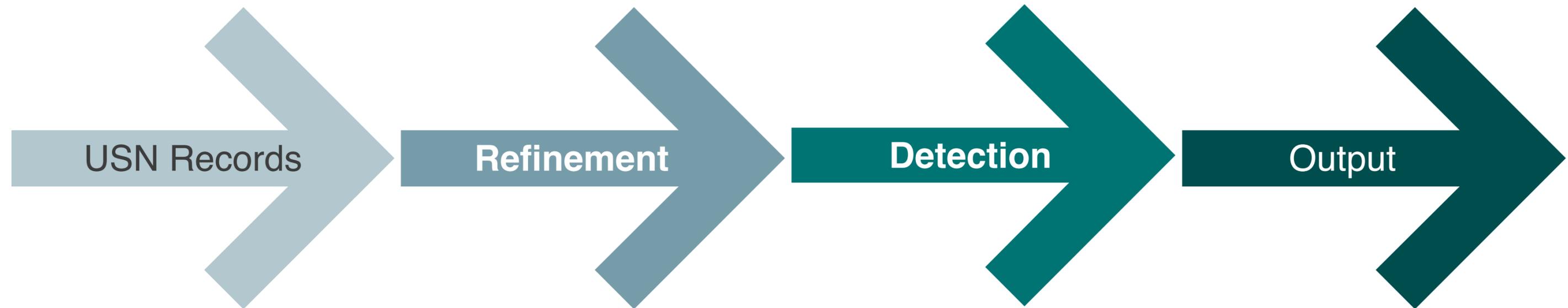
<b>Header</b>	<b>Header1</b>	<b>Header2</b>	<b>...</b>
<i>Record1</i>	<i>Column1</i>	<i>Column2</i>	<i>...</i>
<i>Record2</i>	<i>Column1</i>	<i>Column2</i>	<i>...</i>
<i>Record3</i>	<i>Column1</i>	<i>Column2</i>	<i>...</i>
<i>Record4</i>	<i>Column1</i>	<i>Column2</i>	<i>...</i>

# What is Intelligent Approach?

- Refinement
- Behavior Detection
- Link/Correlation

# USN Analytics

[https://www.kazamiya.net/en/usn\\_analytics](https://www.kazamiya.net/en/usn_analytics)



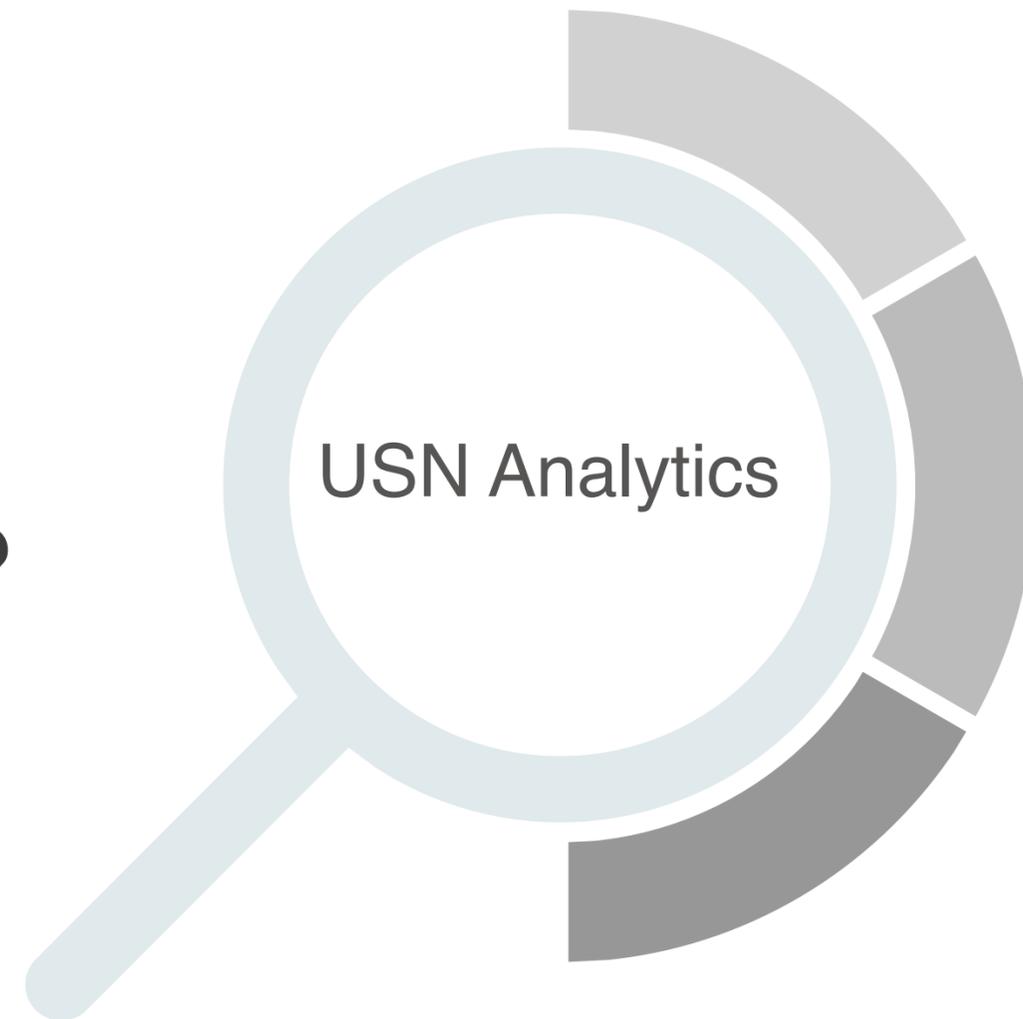
**Multi Platform**

**Refinement Records**

**Behavior Detection**

**and Open Source**

# What is Refinement?



1. Records Bundling
2. Path Reconstruction
3. Change Tracking

# 1. Records Bundling

If a file was written continuously, USN Record shows:

Timestamp	Name	FileID	ParentID	FileAtr	Reason
2018/10/17 12:34:56.789012	setupapi.dev.log	2468	1234	ARCHIVE	EXTEND
2018/10/17 12:34:56.789012	setupapi.dev.log	2468	1234	ARCHIVE	EXTEND   TRUNCATION
2018/10/17 12:34:56.789012	setupapi.dev.log	2468	1234	ARCHIVE	EXTEND   TRUNCATION
2018/10/17 12:34:57.012345	setupapi.dev.log	2468	1234	ARCHIVE	EXTEND
2018/10/17 12:34:57.012345	setupapi.dev.log	2468	1234	ARCHIVE	EXTEND   TRUNCATION
2018/10/17 12:34:57.012345	setupapi.dev.log	2468	1234	ARCHIVE	EXTEND   TRUNCATION

It is possible to bundle multiple records without information loss

Timestamp	TimeTaken	Count	Name	FileID	ParentID	FileAtr	Reason
2018/10/17 12:34:56.789012	0.223333	6	setupapi.dev.log	2468	1234	ARCHIVE	EXTEND   TRUNCATION   CLOSE

# 2. Path Reconstruction

USN Journal also holds information about a folder

Timestamp	Name	FileID	ParentID	FileAtr	Reason
2018/10/17 12:45:33.447152	SoftwareDistributio	3344	1112	DIRECTORY	CREATE   CLOSE
2018/10/17 12:45:33.447152	DataStore	3345	3344	DIRECTORY	CREATE   CLOSE
2018/10/17 12:45:33.447152	Logs	3346	3344	DIRECTORY	CREATE   CLOSE
2018/10/17 12:45:33.517636	Edbres00001.jrs	3369	3346	ARCHIVE	CREATE   EXTEND
2018/10/17 12:45:33.642436	DataStore.edb	3372	3345	ARCHIVE	CREATE   EXTEND

Folder table

Parent ID	ID	Name
1112	3344	SoftwareDistribution
3344	3345	DataStore
3344	3346	Logs

Folder Path List

ID	Name
3344	SoftwareDistribution\
3345	SoftwareDistribution\DataStore
3346	SoftwareDistribution\Logs

If Parent ID is found in "Folder Path List", add to Path information

Timestamp	Name	FileID	ParentID	FileAtr	Reason	Path
2018/10/17	SoftwareDistributio	3344	1112	DIRECTOR	CREATE   CLOSE	
2018/10/17	DataStore\	3345	3344	DIRECTOR	CREATE   CLOSE	SoftwareDistribution\
2018/10/17	Logs\	3346	3344	DIRECTOR	CREATE   CLOSE	SoftwareDistribution\
2018/10/17	Edbres00001.jrs	3369	3346	ARCHIVE	CREATE   EXTEND	SoftwareDistribution\Logs
2018/10/17 12:45:33.642436	DataStore.edb	3372	3345	ARCHIVE	CREATE   EXTEND   CLOSE	SoftwareDistribution\DataStore

# 3. Change Tracking

An operation of rename and move is recorded as reasons of OLD\_NAME and NEW\_NAME

Timestamp	Name	FileID	ParentID	FileAtr	Reason
2018/10/17 12:56:09.872451	<i>Summary.xml.tmp</i>	<i>3961</i>	<i>665</i>	<i>ARCHIVE</i>	<i>OLD_NAME</i>
2018/10/17 12:56:09.872451	<i>Summary.xml</i>	<i>3961</i>	<i>665</i>	<i>ARCHIVE</i>	<i>NEW_NAME</i>
2018/10/17 12:56:09.872451	<i>Summary.xml</i>	<i>3961</i>	<i>665</i>	<i>ARCHIVE</i>	<i>NEW_NAME   CLOSE</i>
2018/10/17 12:56:09.903651	<i>setup.exe</i>	<i>51234</i>	<i>474</i>	<i>ARCHIVE</i>	<i>OLD_NAME</i>
2018/10/17 12:56:09.903651	<i>setup.exe</i>	<i>51234</i>	<i>3288</i>	<i>ARCHIVE</i>	<i>NEW_NAME</i>
2018/10/17 12:56:09.903651	<i>setup.exe</i>	<i>51234</i>	<i>3288</i>	<i>ARCHIVE</i>	<i>NEW_NAME   CLOSE</i>

USN Analytics distinguishes between rename and move

Timestamp	Name	FileID	ParentID	FileAtr	Reason
2018/10/17 12:56:09.872451	<i>Summary.xml.tmp -&gt;</i>	<i>3961</i>	<i>665</i>	<i>ARCHIVE</i>	<i>RENAME</i>
2018/10/17 12:56:09.903651	<i>setup.exe (474 -&gt; 3288)</i>	<i>51234</i>	<i>474</i>	<i>ARCHIVE</i>	<i>MOVE</i>

# How does USN Analytics detect behavior?



4. Program Execution

5. File Open

6. Anomaly File

# 4. Program Execution

- An event of creation or modification of a prefetch file indicates execution
- The USN record provides us with program name (ExeName) and the number of execution (ExeCount)
- This approach has possibility to prove execution program even if corresponding prefetch file is deleted

Timestamp	ExeName	ExeCount	FileName	Reason
2018/10/17 13:02:14.102358	<i>whomai.exe</i>	<i>1</i>	<i>WHOAMI.EXE-B8288E39.pf</i>	<i>CREATE   EXTEND   CLOSE</i>
2018/10/17 13:02:14.130425	<i>cmd.exe</i>	<i>6</i>	<i>CMD.EXE-4A81B364.pf</i>	<i>EXTEND   TRUNC   CLOSE</i>
2018/10/17 13:03:42.797008	<i>cmd.exe</i>	<i>7</i>	<i>CMD.EXE-4A81B364.pf</i>	<i>EXTEND   TRUNC   CLOSE</i>
2018/10/17 13:03:52.658995	<i>reg.exe</i>	<i>1</i>	<i>REG.EXE-E7E8BD26.pf</i>	<i>CREATE   EXTEND   CLOSE</i>
2018/10/17 13:04:03.875327	<i>tasklist.exe</i>	<i>1</i>	<i>TASKLIST.EXE-C6CC193.pf</i>	<i>CREATE   EXTEND   CLOSE</i>
2018/10/17 13:04:22.334656	<i>net.exe</i>	<i>1</i>	<i>NET.EXE-DF44F913.pf</i>	<i>CREATE   EXTEND   CLOSE</i>

# 5. File Opening

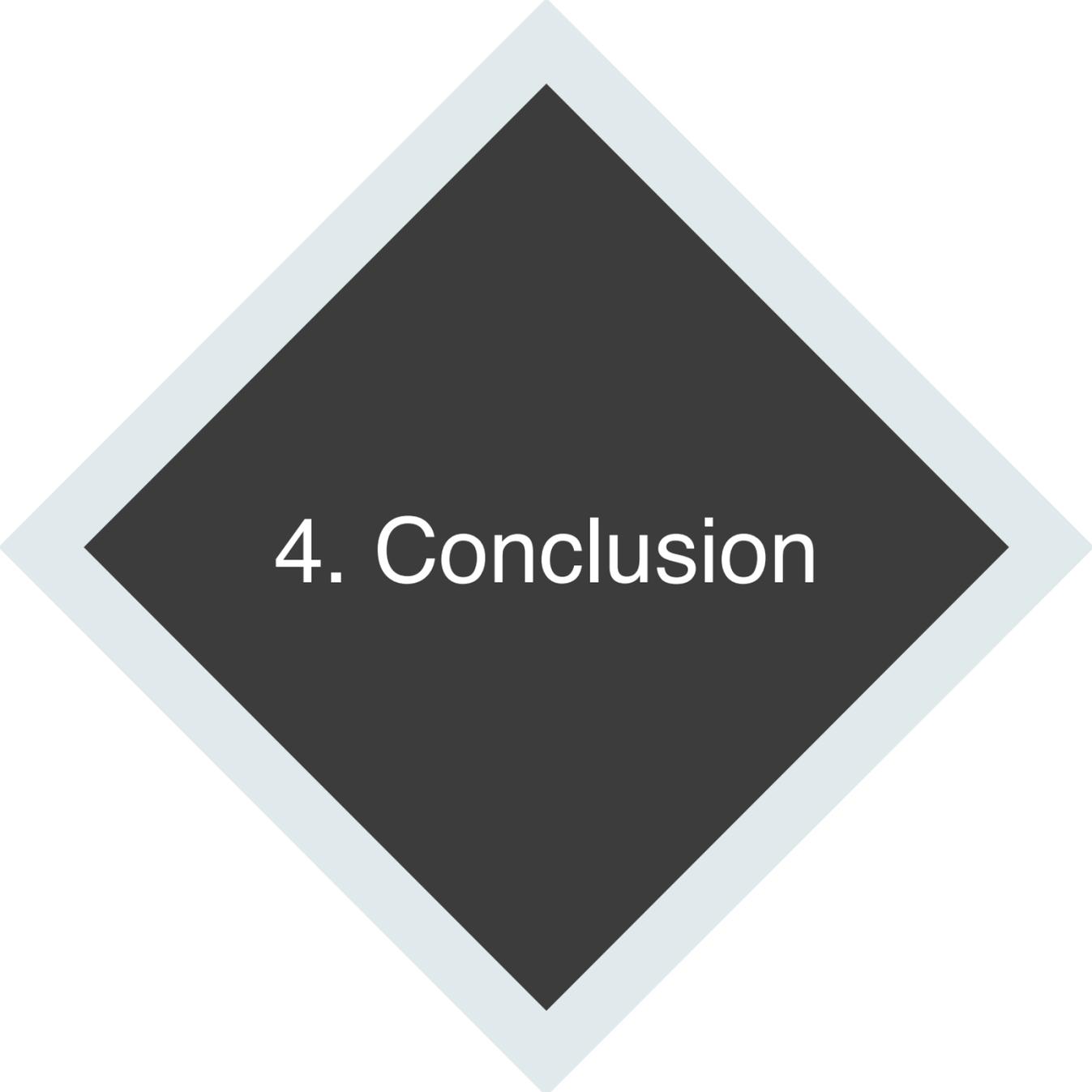
- The event of creation or modification of link file may indicates a user opened a file/folder
- USN record has reason of OBJECT\_ID\_CHANGED, it may also indicate a user opened a file/folder
- This approach has a possibility to prove opening files even if link file is deleted

Timestamp	Path	FileName	Reason
2018/10/17 13:18:32.802946	Desktop\	Notice.txt	OBJECTID   CLOSE
2018/10/17 13:18:53.650331	AppData\Roaming\Microsoft\Windows\Recent\	Notice.txt.Ink	CREATE   EXTEND
2018/10/17 13:22:17.379723	Documents\	Payment.docx	OBJECTID   CLOSE
2018/10/17 13:22:17.380724	AppData\Roaming\Microsoft\Windows\Recent\	Payment.docx.Ink	CREATE   EXTEND

# 6. Anomaly File

- Noteworthy filename extension:
  - job
  - scr
  - bat
  - vbe
  - tck
  - ps1
- Noteworthy filename:
  - PSEXESVC.exe
  - PAExec-hostname.exe

DEMO



## 4. Conclusion

# Key takeaways

## Advanced Carver

Bulk Extractor with Record Carving is appropriate for record carving

### Create Rules for Record Carving

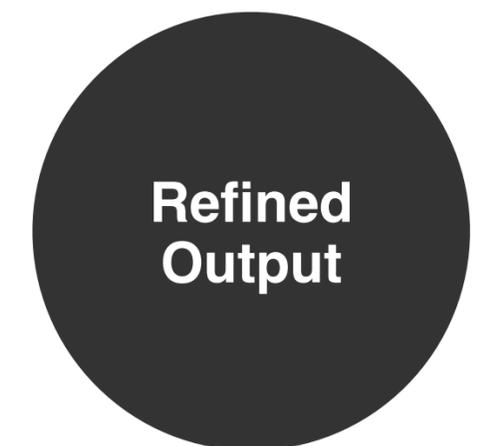
- To create rules, it is important to understand the data/record format
- Repeat trial and error

## Intelligent Parser

USN Analytics can refine output from USN journal

### Refinement

- Without information loss, it bundles multiple USN records
- Furthermore, it adds valuable information



# Next Steps

- Advanced Carver
  - Windows 8+ Hibernation Format
  - Windows 10 Memory compression
  - Additional scanners for record carving
- Intelligent Parser
  - Create more anomaly and behavior detection rules for USN
  - EVTX parser

# Thank you for your time and attention!

Any questions?

Bulk Extractor with Record Carving  
[https://www.kazamiya.net/en/bulk\\_extractor-rec](https://www.kazamiya.net/en/bulk_extractor-rec)

USN Analytics  
[https://www.kazamiya.net/en/usn\\_analytics](https://www.kazamiya.net/en/usn_analytics)

The 9th Annual Open Source Digital Forensics Conference

# References

[1] COMMON PHASES OF COMPUTER FORENSICS INVESTIGATION MODELS

<http://airccse.org/journal/jcsit/0611csit02.pdf>

[2] Carve for Records Not Files

<https://digital-forensics.sans.org/summit-archives/2012/carve-for-record-not-files.pdf>

[3] File Carving

[http://www.forensicswiki.org/wiki/File\\_Carving](http://www.forensicswiki.org/wiki/File_Carving)

[4] Forensic File Carving Tool Specification Version 1.0

<https://www.nist.gov/sites/default/files/documents/2017/05/09/fc-req-public-draft-01-of-ver-01.pdf>

[5] bulk\_extractor: A Stream-Based Forensics Tool

<https://www.osdfcon.org/presentations/2011/osdf-2011-garfinkel-bulk-extractor.pdf>

[6] USN\_RECORD\_V2 structure

[https://msdn.microsoft.com/ja-jp/library/windows/desktop/aa365722\(v=vs.85\).aspx](https://msdn.microsoft.com/ja-jp/library/windows/desktop/aa365722(v=vs.85).aspx)

[7] utmp(5) – Linux manual page

<http://man7.org/linux/man-pages/man5/utmp.5.html>

[8] Windows XML Event Log (EVTX) format

[https://github.com/libyal/libevt/blob/master/documentation/](https://github.com/libyal/libevt/blob/master/documentation/Windows%20XML%20Event%20Log%20(EVTX).asciidoc)

[Windows%20XML%20Event%20Log%20\(EVTX\).asciidoc](https://github.com/libyal/libevt/blob/master/documentation/Windows%20XML%20Event%20Log%20(EVTX).asciidoc)