GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

© 2015 SANS Institute, Author retains full rights.

DFIR Analysis and Reporting Improvements with Scientific Notebook Software

GIAC (GCIH) Gold Certification

Author: Ben S. Knowles, adric@adric.net
Advisor: Manuel Humberto Santander Pelaez, Manuel.Santander@epm.com.co

Draft: March 8, 2015

Abstract

Free and open source scientific notebook software allows responders to perform analysis and record results simultaneously in an open, flexible, portable format for ease of sharing and reporting. Fully worked samples can improve analyst and responder mentoring and education. Use of notebook templates can encourage good practices, uphold standards, and improve investigative rigor for better DFIR science and better incident response. Suggested configuration options and server platform notes for SIFT3 explain notebook setup for forensics. The proposed workflow and methodology show how DFIR process and techniques are integrated into notebooks and the SIFT server environment and a walk through a sample investigation with notebooks illustrates the advantages.

© 2015 The SANS Institute Author retains full rights.

1. Introduction

1.1. Digital Forensics

Digital forensics is defined in the US National Institute of Standards and Technology (NIST) *Guide to Integrating Forensic Techniques into Incident Response* as "the application of science to the identification, collection, examination, and analysis of data while preserving the integrity of the information and maintaining a strict chain of custody for the data" (NIST, 2006).

In the introductory chapter of his seminal 2005 *File System Forensic Analysis* Brian Carrier explains his use of some key terms in a section titled "Digital Investigations and Evidence". He defines digital investigations and digital evidence in relation to computer investigation techniques and forensics science and distinguishes use of these techniques for forensics purposes specifically as digital forensics investigations (DFI) to delineate them from other more general digital investigations and better align them with related fields of law-related forensic techniques such as physical crime scene investigations and criminalistics. Carrier explains, "A digital forensics investigation is a process that uses science and technology to analyze digital objects and that develops and tests theories, which can be entered into a court of law, to answer questions about events that occurred" (Carrier, 2005).

1.2. Incident Response

In The SANS Institute *Hacker Tools, Techniques, Exploits and Incident Handling* course (SEC504) authors Ed Skoudis and John Strand define incident handling as "an action plan for responding to misuse of information systems" (2014).

Incident response, then, is effectively applying those plans in response to an incident to reduce or remove incident effects on protected systems and organizations. Having prepared plans and using them well is vital to effective response. Not having or not using incident response plans is cited as the number one reason for ineffective response in SEC504 (SEC504, 2014). Another common failure is not taking good notes

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

in an incident. This not only hampers effective reporting and communication within the team and to stakeholders, but may also interfere with the reproducibility of key results.

Incident response decisions, and therefore the success of the entire process, rely on effective analysis and communication. Whether implied in the major phases of the standard Plan, Identify, Contain, Eradicate, Recover, Learn (PICERL) process or explicitly identified in a more complex process like that of DOD 6510.B, incident response pivots on analysis, its strength, and how well results are communicated to stakeholders (DOD, 2012).

1.3. DF in IR

Security professionals and educators who make use of digital forensics techniques in incident response have coined the acronym "DFIR" for their specific application of using digital forensic science with incident response and it is commonly defined as “digital forensics, incident response.” Another expansion of the acronym DFIR using Carrier's definition could be: "DFI in IR", using the tools and techniques developed for digital forensics science directly in incident response.

1.4. IPython science notebooks

The global scientific community uses a variety of software tools to aid research and analysis and many are available as free or open source software (FOSS) for free use, modification, and redistribution without license cost. FOSS tools are of particular interest in science because of the inherent transparency of source availability. FOSS tools are also often easier to customize and modify and the absence of software license fees removes many barriers to adoption that have slowed the spread of excellent commercial tools in the same fields. FOSS tools with a large community can develop very rapidly, as is the case with the GNU/Linux and FreeBSD operating systems.

Many popular scientific tools as well as an impressive selection of DFIR tools are written in the open source Python language including Volatility, Rekall, and plaso. The IPython project started as a set of tools to use Python and other languages for scientific computing, mathematics, and analysis (Roussant, 2014). IPython offers an enhanced Python interpreter terminal with built-in editing, syntax highlighting, and auto-

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

completion features and a sophisticated interactive notebook interface driven by the IPython core and using the same kernel.

The IPython system and its notebook format has become so widely used and successful with users of other science programming languages including R and Julia that a new project was launched in 2014 to better support the broader use and push the technology further. Announced at PyCon in July 2014 Project Jupyter will take over the infrastructure of IPython and brings native support for kernels in Python 2, Python 3, Julia, and R. (Perez, 2014) As March 2015, IPython 3.0 has been released and Jupyter development continues. Many of the new Jupyter capabilities are already available in some form including the multi-user notebook Collaboratory, Google Drive notebook support, and ephemeral notebook server tmpnb (Jupyter, 2015).

1.5. DFIR notebooks

The IPython notebook with its power, flexibility, and transparency is an excellent tool to perform DFIR analysis, collaborate with teams, and report results. The open source IPython application stack and notebook format allow leaders as well as investigators to understand and verify the machinery of the analysis workflow or any specific result. IPython notebooks smoothly interface not just with Python software but can also call out to other languages or use system commands. This allows easy integration of any tool with a command line interface, the ability to read and write files, or network access.

Standard notebooks and supporting workflows can improve consistency of analysis amongst teams and aid reproducibility of results. Templates can encourage analysts to follow processes completely and consistent formatting of case tracking information and meta-data aid record-keeping and can smooth handoffs between teams or shifts. Notebooks with good supporting workflow are self-documenting and the analysis and results in a notebook are inherently reproducible by loading the notebook onto another system with access to the same evidence and re-running cells or the entire notebook.

IPython notebooks offer many features to enhance sharing your progress and reporting results whether formally or informally with documents or even slide show

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

presentations. Notebook files can be shared directly or used for reporting and presentations using IPython features, optional libraries, and common office productivity software.

Notebook features also support on-the-job training, mentoring, and formal education allowing analysts and responders to share best practices as blank templates as well as fully or partially worked samples. Current notebook software supports simple "pair analysis" with dual monitor or online collaboration systems and future versions of the software already in development offer multi-user capabilities for live collaboration.

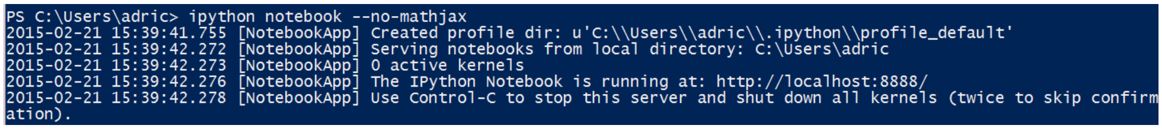
Analysts, responders, and their leaders can use these FOSS science tools for improved analysis and more effective response by using IPython notebook software customized to information security and DFIR as that provided by the DFIRnotes GitHub project (DFIRnotes, 2015).

2. Notebook Technology for DFIR

2.1. IPython Technology

IPython features a modular architecture that has made expansion and integration with other projects easier. With IPython 2.0 the kernel supports three user interfaces (UIs): the IPython enhanced text console, a graphical console powered by Trolltech's QT platform, and the web browser notebook interface. All computation takes place in the kernel and communication with the UIs uses a standard messaging framework. In fact, multiple UIs can be attached to one kernel and share code and data objects while multiple kernels can be run on one system and be isolated from one another.

IPython kernels, UIs, and utilities are configured primarily through the profile directory. A default profile is created on install and additional profiles can be created and invoked when running IPython applications to select a configuration as in this example first invocation of IPython notebook UI.



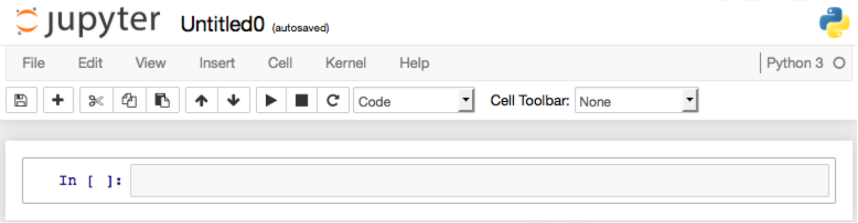
```
PS C:\Users\adric> ipython notebook --no-mathjax
2015-02-21 15:39:41.755 [NotebookApp] Created profile dir: u'C:\\Users\\adric\\.ipython\\profile_default'
2015-02-21 15:39:42.272 [NotebookApp] Serving notebooks from local directory: C:\Users\adric
2015-02-21 15:39:42.273 [NotebookApp] 0 active kernels
2015-02-21 15:39:42.276 [NotebookApp] The IPython Notebook is running at: http://localhost:8888/
2015-02-21 15:39:42.278 [NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirm
ation).
```

Fig 1: IPython first run creates profile

© 2015 The SANS Institute Author retains full rights.

© 2015 SANS Institute, Author retains full rights.

After installation, *ipython notebook* will start the notebook server and open the default web browser to the home page displaying notebooks. A new notebook can be created and opened with one click and will load in a new browser tab with a new code cell open. The built-in logo header menus and toolbar will display unless configured otherwise in the profile or a notebook. With IPython 3.0 (Project Jupyter) notebooks can use a selection of different language kernels and there is a kernel selector in the default toolbar top right.



Code

Cell Toolbar: None

In [ ]:

Fig 2: Jupyter standard notebook header with kernel selector

A user interface tour is offered on first invocation that briefly explains the UI, highlights core notebook functionality, and directs the user to online help for more information.

2.1.1. IPython notebooks

IPython's notebook interface has proven quite popular since its creation in 2011. As explained by Helen Shen in a November 2014 article in the journal *Nature*, although many other commercial and FOSS packages offer a notebook interface or some of the same features, IPython is far more widely adopted (Shen, 2014). Shen quotes Ana Nelson, creator of Dexy (another scientific computing package), "IPython notebook has become one of the most widely adopted programs of its kind". Shen credits this in her article to it being free and open source (FOSS) and to the Python language scientific community and its annual conference SciPy (Shen, 2014)

IPython notebooks have become increasingly popular with researchers and developers in just a few years. Many presentations and papers at conferences (not just at SciPy) use IPython notebooks as demonstrated by the IPython project's "gallery of interesting notebooks" (IPython, 2014a). Some science weblog sites are published using

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

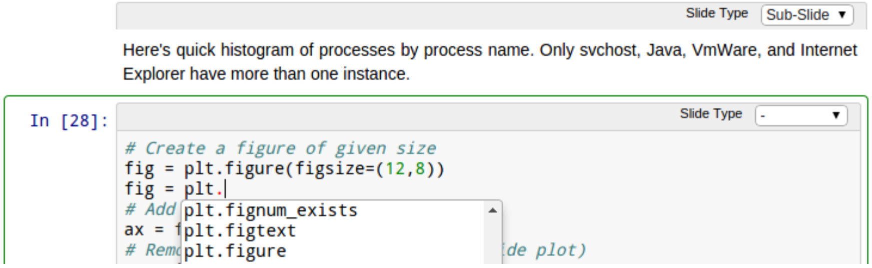
notebooks including Damian Avila's site http://www.damian.oquanta.info which has many useful tips for using and presenting with notebooks (Avila, 2014).

More ambitiously, some teams are using notebooks completely for their publications. For example, the newest IPython book by Cyrille Rossant, *IPython Interactive Computing and Visualization Cookbook*, was composed in notebooks and then processed for electronic and print publication as explained by the author in the Preface (Rossant, 2014).

2.1.2. Notebook Architecture

IPython is written in Python, but with versions 2 and 3 it has included more support for using other languages. Notebooks have always supported use of essentially any accessible tool though shebang, line, and cell magics, and IPython 3 can run kernels in Python (2 or 3) and in other languages including R and Julia. In fact, the technical and social support for using other scientific and statistical analysis languages is so strong within the IPython project and community that they have announced and begun an ambitious plan to spin off the notebook components into their own project, Jupyter (Perez, 2014).

IPython notebooks are comprised of meta-data and cells. A single notebook can include cells input as code, Markdown , or headers, and the output in text, graphics or other formats (IPython, 2014g). Code cells feature syntax highlighting and tab auto-completion to help with code input and readability.



Here's quick histogram of processes by process name. Only svchost, Java, VmWare, and Internet Explorer have more than one instance.

In [28]:

```
# Create a figure of given size
fig = plt.figure(figsize=(12,8))
fig = plt.
# Add plt.fignum_exists
ax = fplt.figtext
# Remcplt.figure                          ide plot)
```

Fig 3: python code cell from win5mem, tab completion

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute

Author retains full rights.

2.1.3. Notebooks Format

The IPython software as well as the notebooks are free and open source, cross-platform, and use an open architecture documented in IPython manuals. Notebook files are text encoded serialized objects in the popular JavaScript Object Notation (JSON) format. (IPython, 2014c) Although the internal format of the notebook is not guaranteed to be stable between releases, the JSON format allows for manual inspection and troubleshooting of notebook contents and various useful programmatic manipulations. For example, notebooks files check in easily to version control systems, such as Git and Fossil and can be processed into other formats directly if more customization or automation is desired than is provided by the notebook application (Rossant, 2014). Robust utilities for some of these transforms are included in IPython in the *nbconvert* package.

2.2. Notebook communication and security features

2.2.1. Notebook reporting & collaboration features

The structure of the IPython notebook lends itself not only to ease of use for programming and analysis but also includes robust support for documentation, reports, and even presentation. The cells of a notebook are configurable to different modes for code, rich text, or headers. Cell modes are switchable with the menus or keyboard shortcuts to code, define multiple levels of headings, or to input full Markdown structured text which will display as styled rich text when the cell is executed.

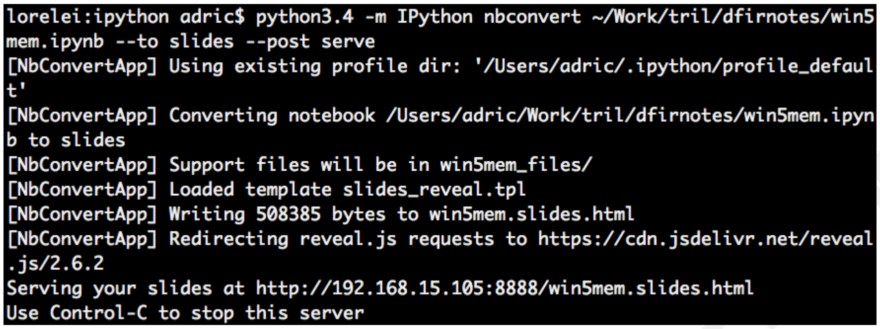
To share progress IPython notebooks can be sent to colleagues as a file attachment or checked into version control. Rather than sharing all work by sending the notebook an HTML rendering of the current notebook including input and output cells can be created and excerpted using the print preview feature (in the File menu). This conversion can also be done offline with *nbconvert*. Rendered notebook content can be shared using the system clipboard and generally pastes nicely into office productivity and messaging applications with formatting intact.

For presenting findings the notebook software natively supports slide shows. Cells in IPython 2 and later can be marked as different levels of slide content, or to be skipped over in a presentation. Once slides are selected *nbconvert* can write out a

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

notebook into a slides-only document in HTML or PDF or run the slideshow live in a browser using the Reveal.js framework (Hattab, 2014).



```
lorelei:ipython adric$ python3.4 -m IPython nbconvert ~/Work/tril/dfirnotes/win5
mem.ipynb --to slides --post serve
[NbConvertApp] Using existing profile dir: '/Users/adric/.ipython/profile_defaul
t'
[NbConvertApp] Converting notebook /Users/adric/Work/tril/dfirnotes/win5mem.ipyn
b to slides
[NbConvertApp] Support files will be in win5mem_files/
[NbConvertApp] Loaded template slides_reveal.tpl
[NbConvertApp] Writing 508385 bytes to win5mem.slides.html
[NbConvertApp] Redirecting reveal.js requests to https://cdn.jsdelivr.net/reveal
.js/2.6.2
Serving your slides at http://192.168.15.105:8888/win5mem.slides.html
Use Control-C to stop this server
```

Fig 4: start Reveal.js slideshow with nbconvert

Hannes Bretschneider, a PhD Student in Computer Science at the University of Toronto, explains how to hide the input cells for notebook slides shows with HTML and CSS in a blog post from November 2013 and provides a simple script demonstrating the technique (Bretschneider, 2013).

IPython notebooks also offer options for sharing completed results in notebooks. The *nbconvert* utility can use available system document processing tools to process notebooks in batch, for example as into portable document format (PDF) files with the *pdflatex* package. The notebook web UI features a print preview function using *pandoc* that renders the notebook to a static HTML document. This can be excerpted with copy and paste into productivity applications such as Microsoft Office or messaging clients with full rich text formatting and graphics or printed in the browser. The charting and visualization libraries available through notebooks can render out graphics files for reuse, such as this graphic from the DFIRnotes *win5mem* memory analysis sample notebook (DFIRnotes, 2014).

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

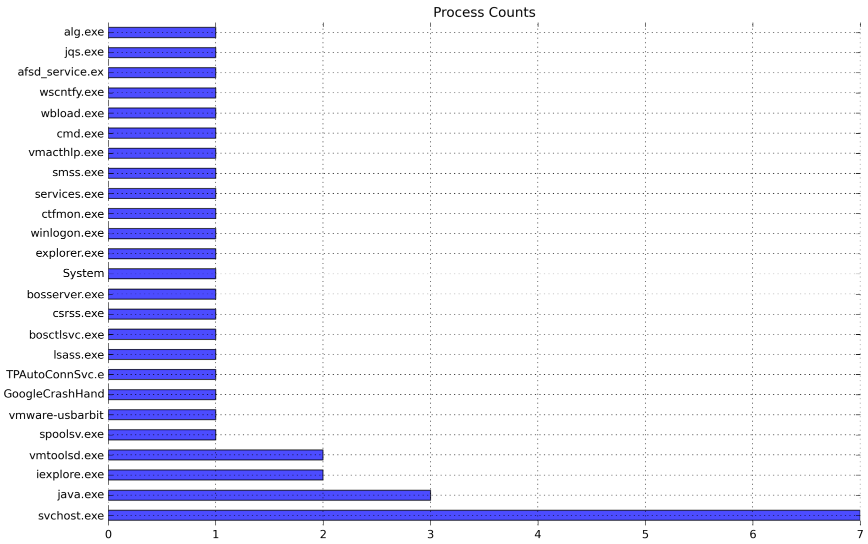


Fig 5: Process Counts plot from win5mem

2.2.2. Notebook security features

The notebook software system has several security features starting inside the notebook files themselves. Since IPython 2.0, the notebook file metadata contain a hash signed by a private key specific to the individual installation. The private key is kept in the user's IPython profile. As detailed by the IPython online documentation, this allows the notebook software to recognize notebooks that were created on that instance and these are regarded as trusted in the security model (IPython, 2014d). Untrusted notebooks have restricted execution of cells to prevent rogue scripts and malicious code from running without permission. For example, an untrusted (foreign) notebook's cells will not execute automatically on load. A notebook can be trusted manually to remove these protections in the Web UI or on the command line with the *trust* utility.

Although it is not recommended to expose the notebook server to hostile networks, the software can be configured for authentication and encrypted connections to reduce risks of unauthorized access and intercepted communications (IPython, 2014f). A shared login password's hash representation can be set in the profile via configuration file and the notebook server software can use transport layer security (TLS) for secure web connections if a certificate is provided. An IPython support document explains the process and configuration options (IPython, 2014f).

© 2015 The SANS Institute Author retains full rights.

Another security system of the notebook software is the separate notebook viewer. Rather than enabling interactivity and computation, *nbviewer* is specifically designed to allow for reading notebooks without executing their content. As explained in the *nbviewer* documentation, "the Notebook Viewer uses IPython's nbconvert to export .ipynb files to HTML" (IPython, 2014e). A public *nbviewer* instance hosts many of the example notebooks from IPython developers and security researchers alike. The development version of *nbviewer* is now part of Jupyter and its source is available through their GitHub project.

The Jupyter project has ambitious goals including multi-user notebooks running in public and private clouds, notebook software as web browser applications (eg in Google Chrome), and quick deploy temporary notebooks (tmpnb). There are details and active development on GitHub at https://github.com/jupyter.

2.3. Notebook Server Platform

Although the IPython notebook system is fully cross-platform and well supported on most Windows, Macintosh, and UNIX systems a standard platform is recommended for consistency of tools and to reduce support issues. The SANS Investigative Forensics Toolkit (SIFT) provides a wealth of analysis tools when installed atop a long term support release (LTS) of Ubuntu GNU/Linux with the *sift-bootstrap* package available from SANS on GitHub (SANS DFIR, 2014). The Ubuntu system package tools and python *pip* utility can be used to install all of the dependencies and optional tools to make good use of notebook software. SIFT version 3 includes IPython but additional utilities and libraries are needed for full notebook server use and some *nbconvert* functions. A sample install script for the dependencies is provided in the DFIRnotes GitHub project.

2.3.1. Offline

One peculiarity of designing systems for incident response or digital forensics is they must usually operate in restricted networks without Internet access or sometimes entirely offline. Although the Linux, IPython, SIFT software are entirely available online it is not recommended for an analysis system in use to (still) have Internet access. This is to reduce risk of contaminated analysis and lower the likelihood of malware propagation or attacker movement through analysis systems. Some simple preparation and additional

© 2015 The SANS Institute Author retains full rights.

downloads can smooth this experience for the analyst and should be done in combination with system hardening to prepare a notebook powered analysis server for response activities.

The IPython notebook software uses the MathJAX JavaScript library to render equations in notebooks and by default expects to find the latest version of the library on a public Internet site. If this feature is not in use it can be disabled in the configuration or execution of the notebook software or if desirable the library can be downloaded for offline use easily.

Some analysis packages use profiles, such as the memory image profiles used by Volatility and Rekall. It is recommended to download the profiles needed for an investigation ahead of time and like all software these may need to be updated or replaced as upstream changes.

The live slideshow feature of IPython notebooks uses the JavaScript library Reveal.js and looks for it online by default when running slideshows out of *nbconvert*. The Reveal.js project provides instructions online for downloading the library and configuring for offline use (Hattab, 2014)

All of the tools tested for this project were easily configured for offline use and a sample script in the DFIRnotes GitHub project contains the detailed steps.

2.3.2. Hardening

Response and analysis systems should be hardened against the potential threats modeled in incident response planning and in compliance with any organizational policy. Many of the analysis packages have network services (beyond the notebook itself) and all should be reviewed. Follow best practices, review all services for need, and protect them with firewalls, authentication, and strong cryptography. Are you printing from your analysis server, sharing files, streaming media? Consider uninstalling any packages not used in your environment or standard work process to reduce the attack surface of your analysis servers. Finally, strike a balance between the offline or restricted networking requirements of these systems and keeping them patched for any security problems. Network services, file and protocol parsers, and systems dealing with malware or attacker

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

artifacts are all at heightened risk for exploitation as detailed in SEC504 Day 4 (SEC504, 2014) and these analysis systems are all of those things at once.

2.3.3. IPython profile

The IPython notebook software is user configurable through the invocation arguments to the tools and through the IPython profile and configuration files. Defaults for settings like the IP address and port to bind the notebook server to, various security and privacy settings, and feature selection (eg disable MathJAX) can be set in the notebook configuration files within a profile. Scripts can be configured to provide additional features in this way. A file included in the default profile at *.ipython/profile_default/startup/README* explains how to add scripts that run with the application server startup, such as an example script from the IPython project cookbook which logs all IPython commands (not just notebook cell entry) to a separate log file when copied or linked into your profile as *.ipython/profile_default/startup/05_log.py* (IPython, 2014b).

Startup scripts, and all profile modifications, should be used carefully and compatibility with all software checked. The example logging script *05_log.py* uses Python 2.7 syntax that does not work correctly under Python 3.

Also available within the profile are file locations for custom CSS and JavaScript. The CSS file is used not only for styling of the notebook web UI, but also notebooks browsed through *nbviewer* or processed by *nbconvert* (Roussant, 2014). Using custom JavaScript in the profile enables injection and manipulation of cell content, notebook metadata, and reconfiguring of the notebook user interface (Roussant, 2013). These JavaScript APIs are available live in a notebook cells, but are limited in scope. Scripts configured in the profile will execute for each notebook opened or created and can be quite powerful.

The DFIRnotes sample install script also creates an IPython profile for DFIR notebooks and configures some useful settings as well as a custom theme. It is available in the DFIRnotes GitHub project.

© 2015 The SANS Institute Author retains full rights.

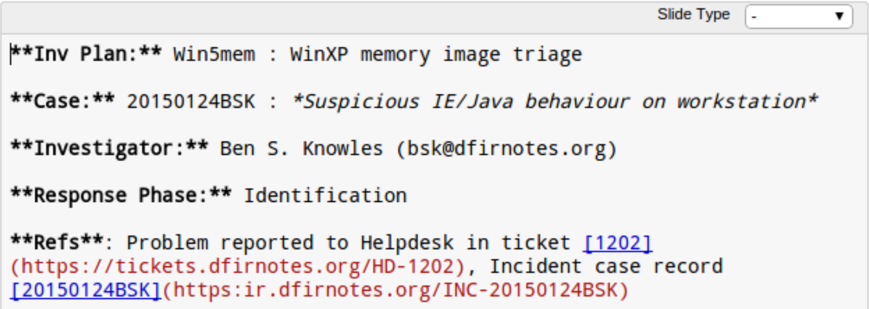
2.4. Notebook powered DFIR Workflow

An example workflow for memory evidence triage will illustrate the advantages of notebook software for DFIR. In this scenario an incident handler who is presented with raw evidence and a report of suspicious behaviour must quickly determine if an incident has occurred and then report back to their team with recommended action.

Pre-configured templates support analysts in following process and provide best practices in executable form. For this example the template *win5mem* used is for Volatility powered triage analysis of a Windows XP memory image. The template provides a case data header, the basic investigation plan, and some scripting and analysis code. The *win5mem* template and a completed sample are provided in the DFIRnotes GitHub project (DFIRnotes, 2014).

2.4.1. New investigation

Upon receiving the case the handler opens a new notebook from the template matching the investigation, in this case Windows XP memory triage (*win5mem*), fills in the case details, and reviews the investigation plan.



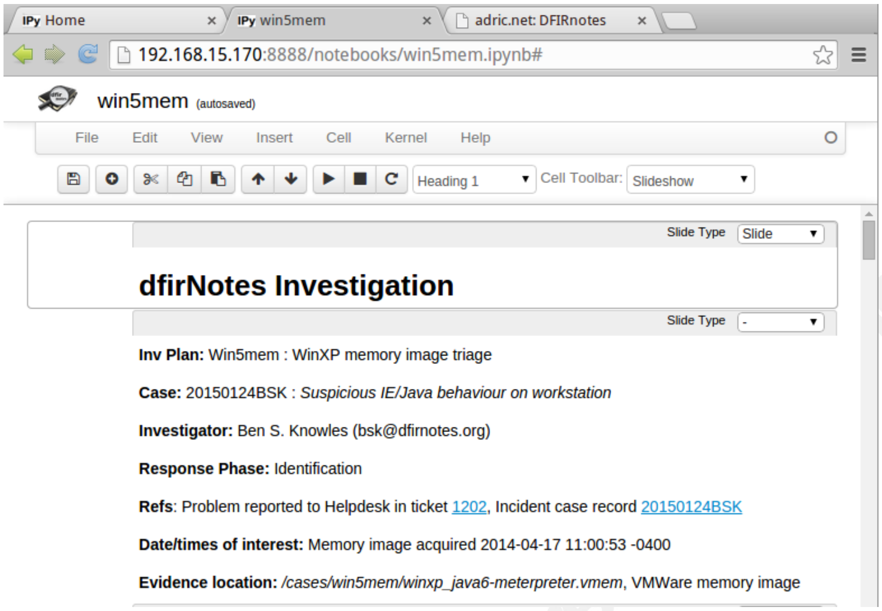
```
**Inv Plan:** Win5mem : WinXP memory image triage

**Case:** 20150124BSK : *Suspicious IE/Java behaviour on workstation*

**Investigator:** Ben S. Knowles (bsk@dfirnotes.org)

**Response Phase:** Identification

**Refs**: Problem reported to Helpdesk in ticket [1202]
(https://tickets.dfirnotes.org/HD-1202), Incident case record
[20150124BSK](https:ir.dfirnotes.org/INC-20150124BSK)
```

Fig 6: Case meta from win5mem, Markdown

Once the header cell meta-data form is completed in Markdown executing the cell with IPython notebook toolbar buttons or keyboard shortcuts renders it as rich text.

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute
Author retains full rights.



dfirNotes Investigation

Inv Plan: Win5mem : WinXP memory image triage

Case: 20150124BSK : *Suspicious IE/Java behaviour on workstation*

Investigator: Ben S. Knowles (bsk@dfirnotes.org)

Response Phase: Identification

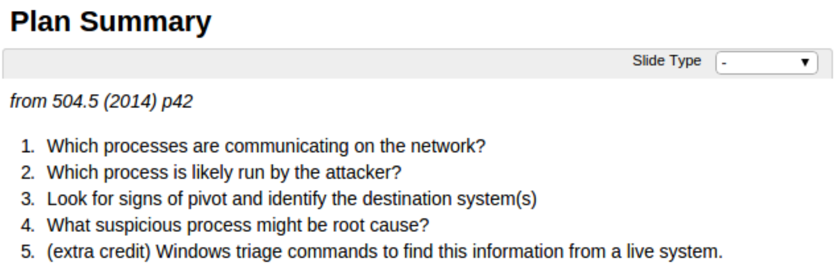
Refs: Problem reported to Helpdesk in ticket 1202, Incident case record 20150124BSK

Date/times of interest: Memory image acquired 2014-04-17 11:00:53 -0400

Evidence location: */cases/win5mem/winxp_java6-meterpreter.vmem*, VMWare memory image

Fig 7: Case meta-data from win5mem

The completed investigation plan should detail the questions to answer and contain any hypotheses to disprove for the case. For memory analysis in the triage scenario we use the methodology from SEC504 Day 5 to look first for processes that were communicating on the network (SEC504, 2014).



Plan Summary

from 504.5 (2014) p42

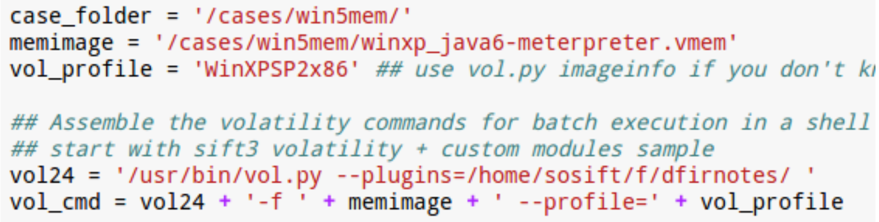
1. Which processes are communicating on the network?
2. Which process is likely run by the attacker?
3. Look for signs of pivot and identify the destination system(s)
4. What suspicious process might be root cause?
5. (extra credit) Windows triage commands to find this information from a live system.

Fig 8: investigation plan from 504.5, in win5mem

After completing the investigation information block and saving the new notebook under its correct name, the handler updates the variable names specific to the evidence in this case specifying the path to the memory image and selecting (and verifying) a Volatility profile for that image. Executing this code cell sets the variables we will use for the analysis.

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

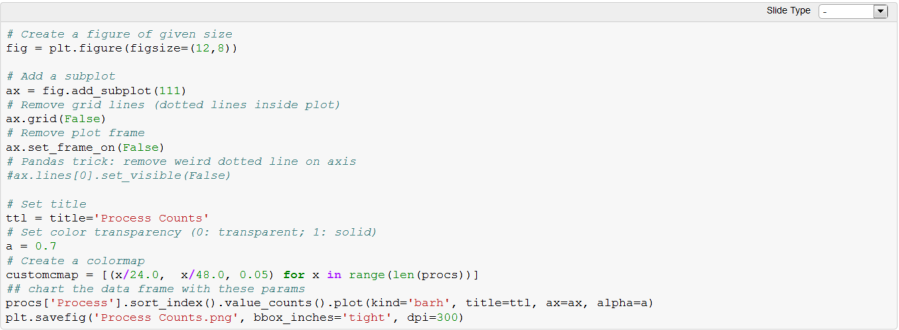


```
case_folder = '/cases/win5mem/'
memimage = '/cases/win5mem/winxp_java6-meterpreter.vmem'
vol_profile = 'WinXPSP2x86' ## use vol.py imageinfo if you don't k

## Assemble the volatility commands for batch execution in a shell
## start with sift3 volatility + custom modules sample
vol24 = '/usr/bin/vol.py --plugins=/home/sosift/f/dfirnotes/ '
vol_cmd = vol24 + '-f ' + memimage + ' --profile=' + vol_profile
```

Fig 9: code cell variables from win5mem

2.4.2. Analysis

Using the defined variables the handler executes the code cells provided from the template to quickly complete the first stage of the investigation plan by enumerating the process and connections with a batch of Volatility queries. The output is then read into Pandas dataframes to plot graphs using some tips from the Data Science Lab's Wordpress site (Data Science Lab, 2013).



```
# Create a figure of given size
fig = plt.figure(figsize=(12,8))

# Add a subplot
ax = fig.add_subplot(111)
# Remove grid lines (dotted lines inside plot)
ax.grid(False)
# Remove plot frame
ax.set_frame_on(False)
# Pandas trick: remove weird dotted line on axis
#ax.lines[0].set_visible(False)

# Set title
ttl = title='Process Counts'
# Set color transparency (0: transparent; 1: solid)
a = 0.7
# Create a colormap
customcmap = [(x/24.0,  x/48.0, 0.05) for x in range(len(procs))]
## chart the data frame with these params
procs['Process'].sort_index().value_counts().plot(kind='barh', title=ttl, ax=ax, alpha=a)
plt.savefig('Process Counts.png', bbox_inches='tight', dpi=300)
```

Fig 10 pandas plot code from win5mem

Based on the occurrence counts and details from the process and connection data the handler further examines some processes and connections of interest and charts that data as well.

© 2015 SANS Institute Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

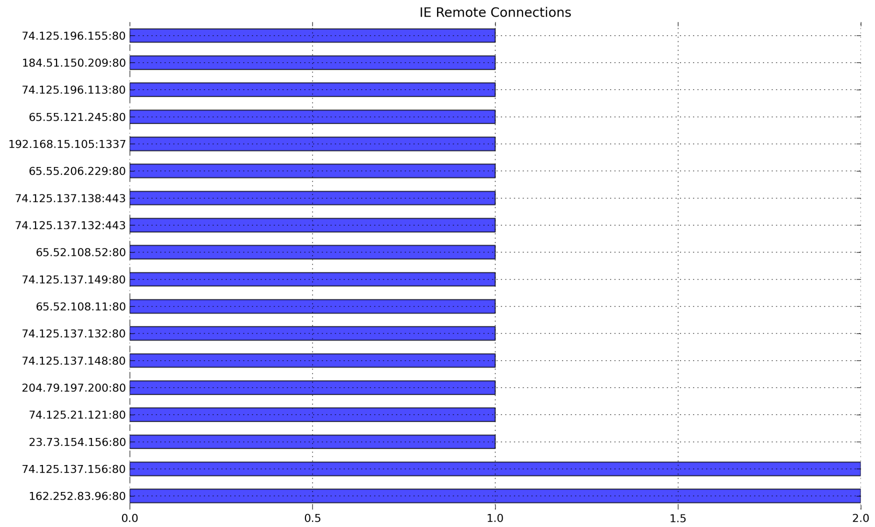
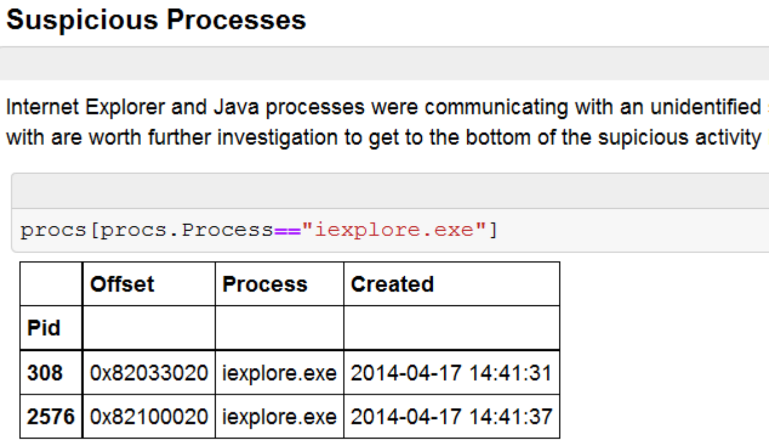


Fig 11: IE process connections graph from win5mem

2.4.3. Recommend action

From the process and connection data and an understanding of the Windows platform, the handler identifies suspicious processes and another host implicated in the activity. The handler summarizes the results with charts sliced from the dataframes and writes brief statements to recommends action, in this case containment and further investigation.



Suspicious Processes

Internet Explorer and Java processes were communicating with an unidentified
with are worth further investigation to get to the bottom of the supicious activity

```
procs[procs.Process=="iexplore.exe"]
```

	Offset	Process	Created
Pid			
308	0x82033020	iexplore.exe	2014-04-17 14:41:31
2576	0x82100020	iexplore.exe	2014-04-17 14:41:37

Fig 12: Results process chart from win5mem

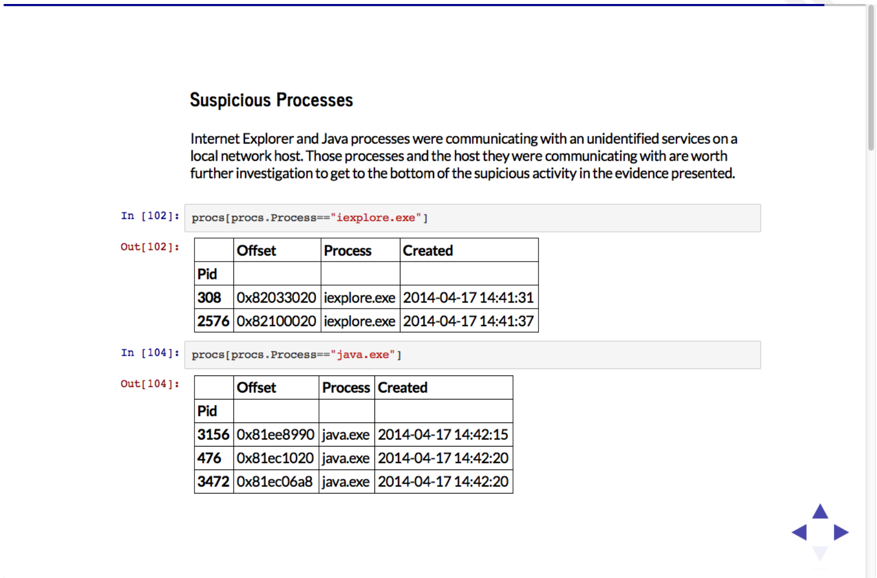
© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

2.4.4. Collaborate

With the investigation plan completed and results in hand the handler can consult with peers by sharing the notebook application session live with conferencing software or present to a group by using *nbconvert* to run a Reveal.js slideshow from the notebook.

Reveal.js loads the slides in a web browser application with navigational aids including a subtle progress bar at the top and optional separate display of speaker notes.



Suspicious Processes

Internet Explorer and Java processes were communicating with an unidentified services on a local network host. Those processes and the host they were communicating with are worth further investigation to get to the bottom of the supicious activity in the evidence presented.

```
In [102]: procs[procs.Process=="iexplore.exe"]
```

Out[102]:

	Offset	Process	Created
Pid			
308	0x82033020	iexplore.exe	2014-04-17 14:41:31
2576	0x82100020	iexplore.exe	2014-04-17 14:41:37

```
In [104]: procs[procs.Process=="java.exe"]
```

Out[104]:

	Offset	Process	Created
Pid			
3156	0x81ee8990	java.exe	2014-04-17 14:42:15
476	0x81ec1020	java.exe	2014-04-17 14:42:20
3472	0x81ec06a8	java.exe	2014-04-17 14:42:20

Fig 13: Suspicious Process slide in Reveal.js

2.4.5. Reporting

Once any team contributions are incorporated the completed notebook file can be saved (ipynb extension). The notebook can be checked into revision control, attached to an incident case record for tracking, or just emailed as a file attachment. Plots and other generated artifacts can be created as independent image files in standard formats displayed in a notebook. The images are then portable on their own. This technique was used in the sample notebook for the graphs shown in this paper.

Once analysis is completed the handler can pull the results as rich text into an office document via copy and paste and include the generated graphic files in formal reports.

© 2015 The SANS Institute Author retains full rights.

2.4.6. Workflow possibilities

The workflow described is flexible enough to support teams of different sizes and capabilities and offers many points for possible automation through scripting and programming. A sample new case script is available from the DFIRnotes GitHub project that includes some simple workflow automation hooks.

More sophisticated scripts could incorporate user input to select templates, set up case meta-data, or even start background processing of evidence. Interactive forms are possible with use of the IPython notebook widgets and JavaScript. With even more automation, notebook applications could be created and started on demand by remote invocation from an investigation management system like Request Tracker Incident Response (RTIR) from Best Practical.

2.5. Conclusion

The notebook technology from IPython is a powerful toolset in its own right and can be easily and effectively customized to enhance DFIR activities by supporting stronger analysis, enabling better reporting, and by powering analyst and responder mentoring and education. Notebooks are a flexible interface to DFIR software and are already used by DFIR tool projects like Workbench and Rekall (Workbench, 2014) and many sample analyses are posted by projects and individuals alike. A selection of educational public DFIR notebooks from various authors are linked in the DFIRnotes GitHub project.

The IPython notebook and included data analysis packages bring DFIR data into the standard formats including Pandas dataframe and Hierarchal Data Format (HDF) used by researchers worldwide with science software packages of all kinds. This make DFIR notebooks a bridge that can link process driven DFIR practices with the impressive techniques of the emerging field of data science. Responders can then take advantage not only of descriptive and inferential statistical analysis but also newer analysis technologies like map-reduce and machine learning in order to solve the challenges faced from processing ever larger, richer data sets to defend today's complex systems.

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

3. References

Avila, Damian (2014). Using a local Reveal.js library with your IPython slides [Web log post]. Retrieved from http://www.damian.oquanta.info/posts/using-a-local-revealjs-library-with-your-ipython-slides.html

Bretschneider, Hannes (2013). IPython Slideshows will change the way you work [Web log post]. Retrieved from http://hannes-brt.github.io/blog/2013/08/11/ipython-slideshows-will-change-the-way-you-work/

Carrier, Brian (2005). *File System Forensic Analysis*. Addison-Wesley: Upper Saddle River, NJ.

Data Science Lab (2013). Beautiful plots with Pandas and Matplotlib [Web log post]. Retrieved from https://datasciencelab.wordpress.com/2013/12/21/beautiful-plots-with-pandas-and-matplotlib/

Department of Defense (2012). *Cyber Incident Handling Program*. Retrieved from http://dtic.mil/cjcs_directives/cdata/unlimit/m651001.pdf

DFIRNotes Project (2015). https://github.com/adricnet/dfirnotes/

Hattab, Hakim El (2014). Reveal.js Full Install [Web log post]. Retrieved from https://github.com/hakimel/reveal.js/#full-setup

IPython Project (2014a). A gallery of interesting IPython notebooks [Wiki entry]. Retrieved from https://github.com/ipython/ipython/wiki/A-gallery-of-interesting-IPython-Notebooks

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

IPython Project (2014b). Cookbook: Dated logging [Wiki entry]. Retrieved from https://github.com/ipython/ipython/wiki/Cookbook:-Dated-logging

IPython Project (2014c). Notebook JSON file format [Article]. Retrieved from http://ipython.org/ipython-doc/stable/notebook/nbconvert.html#notebook-format

IPython Project (2014d). Notebook security features [Article]. Retrieved from http://ipython.org/ipython-doc/dev/notebook/security.html

IPython Project (2014e). Notebook viewer FAQ [Article]. Retrieved from http://nbviewer.ipython.org/faq

IPython Project (2014f). Securing a notebook server [Article]. Retrieved from http://ipython.org/ipython-doc/dev/notebook/public_server.html

IPython Project (2014g). Structure of a notebook document [Article]. Retrieved from http://ipython.org/ipython-doc/stable/notebook/notebook.html#structure-of-a-notebook-document

Jupyter Project (2014). Project Jupyter [Source code]. Retrieved from https://github.com/jupyter

National Institute for Standards and Technology (NIST) (2006). *Guide to Integrating Forensic Techniques into Incident Response* (SP 800-86). Retrieved from http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf

Perez, Fernando (2014). Project Jupyter [Presentation]. Retrieved from https://speakerdeck.com/fperez/project-jupyter

Rossant, Cyrille (2013). *Learning IPython for Interactive Computing and Data Visualization*. Birmingham, UK: Packt Press.

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

Rossant, Cyrille (2014). *IPython Interactive Computing and Visualization Cookbook*. Birmingham, UK: Packt Press.

SANS DFIR (2014). Sift 3 Bootstrap instructions [Article]. Retrieved from https://github.com/sans-dfir/sift-bootstrap

Skoudis, Ed, Strand, John, SAN Institute (2014). *Hacker Techniques, Exploits, and Incident Handling* (SEC504). Bethesda, MD: SANS Institute.

Shen, Helen (2014). Interactive notebooks: Sharing the code. *Nature.* Retrieved from http://www.nature.com/news/interactive-notebooks-sharing-the-code-1.16261

Workbench project (2014). Workbench: A scalable python framework for security research and development teams [Article]. Retrieved from http://workbench.readthedocs.org/en/latest/

© 2015 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.