6th Annual
#OSDFCon
OPEN SOURCE DIGITAL FORENSICS CONFERENCE

BASIS TECHNOLOGY

Python Autopsy: A Quick Introduction to Scripting Autopsy

Brian Carrier

Basis Technology

October 28, 2015 | Westin Washington Dulles, Herndon, VA

Intended Takeaways

- Python is a good language to learn.
- Autopsy is a good platform for writing Python scripts.
- You should try it. All the cool kids are doing it.

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!

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, 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.

Background: Very High-level Programming Concepts

Variables and Classes

- **Variable**: A name for some value. Think Algebra.

  $A^2 + B^2 = C^2$

  A, B, and C are variables.

  In Python: `fileName = "badfile.exe"`

- **Class**: A collection of data.
  - A "File" class would have data for its name, size, times etc.
  - You can get the data from the class:

```
fileName = file.getName()
```

Methods

- **Method**: A set of instructions with a name

```
def openDoor():
    extend arm to doorknob
    grab doorknob with hand
    turn doorknob clockwise
    push door
    let go of doorknob
```

- Methods can then be called in a single line:

```
openDoor()
```

Methods with Arguments

- You can pass in information to the method via an argument

```
def openDoor(direction):
    extend arm to doorknob
    grab doorknob with hand
    turn doorknob direction
    push door
    let go of doorknob
```

- Specify the arguments in each call

```
openDoor(“clockwise”)
openDoor(“counter clockwise”)
```

Writing An Autopsy Module

4 Basic Steps

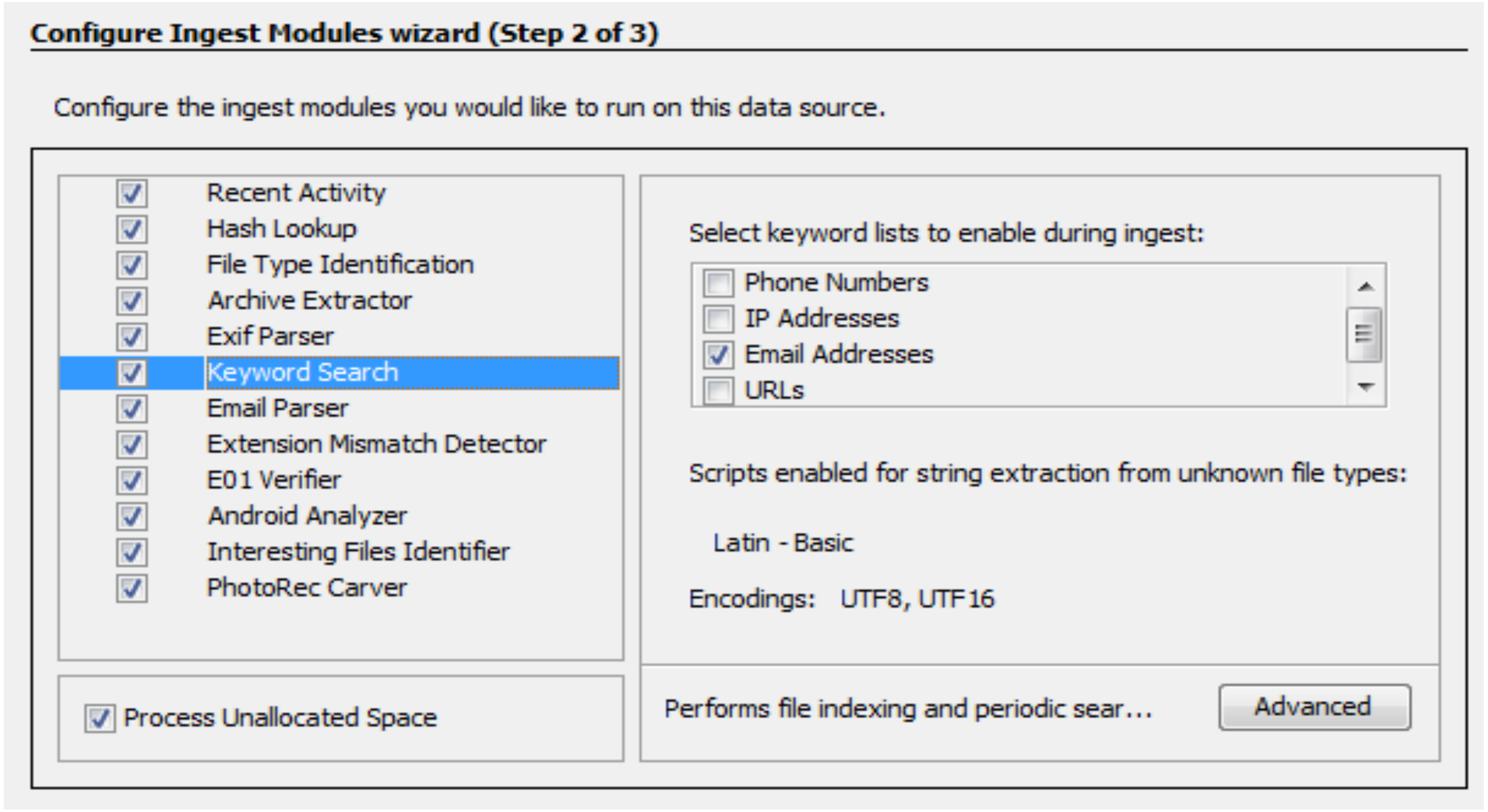
1. Pick your module type.
2. Find the closest Autopsy template or tutorial to copy.
3. Search for the word “TODO” and put in your own names, etc.
4. Write your analytics in the “analysis method”.

Step #1: Pick Your Module Type

- There are 8 module types in Autopsy.
- Only 3 of which can be written in Python though.

Ingest Modules

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



Configure Ingest Modules wizard (Step 2 of 3)

Configure the ingest modules you would like to run on this data source.

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

[✓] Process Unallocated Space

Select keyword lists to enable during ingest:

- [ ] Phone Numbers
- [ ] IP Addresses
- [✓] Email Addresses
- [ ] URLs

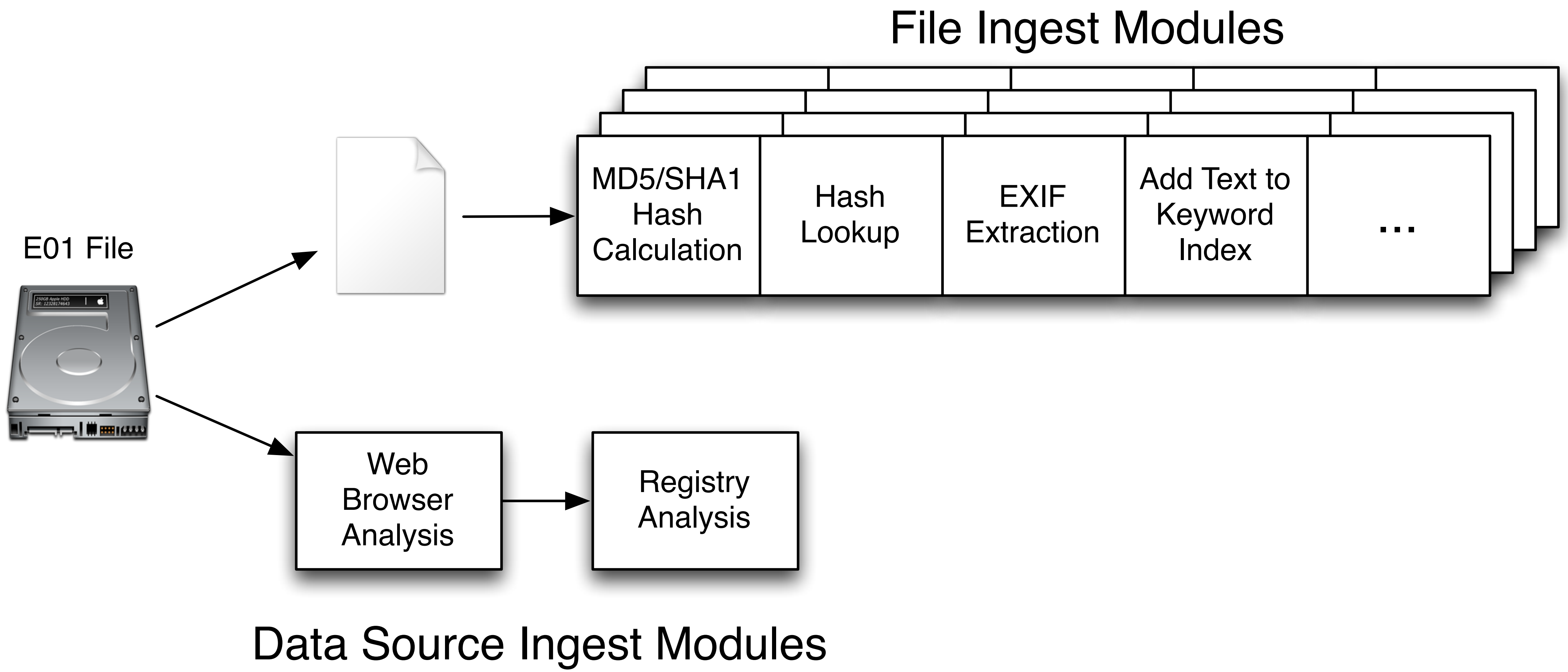
Scripts enabled for string extraction from unknown file types:

Latin - Basic

Encodings: UTF8, UTF16

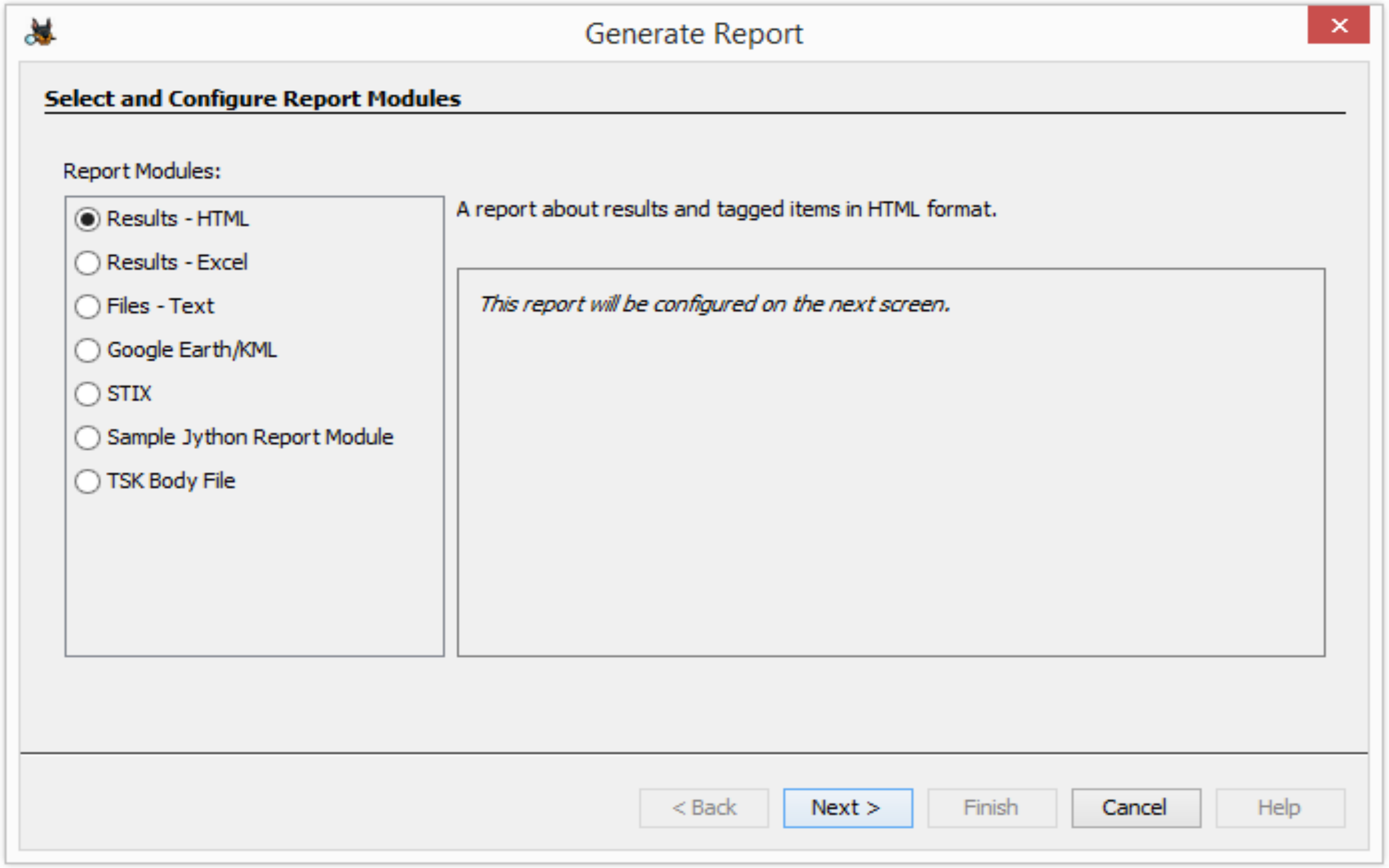
Performs file indexing and periodic sear...

Types of Ingest Modules



Report Modules

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


Generate 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.

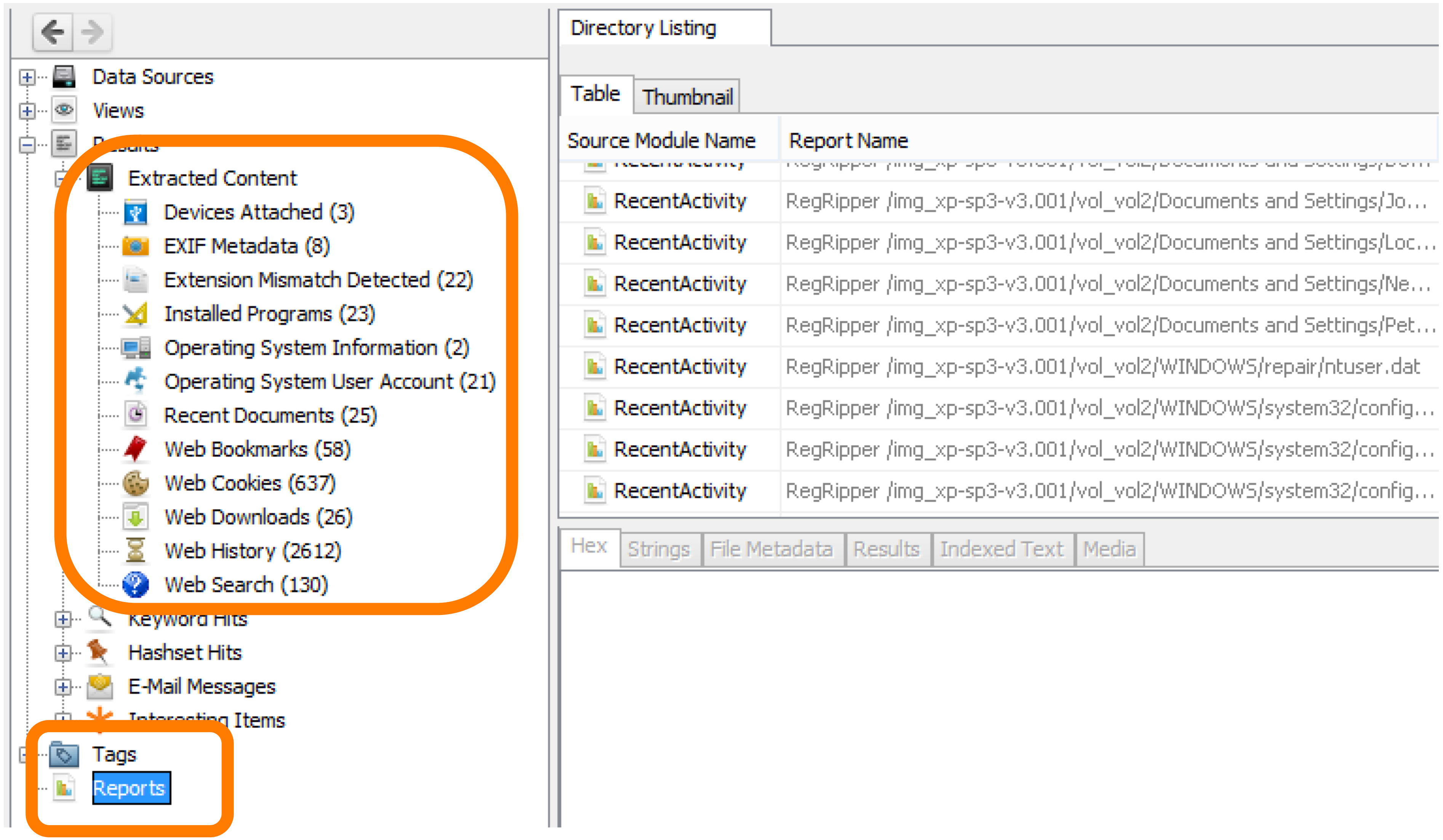
Step #4: Publish to User

- 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 28, 2015
- Artifacts and reports are both shown in the tree.

Seeing The Results

Artifacts

Reports



- Data Sources
- Views
- Results
  - Extracted Content
    - Devices Attached (3)
    - EXIF Metadata (8)
    - Extension Mismatch Detected (22)
    - Installed Programs (23)
    - Operating System Information (2)
    - Operating System User Account (21)
    - Recent Documents (25)
    - Web Bookmarks (58)
    - Web Cookies (637)
    - Web Downloads (26)
    - Web History (2612)
    - Web Search (130)
  - Keyword Hits
  - Hashset Hits
  - E-Mail Messages
  - Interesting Items
- Tags
- Reports

Directory Listing

Table | Thumbnail

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...

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 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

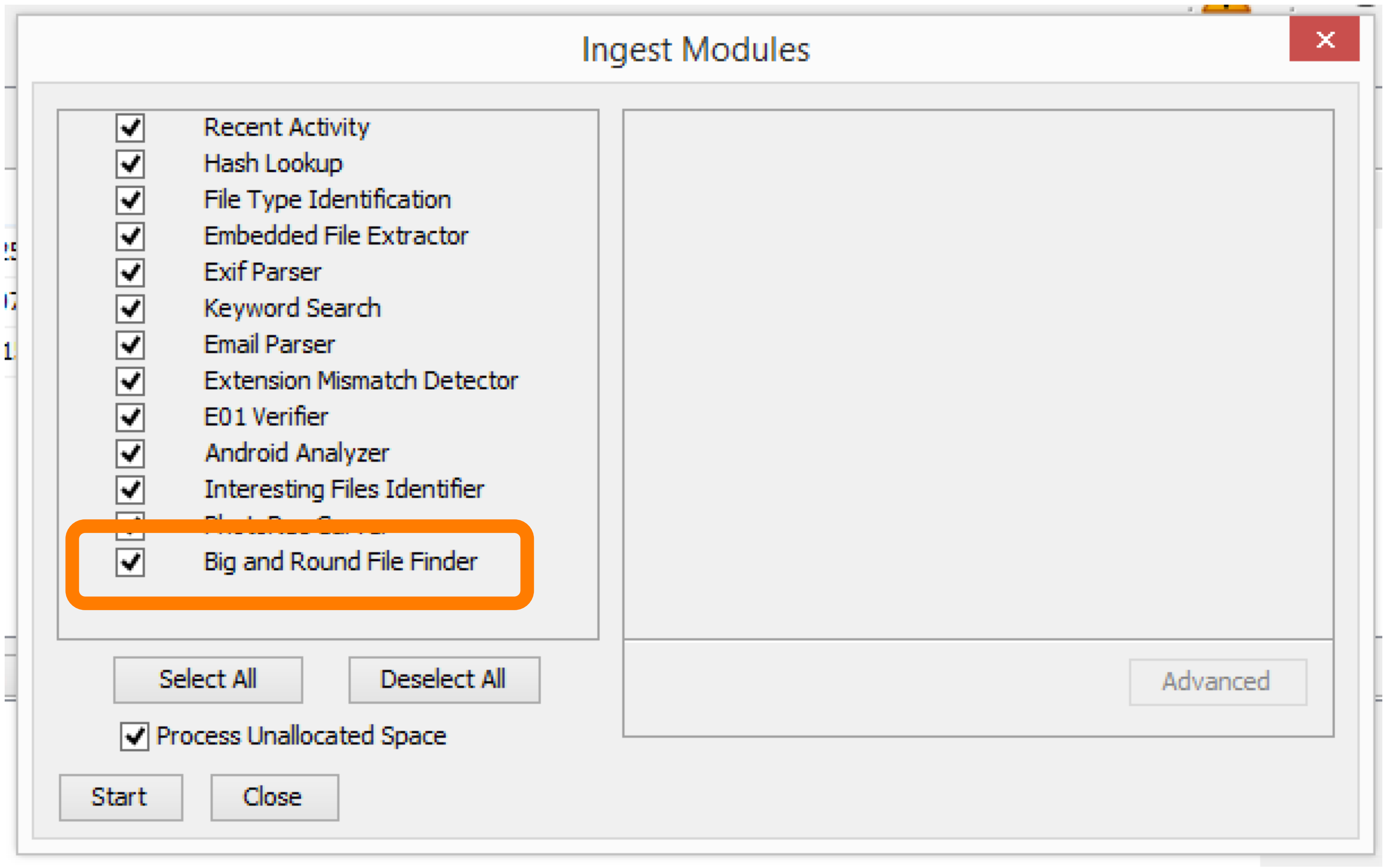
```
art = file.newArtifact(TSK_INTERESTING_FILE_HIT)

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, compound files, carved files, etc.
- This provides easy feedback to the user.

How the User Uses It

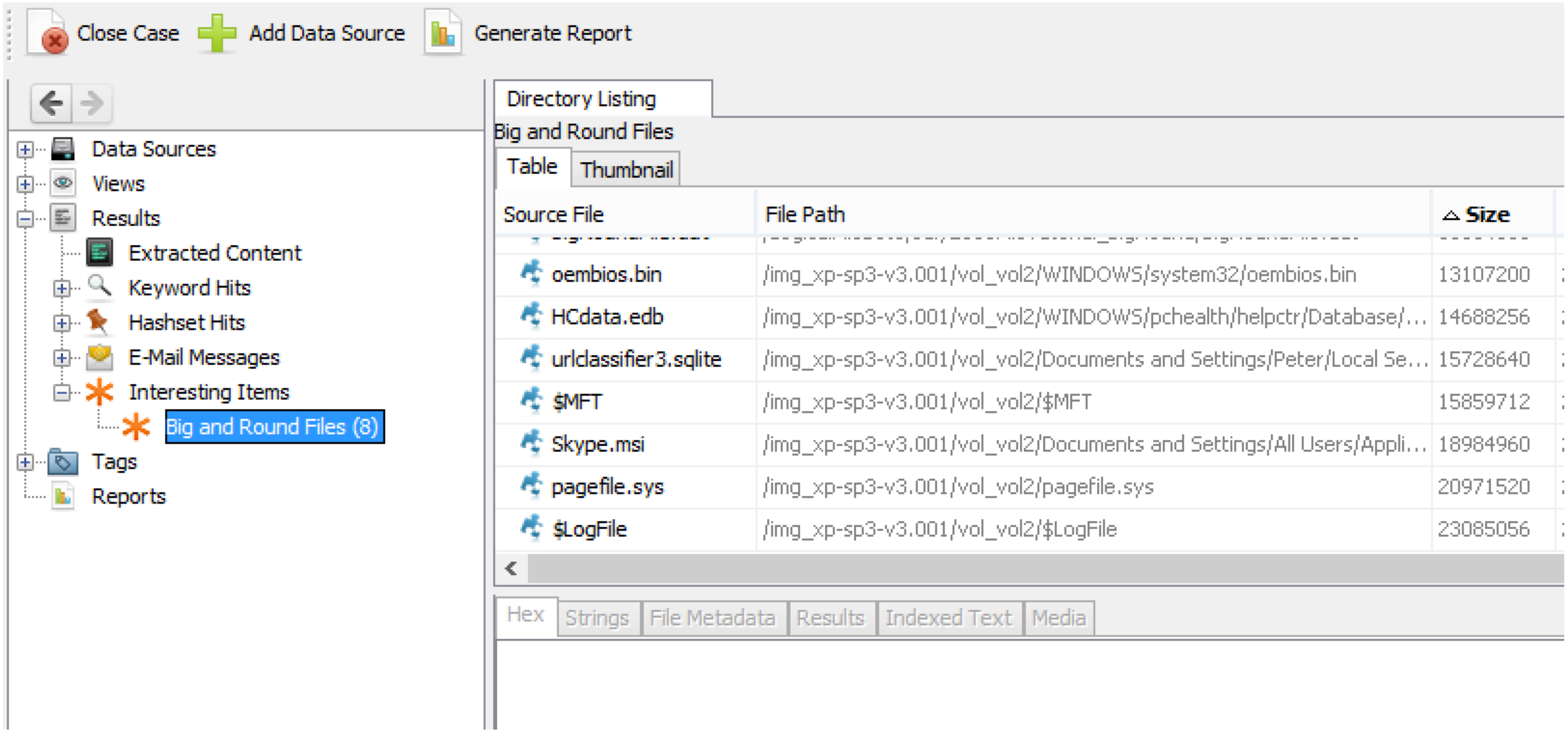


Ingest Modules

- [✓] 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

Brian Carrier
brianc@basistech.com
617-386-2000