Writing Autopsy Python Modules

Eugene Livis, *Basis Technology*

Why Did We Choose Python?

~~We were visionaries~~

Everyone was asking for it.

- It's an easy language to start using.
- Lots of other tools support it.

It was easy for us to integrate (Jython).

It was much easier than writing our own language!

Intended Takeaways

Autopsy is a good platform for writing Python scripts. Autopsy takes care of a lot of the infrastructure for you (UI, data sources, reporting, etc.)

It's easy to get started by copying a tutorial and modifying it

You should try it. All the cool kids are doing it.

Why Should You Write For Autopsy?

Developing forensics applications has three challenges:

1. **Input Types:** File systems, image formats, logical files, ZIP file contents, file carving, virtual machine contents, etc.
2. **User Interaction:** interfaces, reports, etc.
3. **Analytics:** Finding a certain file, parsing its contents, etc.

Autopsy takes care of #1 & #2. Allowing you to focus on #3.

Writing An Autopsy Module

4 Basic Steps

1. Pick your module type.
2. Find the closest Autopsy template or tutorial to copy.
3. Search for the word “TODO” and put in your own names, etc.
4. Write your analytics in the “analysis method”.

Step #1: Pick Your Module Type

Ingest Modules: Analyze content in a data source after it is added to a case (most common).

- Hash calculation and lookup
- Keyword search
- EXIF
- ZIP files

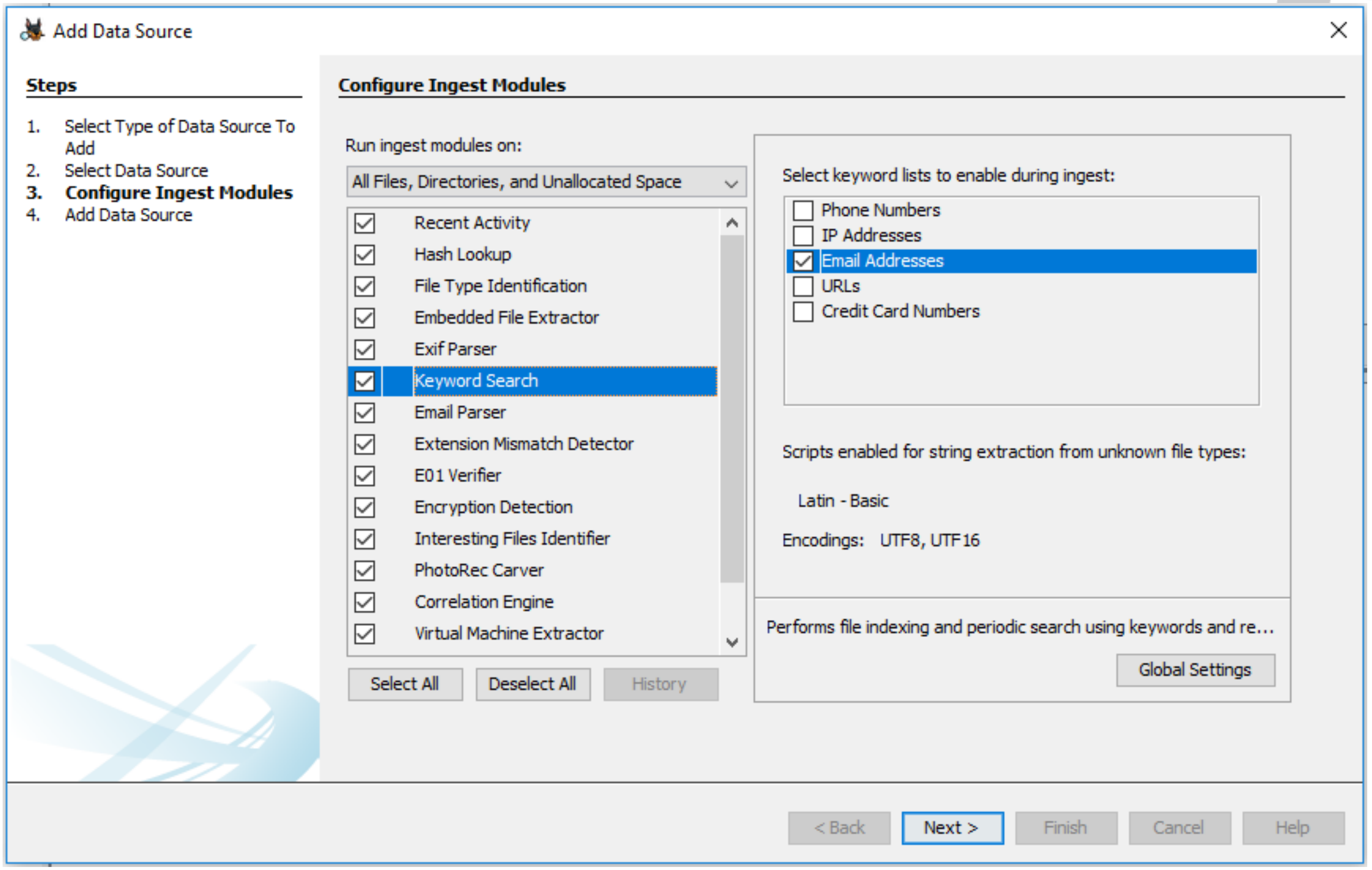
Report Modules: Run after all analysis is complete to create an output report.

- HTML
- XML
- CSV

The other Autopsy modules are currently Java only.

Ingest Modules

Analyze content in a data source after it is added to a case.



Run ingest modules on: All Files, Directories, and Unallocated Space

- [x] Recent Activity
- [x] Hash Lookup
- [x] File Type Identification
- [x] Embedded File Extractor
- [x] Exif Parser
- [x] Keyword Search
- [x] Email Parser
- [x] Extension Mismatch Detector
- [x] E01 Verifier
- [x] Encryption Detection
- [x] Interesting Files Identifier
- [x] PhotoRec Carver
- [x] Correlation Engine
- [x] Virtual Machine Extractor

Select keyword lists to enable during ingest:

- [ ] Phone Numbers
- [ ] IP Addresses
- [x] Email Addresses
- [ ] URLs
- [ ] Credit Card Numbers

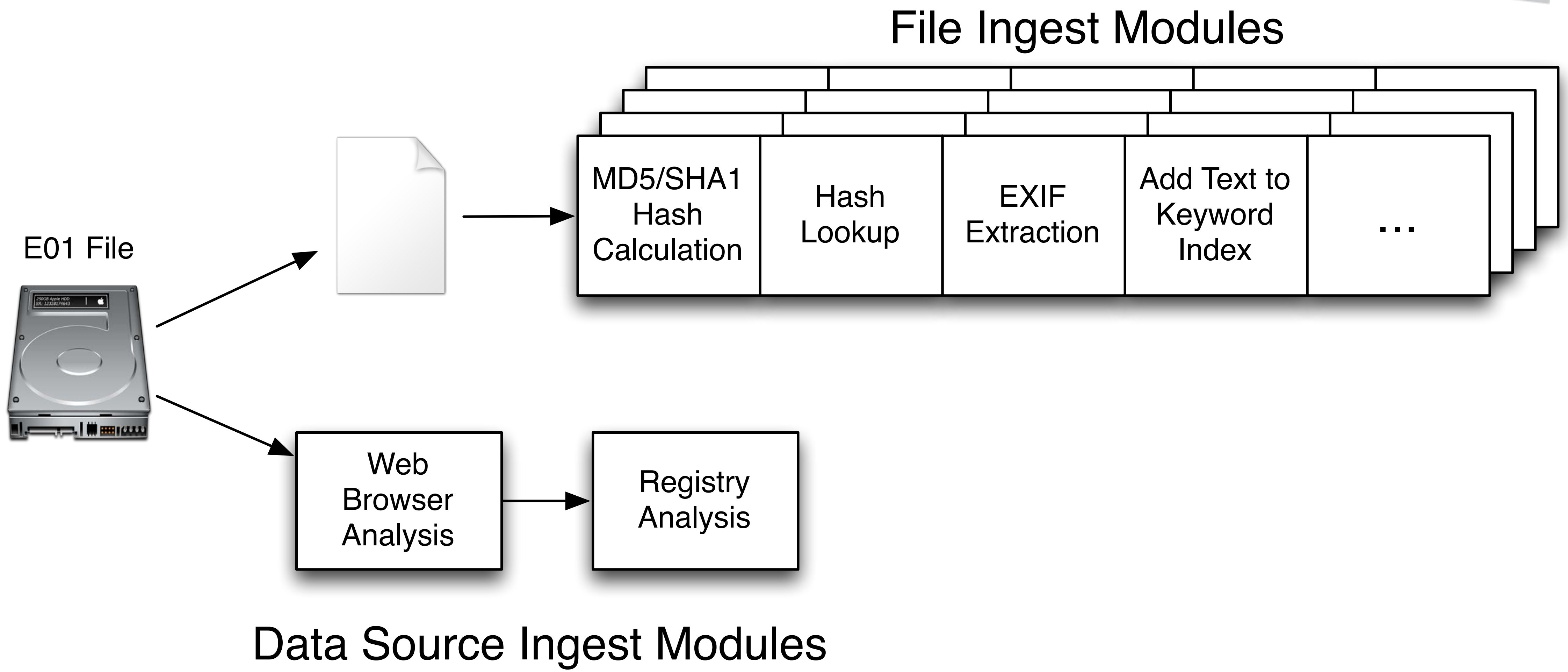
Scripts enabled for string extraction from unknown file types:

Latin - Basic

Encodings: UTF8, UTF16

Performs file indexing and periodic search using keywords and re...

Types of Ingest Modules



File Ingest Modules

Are passed in a reference to each file in the data source.

- Includes files inside of ZIP files, carved files, files inside Virtual Machines, etc.

These are the easiest to write, but not efficient for all uses.

Example uses:

- Hash calculation and lookup
- File type identification
- Keyword indexing

Data Source Ingest Modules

Are passed in a reference to the full data source.

It is up to the module to find the files that are relevant by querying the backend database.

May run before all ZIP files are opened.

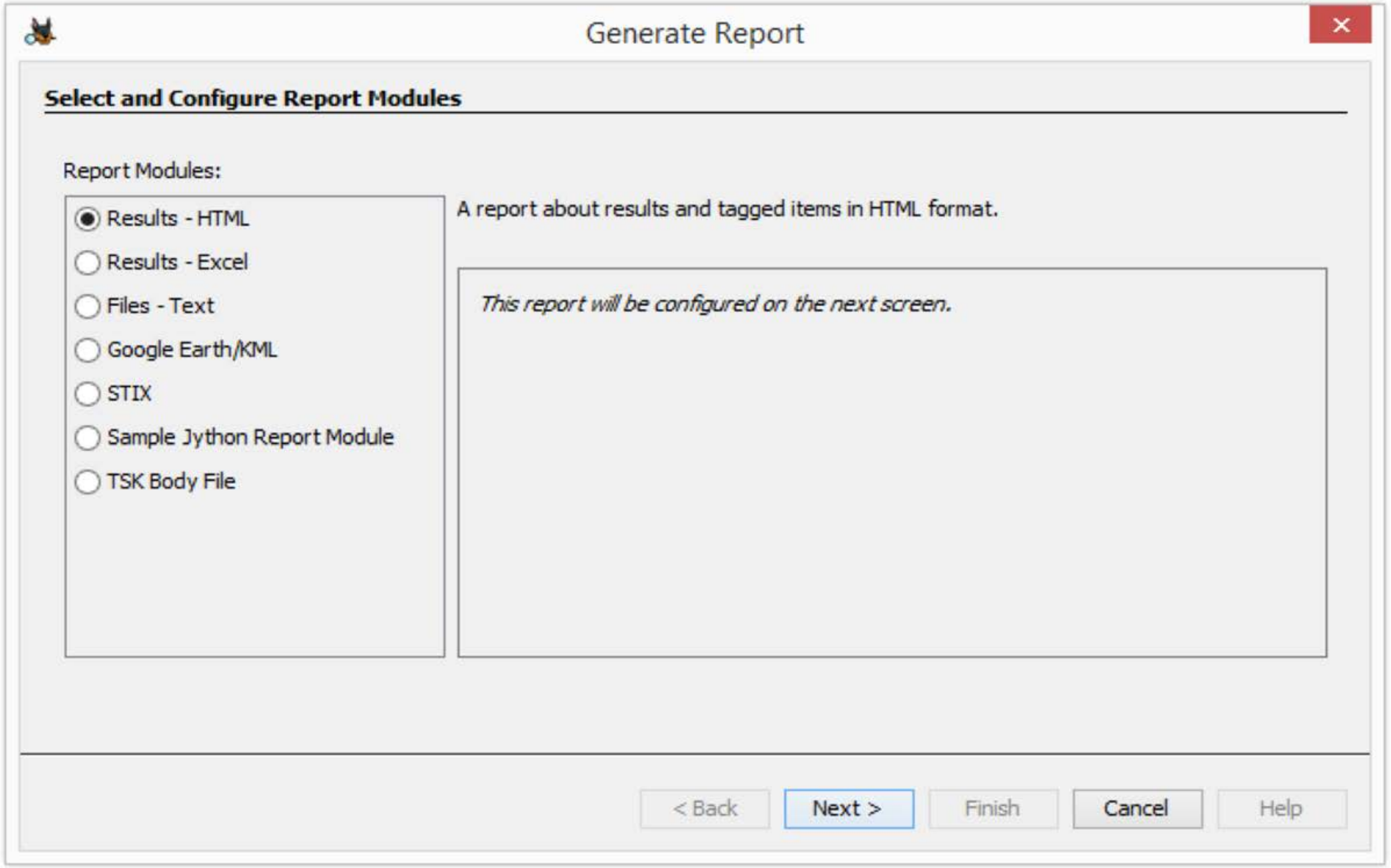
These are often used when we know where the file will be or we are calling an external tool.

Examples:

- Web analytics where we know the name and path of the databases.
- Registry analysis where we know the path of the hives.

Report Modules

Run after all analysis is complete to create an output report.



Select and Configure Report Modules

Report Modules:

- [✓ Results - HTML]
- Results - Excel
- Files - Text
- Google Earth/KML
- STIX
- Sample Jython Report Module
- TSK Body File

A report about results and tagged items in HTML format.

This report will be configured on the next screen.

Summary of Python Module Options

Pick the type based on your analysis needs.

Do you need to see every file?

Do you know the name of the files you want?

Do you want to run after everything has been run?

Step #2: Find Something to Borrow

Find the closest tutorial:

- File Ingest Module: Flag files based on size.
- Data Source Ingest Modules:
  - Find SQLite databases and parse them.
  - Run a command line tool on a disk image.
- Report Module: Create CSV report.

Review code in the templates on github:

- https://github.com/sleuthkit/autopsy/tree/develop/pythonExamples

Step #3: Search for “TODO”

Adapt the templates to you

```
# TODO: give it a unique name.  Will be shown in module
list
moduleName = "Sample File Ingest Module”
```

Step #4: Write the “Analysis Method”

Each module type has a method that does the analytics.

For example, File Ingest Modules have a method named “process” that is passed in a file to analyze.

```
    def process(self, file):
```

It is defined in the template you copied.

You write the steps in the method to do whatever you want.

Step #4: Publish to User

You need to get your results to the user somehow.

Two common ways:

1. Lazy: Save output to a file and add file as a “Report”.
2. Better: Create an artifact and post it to the blackboard.

   ARTIFACT: WEB_BOOKMARK

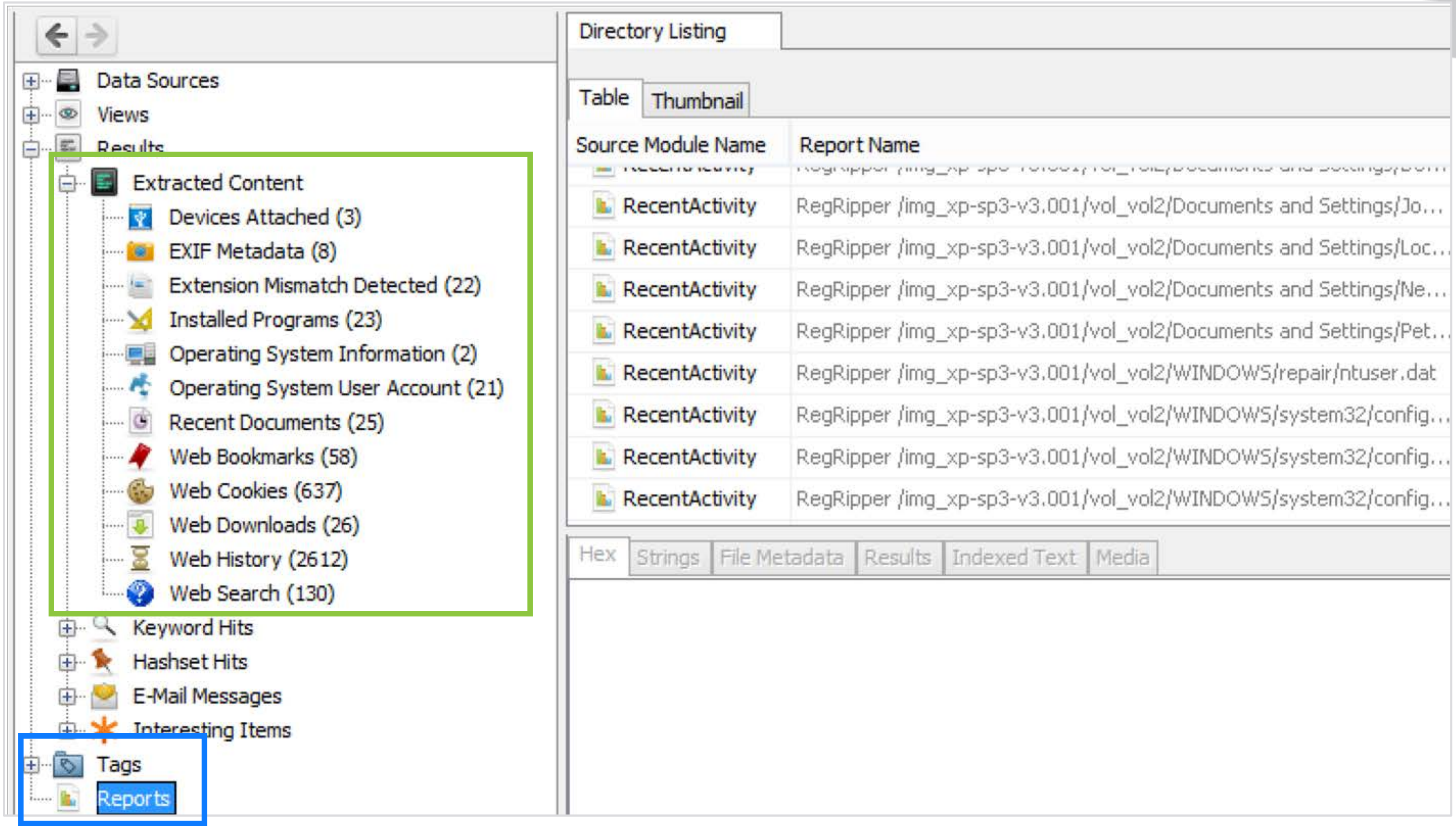
   - URL: http://www.sleuthkit.org/
   - DATE: October 17, 2018

Artifacts and reports are both shown in the tree.

Seeing The Results

Artifacts

Reports



Source Module Name	Report Name
RecentActivity	RegRipper /img_xp-sp3-v3.001/vol_vol2/Documents and Settings/Jo...
RecentActivity	RegRipper /img_xp-sp3-v3.001/vol_vol2/Documents and Settings/Loc...
RecentActivity	RegRipper /img_xp-sp3-v3.001/vol_vol2/Documents and Settings/Ne...
RecentActivity	RegRipper /img_xp-sp3-v3.001/vol_vol2/Documents and Settings/Pet...
RecentActivity	RegRipper /img_xp-sp3-v3.001/vol_vol2/WINDOWS/repair/ntuser.dat
RecentActivity	RegRipper /img_xp-sp3-v3.001/vol_vol2/WINDOWS/system32/config...
RecentActivity	RegRipper /img_xp-sp3-v3.001/vol_vol2/WINDOWS/system32/config...
RecentActivity	RegRipper /img_xp-sp3-v3.001/vol_vol2/WINDOWS/system32/config...

Blackboard Artifacts

All artifacts have a type:

- Web Bookmark
- Call Log
- Hashset Hit

Autopsy / The Sleuth Kit define 30+ of them.

You can define your own.

All artifacts are associated with a file.

- Web bookmarks are associated with file they were parsed from.

Blackboard Attributes

Artifacts have attributes, which are name and value pairs:

- URL: http://www.sleuthkit.org
- SET_NAME: Bad Pictures

Autopsy / TSK define 100+ attribute types defined.

Values can be strings or integers.

Example:

- Artifact: WEB_HISTORY
- Attributes:
  - URL: http://www.sleuthkit.org
  - DATE: April 1, 2015
  - PROGRAM: Firefox

Ingest Module Summary

1. Copy sample module
2. Edit class and display names
3. Edit the logic in the process() method

That’s it!

You don’t care where data is coming from and it will appear in the UI and reports.

Example: Find big and round files

July ‘15 Tutorial on www.basistech.com

Big and round files:

- Bigger than 10MB and multiple of 4096 bytes
- Could be encrypted volumes

Step #1: Pick the module type

- We want to look at all files, even ZIP file contents.
- File Ingest Module.

Steps #2 and #3: Copy the file ingest template and update its name, etc.

Find big and round files (contd.)

Step #4: Write the analysis logic:

- Check the size of each file
- If it is big and round, flag it

Recall that file-level Ingest Modules are passed in a file:

```
def process(self, file):
```

We check the size of the file:

```
if ((file.getSize() > 10000000) and ((file.getSize() % 4096) == 0)):
    # YEA!!!, do something with it
else:
    return OK
```

Let’s Tell The World About It!

We’re going to make an “Interesting File” artifact

```
art = file.newArtifact(TSK_INTERESTING_FILE_HIT)
att = BlackboardAttribute(TSK_SET_NAME, "Big and Round
Files")
art.addAttribute(att)
```

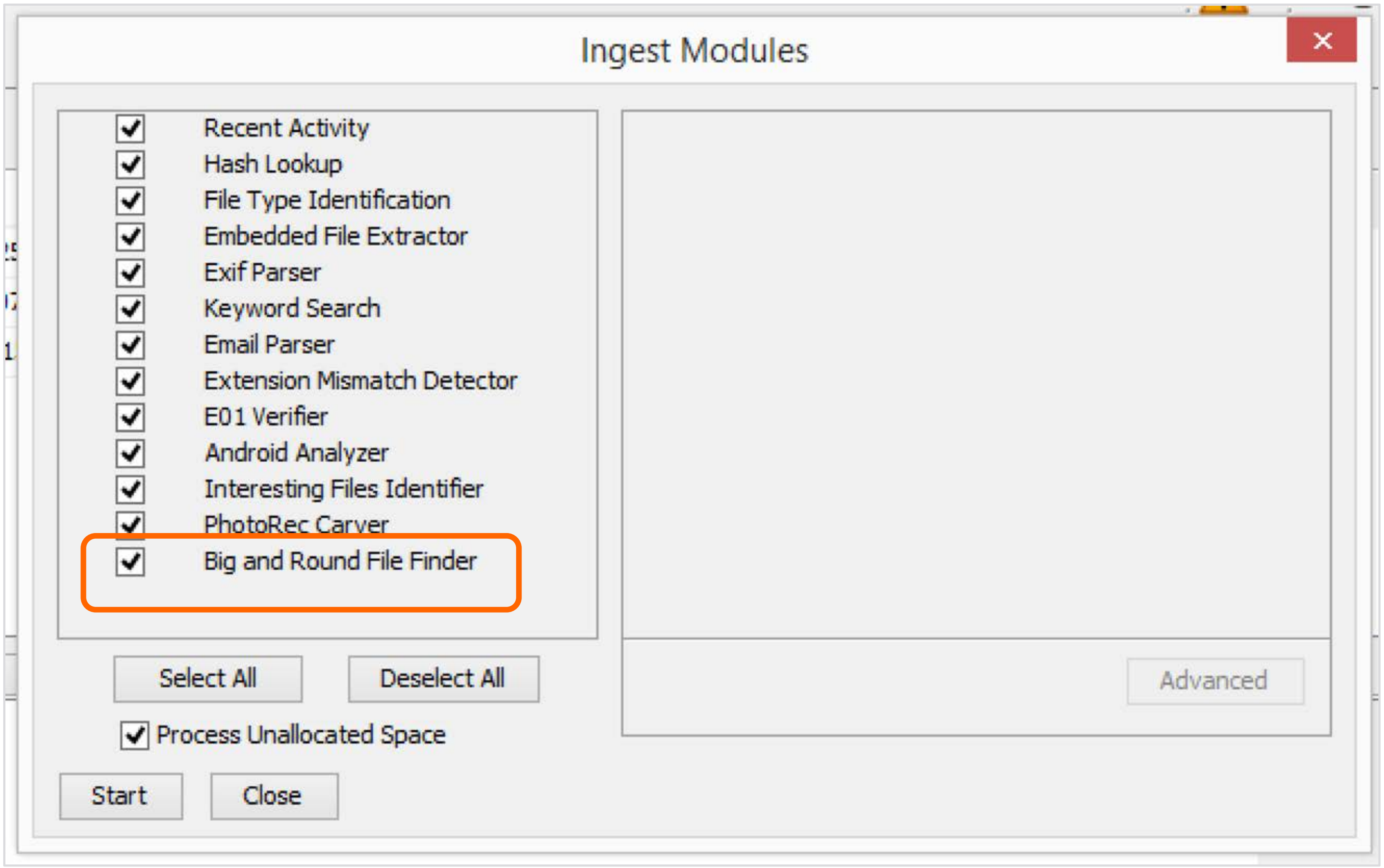
Final Method

```
def process(self, file):
    if ((file.getSize() > 10000000) and ((file.getSize() % 4096) == 0)):
        art = file.newArtifact(TSK_INTERESTING_FILE_HIT)
        att = BlackboardAttribute(TSK_SET_NAME, "Big and Round Files")
        art.addAttribute(att)
    return OK
```

This will find files in all file systems, archive files, carved files, virtual machine contents, etc.

This provides easy feedback to the user.

How the User Uses It

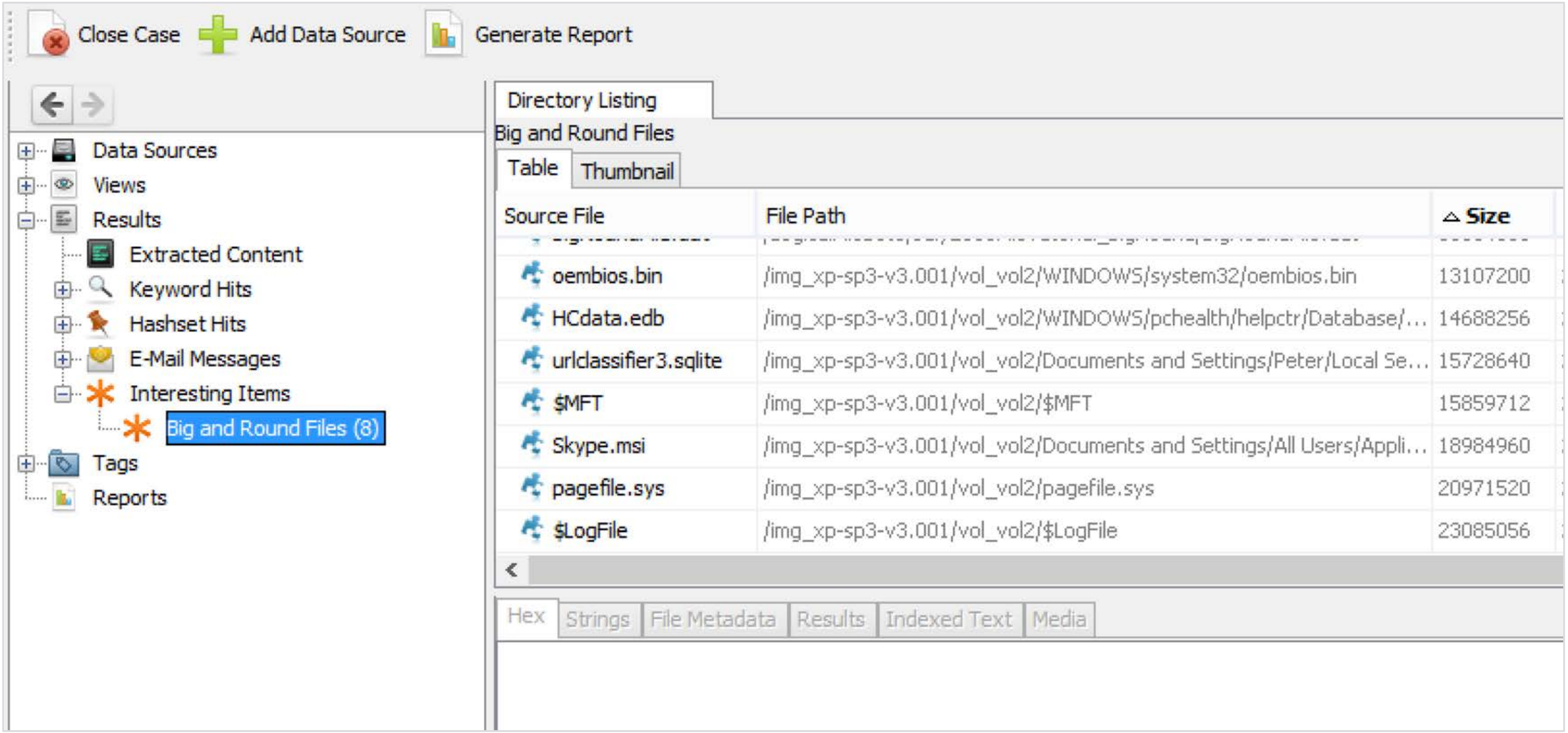


Ingest Modules

- [✓] Recent Activity
- [✓] Hash Lookup
- [✓] File Type Identification
- [✓] Embedded File Extractor
- [✓] Exif Parser
- [✓] Keyword Search
- [✓] Email Parser
- [✓] Extension Mismatch Detector
- [✓] E01 Verifier
- [✓] Android Analyzer
- [✓] Interesting Files Identifier
- [✓] PhotoRec Carver
- [✓] Big and Round File Finder

- [✓] Process Unallocated Space

How the User Sees the Results


Close Case · Add Data Source · Generate Report

- Data Sources
- Views
- Results
  - Extracted Content
  - Keyword Hits
  - Hashset Hits
  - E-Mail Messages
  - Interesting Items
    - Big and Round Files (8)
- Tags
- Reports

Directory Listing

Big and Round Files

Table | Thumbnail

Source File	File Path	Size
oembios.bin	/img_xp-sp3-v3.001/vol_vol2/WINDOWS/system32/oembios.bin	13107200
HCdata.edb	/img_xp-sp3-v3.001/vol_vol2/WINDOWS/pchealth/helpctr/Database/...	14688256
urlclassifier3.sqlite	/img_xp-sp3-v3.001/vol_vol2/Documents and Settings/Peter/Local Se...	15728640
$MFT	/img_xp-sp3-v3.001/vol_vol2/$MFT	15859712
Skype.msi	/img_xp-sp3-v3.001/vol_vol2/Documents and Settings/All Users/Appli...	18984960
pagefile.sys	/img_xp-sp3-v3.001/vol_vol2/pagefile.sys	20971520
$LogFile	/img_xp-sp3-v3.001/vol_vol2/$LogFile	23085056

Hex | Strings | File Metadata | Results | Indexed Text | Media

Conclusion

It’s easy to get started with writing Python modules for Autopsy.

Autopsy does all of the infrastructure work for you.

Contact Information

Eugene Livis

elivis@basistech.com

617-386-2000