Global Information Assurance Certification Paper

Forensicator FATE - From Artisan To Engineer

GIAC (GCFA) Gold Certification

Author: Barry Anderson, shori@bigpond.net.au
Advisor: Chris Walker

Accepted: October 5, 2014

Abstract

The SIFT workstation is an incredibly useful collection of artisan's tools and the processes, like the creation of an artwork, are often laborious. This paper presents an approach to automating the mundane and repetitive tasks, freeing the reader up to spend more time analysing the generated data, and enabling the handling of more cases in parallel. The approach also allows multiple analysts to work collaboratively across one or many cases. The software described herein is also provided.

Acknowledgement

Bianca Munoz-Greco (PWC) spent many hours beta-testing the install scripts and environment described in this paper.

1. Introduction

The SANS Investigative Forensic Toolkit (SIFT) is an awesome set of (free!) tools for the forensics professional. Using these tools effectively however can be overwhelming, especially in the case of a large complex case such as an APT intrusion. The processes are laborious, and it's not easy to share results between members of a geographically dispersed team – especially across time zones!

Forensicator FATE aims to change all that, bringing the learnings of DevOps to Digital Forensics and Incident Response (DFIR), enabling one responder to easily switch back and forth between multiple cases, and multiple responders to collaborate on the same case, which empowers specialization, and allows for transforming DFIR from an artisanal craft to an engineering process.

2. Building a distributed DFIR lab

Our intention in extending the SIFT workstation with this toolchain is that, instead of obtaining actionable evidence requiring a DFIR artisan needing to take tens or even hundreds of small interrelated actions to produce actionable data, for the classic APT/Windows investigation, even a neophyte DFIR engineer can follow a repeatable six step process (of which Step 1 is transferring the data onto the SIFT workstation, and Step 6 is viewing the timeline produced). See Figure 1.

So, let's first describe what exactly we're out to build, and then build it!

There are five pieces to what we're out to build. They are:

1) The SIFT workstation – our toolkit and the truly essential part of the solution;
2) An Elasticsearch server (actually Elasticsearch, Logstash, Kibana) – this is how we visualize the data we generate;
3) Jenkins – a Continuous Integration server that allows us to automate our processes and monitor them ongoingly (when extended with the appropriate plugins);
4) A PostgreSQL database – this is where we store the case metadata that doesn't end up in Elasticsearch;
5) A lightweight DFIR Case Manager that frees us from having to deal with even the complexities of Jenkins for a standard Windows investigation.

   Importantly, whilst obviously more resources (CPU, RAM, Fiber-attached SAN) are better, we can stand up an environment that not only demonstrates the approach but also allows real work on two VM's which can even run on the same workstation. (Two cores at least are recommended for each of the

SIFT and ELK VM's, and the more memory for the SIFT workstation the better, though the ELK VM will run quite comfortably with 1G of RAM.)

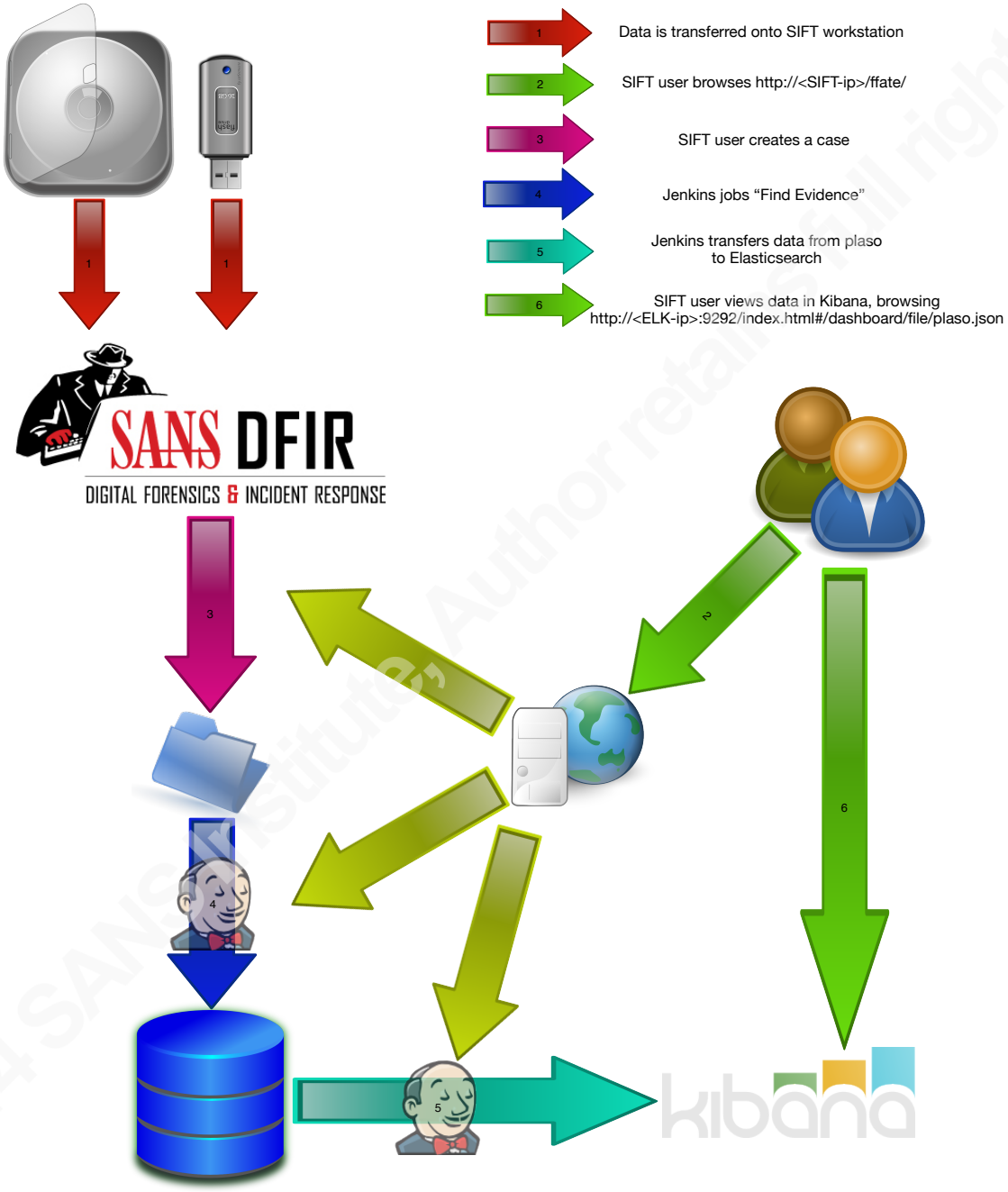

Figure 1, the six step process to actionable intelligence

2.1. SIFT Workstation(s)

First we start with the SIFT workstation(s), created by SANS Faculty Fellow Rob Lee and a team of forensics experts and made freely available to the community.

Barry Anderson, shori@bigpond.net.au

Version 3 is available to download as a zipped VM of the workstation at: https://digital-forensics31.sans.org/community/download-sift-kit/3.0. This is probably the best approach if you wish to have your SIFT workstation be a virtual machine and are using any of the VMWare products (Player, Workstation, or Fusion).

Alternatively, if you already have an Ubuntu 12.04 workstation (either physical or virtual), you can use the script at https://github.com/sans-dfir/sift-bootstrap to bootstrap a SIFT workstation.
(As at September 2014, work is ongoing to port this to Ubuntu 14.04, but this is not yet complete. Check GitHub for the status of this effort before proceeding to attempt to deploy a SIFT workstation on Ubuntu 14.04.)

Points to note about the SIFT workstation when setting up a distributed lab:

- it shares information over SMB so not only does traffic need to be routable between endpoints, but the appropriate network ports need to be open through any firewalls between endpoints (137/udp, 138/udp, 139/tcp, and 445/tcp);
- SMB involves broadcasts of the name of the node, so to avoid confusion we need to ensure that SIFT hosts are renamed e.g. sift1, sift2 (in /etc/hosts, /etc/hostname, and /etc/samba/smb.conf). Note: do not name the SIFT workstations siftworkstation1, siftworkstation2 etc; Microsoft Windows limits computer names to 15 characters so these names are indistinguishable from one another.

Now we add the software, scripts and tasks we need to the SIFT workstation. If your SIFT workstation is Internet connected, you can simply open a terminal and run the following commands, which will:

- place the contents of all the scripts required for the rest of this work in a directory forensicator-fate; and
- run the bundle installer.

All you need to do is wait approximately 15 minutes, occasionally hitting the Enter key when prompted with:

```
Do you want to continue [Y/n]?
```

```
sansforensics@sift1:~$ git clone https://github.com/z3ndrag0n/forensicator-
fate.git
sansforensics@sift1:~$ forensicator-fate/scripts/ffate-bundle-installer.sh
```

The bundle installer will:

- install pyelasticsearch to allow us to transfer data into ElasticSearch for visualization (see Section 2.2);
- create additional network shares on the SIFT workstation to allow us to collaborate more effectively with other teams (see Section 2.3);

Barry Anderson, shori@bigpond.net.au

- install Jenkins to allow us to automate our evidence processing and visually check the status of those automated tasks (see Section 3.2);
- install the PostgreSQL database (and the Python language bindings to it) that allow us to store case metadata safely (see Section 3.3);
- install the Forensicator FATE DFIR Case Manager (see Section 3.4).

If your workstations are not Internet connected, but you can transfer files on to them via the network or removable media, perform the above command on an Internet connected workstation, then copy the resulting directory as appropriate. At worst, the appendices of this paper contain the versions of these scripts as at the publication of this paper (of course if you are in this situation, you will need to work out another way of getting the software distribution archives on to your SIFT workstation and tailor the scripts appropriately). This approach is **not** for the faint of heart. On a map, it would be labeled "Here be Dragons."

2.2. Visualising our evidence

Now we setup a logging and data visualization workstation. We use the ElasticSearch, Logstash and Kibana (ELK) combination, to allow for rapid searching of essentially unstructured and semi-structured data. All results go into ElasticSearch, so as new Indicators of Compromise (IOCs) are discovered, any member of the team can search for them.

For consistency with FOR572 (Advanced Network Forensics and Analysis) and to enable processing and visualizing network data, we use the ELK distribution that contains Xplico produced by Phil Hagen for FOR572 and available at http://bit.ly/for572-logstash-readme.

Plaso (the successor to log2timeline), allows for direct export into ElasticSearch. To test that Plaso is ElasticSearch enabled, simply run the following command:

```
sansforensics@sift1:~$ psort.py -o list
****************************** Output Modules *******************************
    L2tcsv : CSV format used by log2timeline, with 17 fixed fields.
    L2ttln : Extended seven field pipe deliminated TLN; L2T 0.65 style.
   Elastic : Saves the events into an ElasticSearch database.
   Dynamic : Dynamic selection of fields for a separated value output format.
     Rawpy : Prints out a "raw" interpretation of the EventObject.
    Sql4n6 : Saves the data in a SQLite database, used by the tool 4n6Time.
  Pstorage : Dumps event objects to a plaso storage file.
       Tln : Five field TLN pipe deliminated outputter.
-------------------------------------------------------------------------------
```

If the output contains "Elastic" as highlighted above, you are ready to proceed, and can send Plaso data to an ElasticSearch repository with a command such as:

```
sansforensics@sift1:~$ psort.py -z EST5EDT -o Elastic \
--elastic_server_ip=172.16.223.144 plaso.dump
```

In the above command:

EST5EDT is the timezone of the target system;
172.16.223.144 is the IP address assigned to your ELK VM (the console of the VM shows you the correct address on startup);
plaso.dump is the plaso database used to store the results of your evidence processing.

Now you need to download a dashboard. This is not simply a script, but a description for Kibana (the 'K' in ELK) of how we want our visualization interface to look and behave. The sample originally produced by the plaso team (which provides queries, filters, a histogram and a source distribution breakdown by plaso parser) can be found at:
https://plaso.googlecode.com/git/extra/plaso_kibana_example.json. Once downloaded it must be copied on to the ELK VM with scp and then moved to where Kibana can find it with the following command:

```
[ls_user@for572-logstash ~] sudo cp plaso_kibana_example.json \
/opt/logstash/vendor/kibana/app/dashboards/plaso.json
```

With the plaso Kibana dashboard installed, viewing the data stored in ElasticSearch is as simple as pointing a browser at
http://172.16.223.144:9292/index.html#/dashboard/file/plaso.json.

Barry Anderson, shori@bigpond.net.au

You will see something like Figure 2, below.

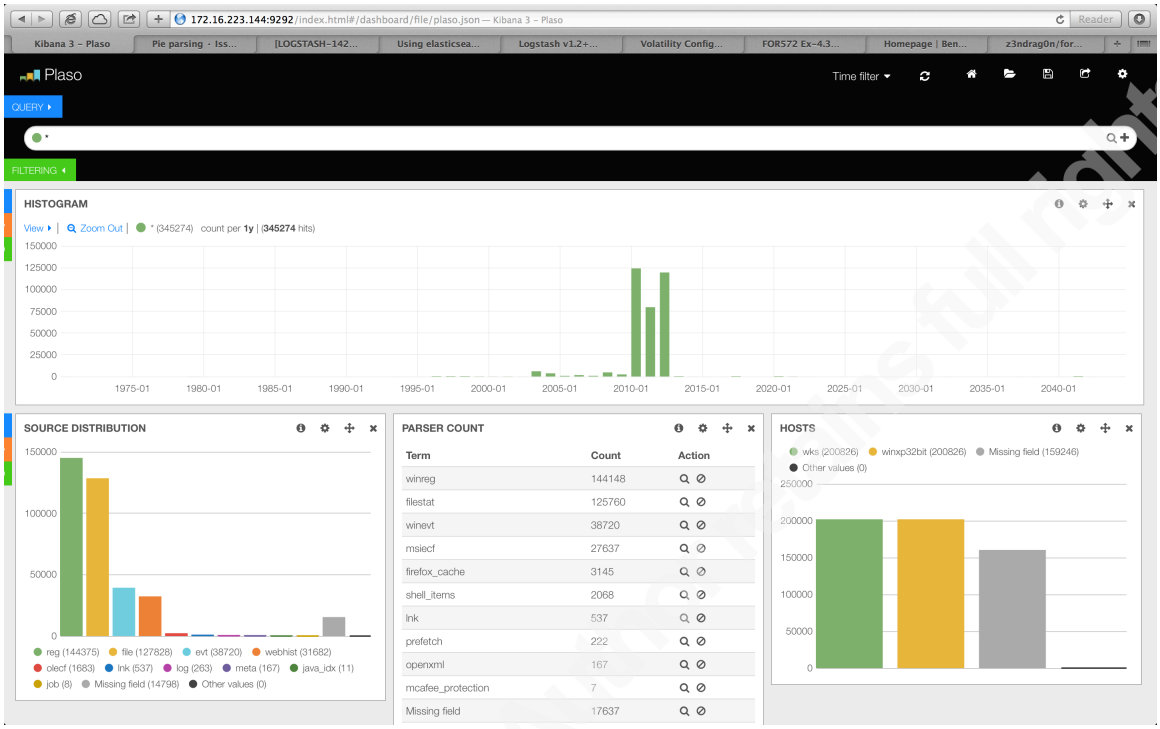

Figure 2 – the plaso Kibana dashboard

If you scroll down within this window, you come to the documents pane, which contains our ***supertimeline*** records, as per Figure 3.

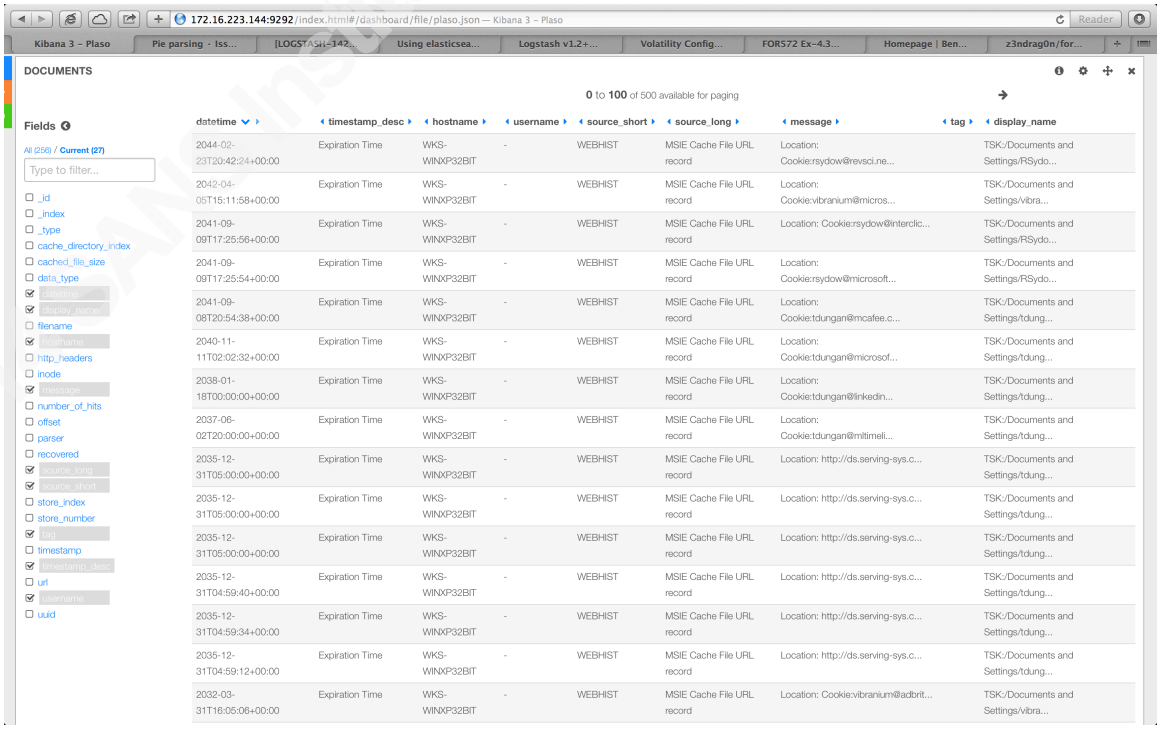

Figure 3 - the documents pane containing supertimeline records

Barry Anderson, shori@bigpond.net.au

Click on the magnifying glass to the right of "prefetch" in the Parser Count table, and the ***supertimeline*** records are filtered to only pre-fetch records, as per Figure 4.

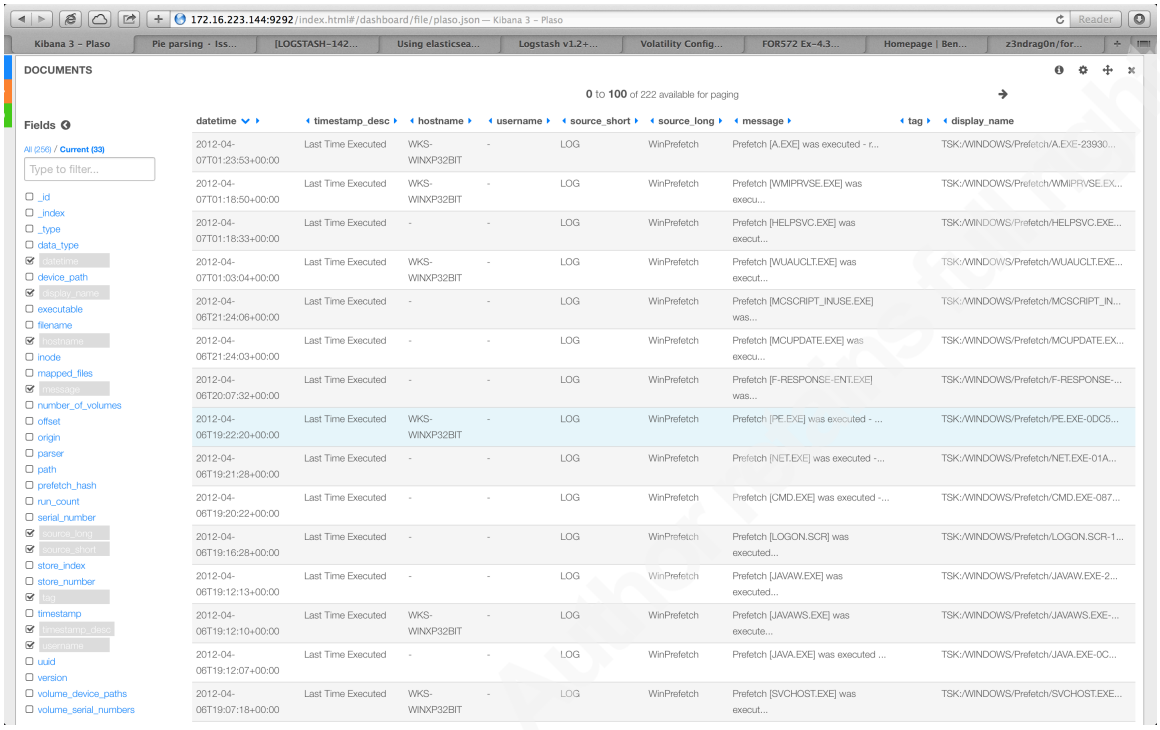

Figure 4 - the *supertimeline*, filtered for prefetch records

To expand a record, click its summary line, and the display changes as per Figure 5.

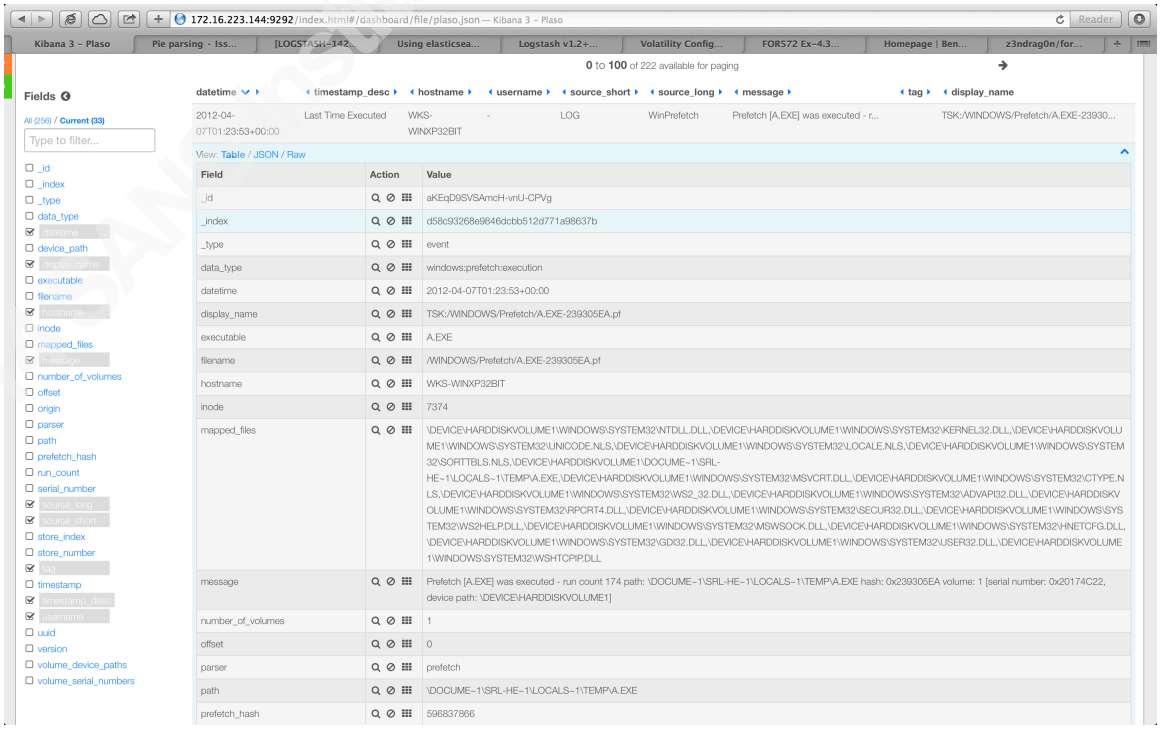

Figure 5 - details of a prefetch event expanded

Barry Anderson, shori@bigpond.net.au

You can have multiple records expanded at the same time; to collapse a record, simply click on the summary line again.

2.3. Collaboration empowers specialization

Allowing multiple responders to collaborate on the same case empowers specialization, which allows for transforming DFIR from an artisanal craft to an engineering process. Doing this effectively requires sharing information quickly and at low cost between specialists, so we added to the standard SIFT shares of /cases and /mnt, in order to share the information required for other tools and processes:

- /reverse - as we discover new targets for malware analysis, we want our reverse engineers to be able to access them;
- /ioc - as we uncover new evidence, we may wish to generate Indicators of Compromise (IOCs), which we would like to be immediately available to colleagues using Mandiant Redline to perform memory analysis, in addition to indicators from http://www.openioc.org and http://www.iocbucket.com;
- /whitelist - where we place any "known good" hashes - a copy of the National Software Reference Library (NSRL), and any hashes produced in house, e.g. as per: http://sniperforensicstoolkit.squarespace.com/malwaremanagementframework;
- /blacklist - where to store the HashKeeper database if your team has access to it, as well as any hashes from previous cases, or any Threat Intelligence sharing groups your organization belongs to;
- /artifacts – where we store those artifacts created as we process the evidence;

Browsing the SIFT workstation across the network now looks like the following:

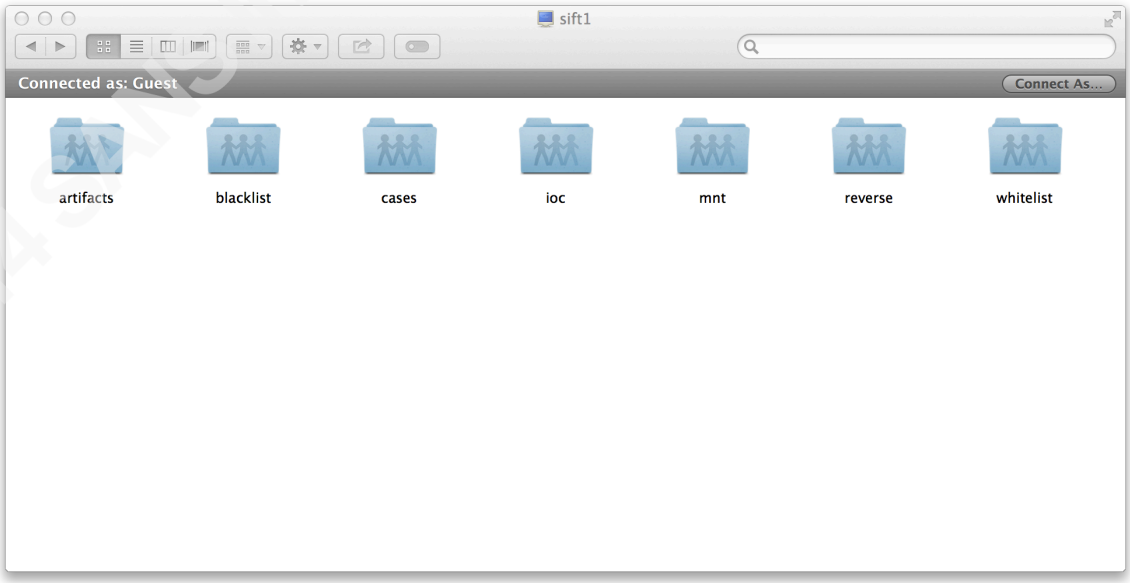

Figure 6, our extended set of SIFT shares

3. Bringing DevOps to DFIR

3.1. What is DevOps?

DevOps can be thought of as the fusion of Development and Operations, where Developers develop with Operations in mind and Operations avail themselves of the same tools and techniques as the Developers. (Security Professionals, in case you're not clear, fall under Operations in this view of the world.)

There is a **lot** of information on the web, twitter and IRC about DevOps.
Two good examples of places to start if you are unfamiliar with the concept and interested in learning more are:
http://theagileadmin.com/what-is-devops/ and
http://www.jedi.be/blog/2010/02/12/what-is-this-devops-thing-anyway/.
The later contains a wide-ranging list of resources, if you want to learn more about the topic.

The aspects of the tools and techniques we concentrate on bringing to our world of DFIR Operations are automation and Continuous Integration.

3.2. Jenkins

When Forensicators are (geographically or temporally) dispersed, what works is for everyone on the team to be able to see the status of any tasks at a glance.

This is where we bring automation and Continuous Integration to DFIR. We use Jenkins (available at https://jenkins-ci.org), extended with specific plugins to manage the various tasks that make up a Forensic Investigation, and the results of each task go as appropriate to the database, ElasticSearch, /reverse, /whitelist, /artifacts and so on.

In addition to installing Jenkins, the installer performed the following tasks:

- changed the group ownership of shared directories to jenkins, and ensured all directories are group writeable (otherwise our jobs would fail mysteriously due to Unix permissions issues);
- updated the sudo configuration to allow jenkins to run commands as root without supplying a password (so that for example required filesystem mounts succeed);
- installed the required Jenkins plugins that allow us to build complex job sequences with dependencies without hard-coding the values of case names and other variables (namely parameterized-trigger and conditional-buildstep and its dependencies);

Barry Anderson, shori@bigpond.net.au

- installed the 20 jobs in Appendix D and some views to provide discrete sets of jobs – one each for filesystem and memory analysis, one for the helper tasks for IOCs and the NSRL whitelist, and one for "Find Evidence";

After the script completes you should be able to point a browser at the URL: http://172.16.223.159:8080 (where 172.16.223.159 is the IP address of the SIFT workstation you have just installed Jenkins on) and see something like Figure 7, below.

Before proceeding with using Jenkins, under Manage Jenkins > Configure System:

- under Shell, set the value of "Shell Executable" to /bin/bash;
- set # of executors to a larger number than the default, such as 9;

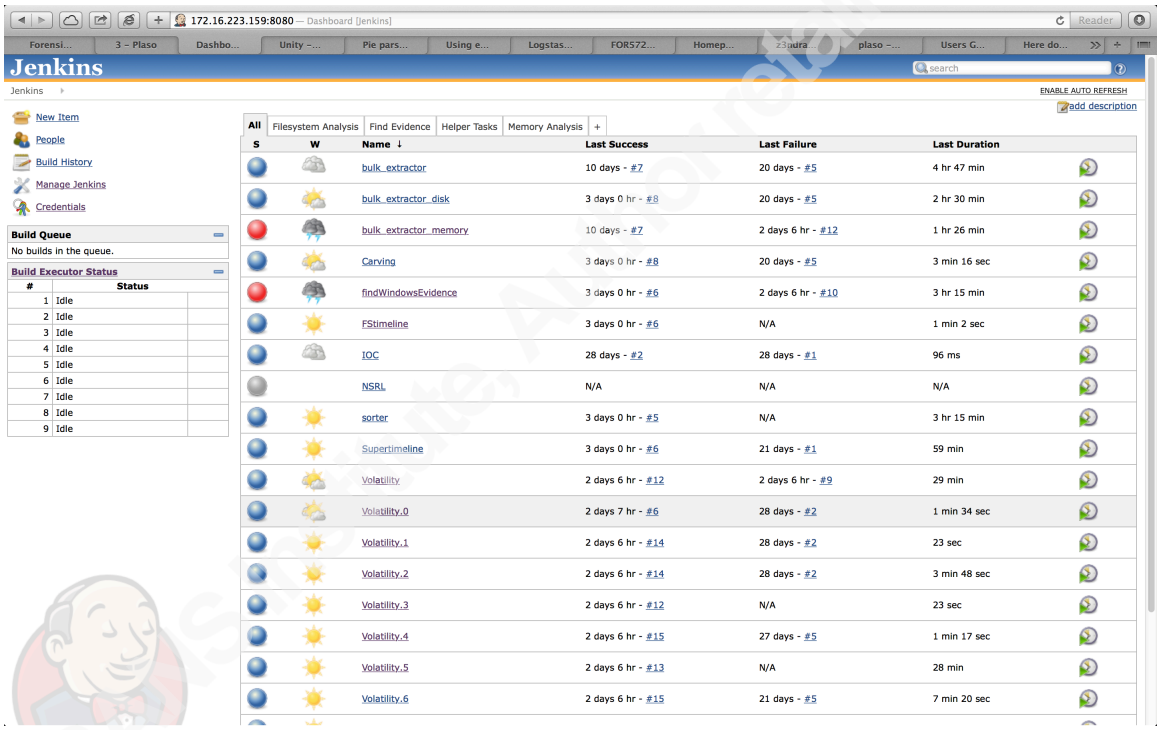

Figure 7 - The Jenkins dashboard

You can see from the dashboard above that findWindowsEvidence failed the last time it was run (the red ball to the left, in the 'S' or Status column). This was because one of the tasks it spawns (bulk_extractor_memory) failed. (This in turn is because bulk_extractor is built without a library it requires to be restartable, and the evidence directory from a previous run had not been deleted.) You can also see in the 'W' (or Weather) column that these jobs tend to be problematic – the Weather column is simply a visual indicator of how many of the last five runs of this job have failed.

Barry Anderson, shori@bigpond.net.au

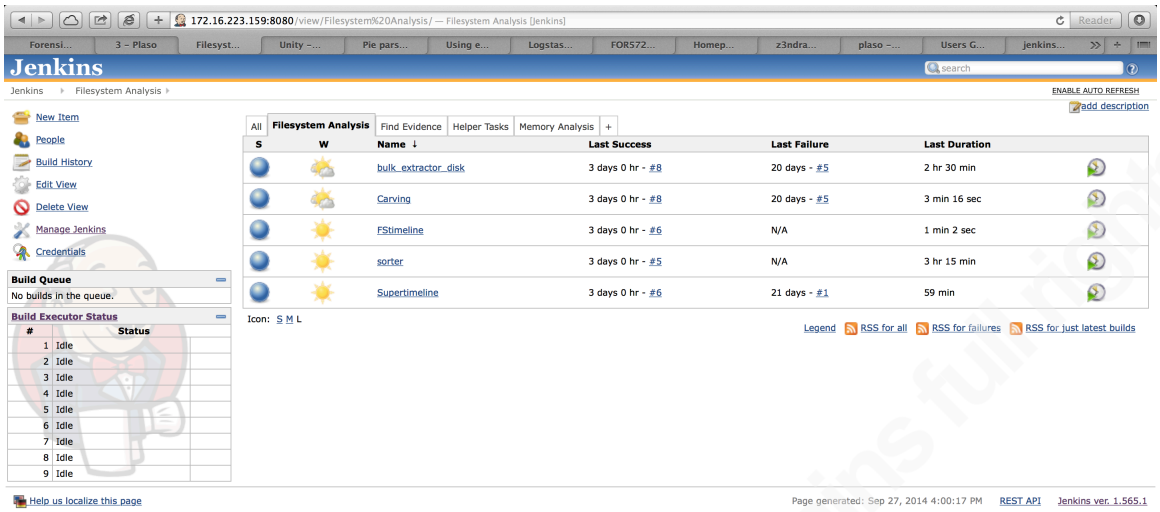

Figure 8 - Filesystem Analysis tasks

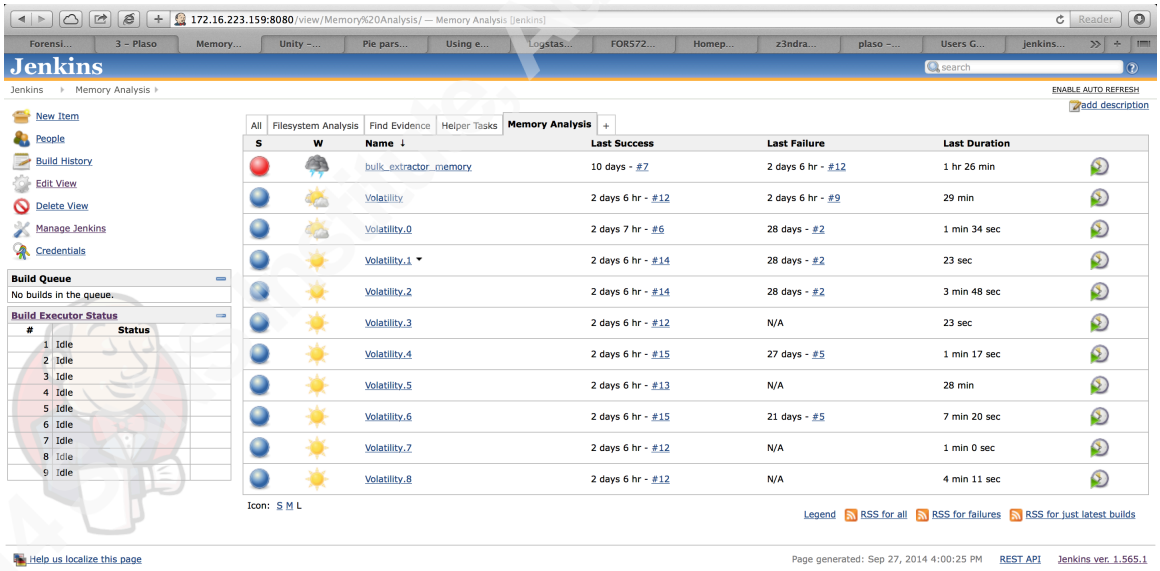

Figure 9 - Memory Analysis tasks

Barry Anderson, shori@bigpond.net.au

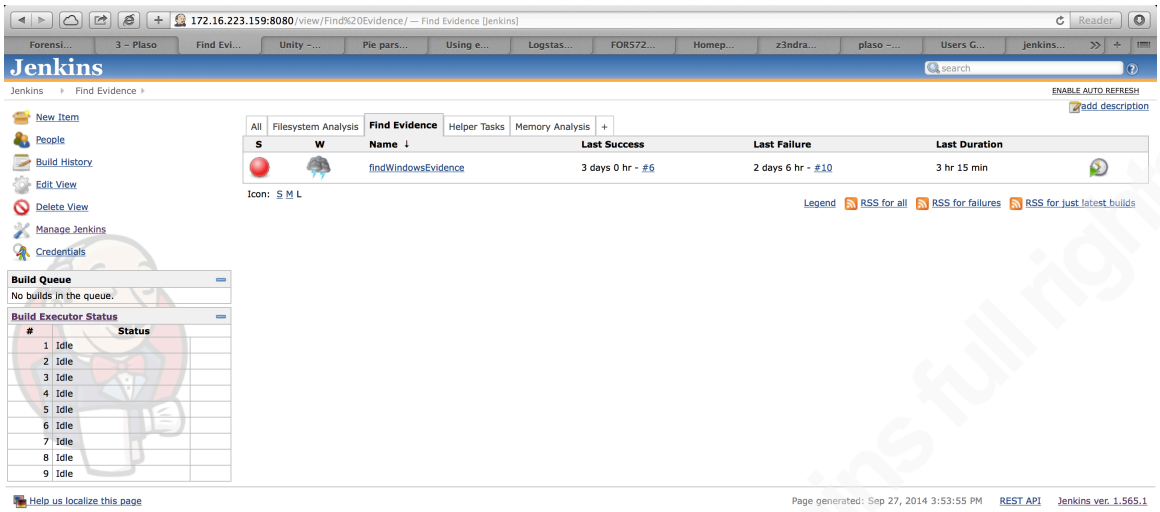

Figure 10 - The Find Evidence view

Now to schedule a particular task, click on the clock with the arrow over on the right hand side, and set the parameters for the task you're working on, as per Figure 11, below.

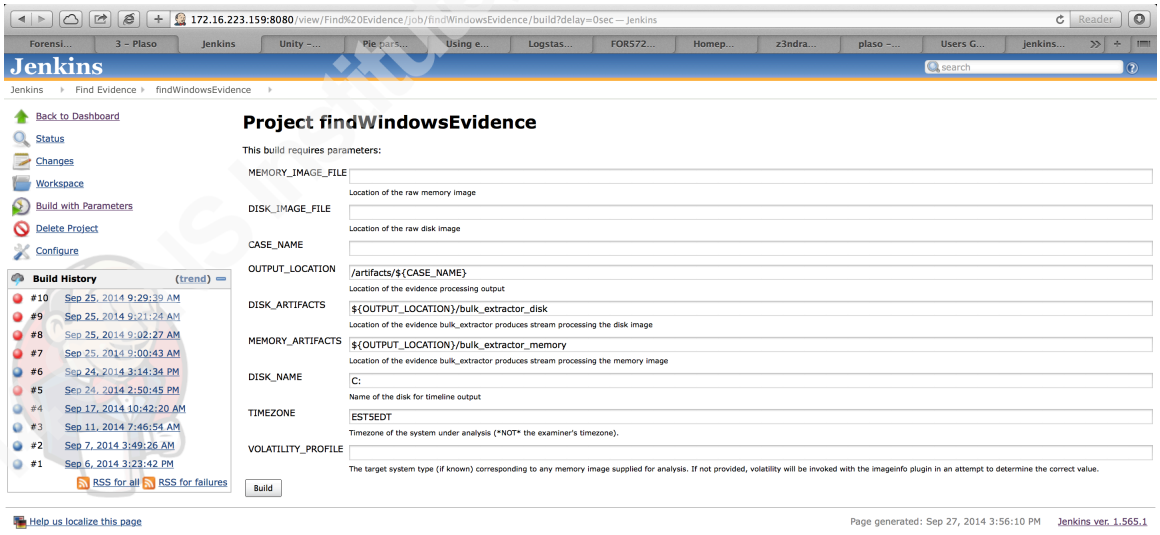

Figure 11 – Scheduling findWindowsEvidence

Barry Anderson, shori@bigpond.net.au

Performing a standard Volatility memory analysis is now as simple as scheduling the Jenkins Volatility job as described above, likewise a Filesystem timeline or Supertimeline.

The most useful job however, for a Windows case, is findWindowsEvidence. Think of the "Find Evidence" button in Forensicator Pro (though just for Windows). If you provide only the path to a disk image, it will perform only the filesystem analysis tasks; provide only a memory image, and it will perform only the memory analysis tasks; provide both, and it will perform all analysis tasks.

If you know the Volatility memory profile of the image, you can provide that here, but if you don't the Volatility job will even run the `vol.py imageinfo` command and parse the output, providing a "best guess" at the memory profile for you!

3.3. Structured Data

Structured data goes in a database. This means that all case metadata is available to everyone on the team, all the time, from anywhere, in exactly the same way that placing the timeline data in ElasticSearch makes it available.

PostgreSQL is our chosen database – though we could just as easily use MySQL, Oracle or any other relational database with Python language bindings.

3.4. Forensicator FATE – a DFIR Case Manager

Everything you need to see, nothing you don't. This provides the holistic view of the investigation, allowing our neophyte DFIR Engineer to create new cases and "hit the Find Evidence button."

To enter a new case, you simply enter:
Case Name (a unique identifier for the case - this will almost certainly be the subject of an naming standard);
Memory Image (the location of the Memory Image file on the SIFT workstation filesystem – this should be under a directory named with the case name under /cases);
Disk Image Image (the location of the Disk Image file on the SIFT workstation filesystem – this should be under a directory named with the case name under /cases);
Disk Name – a Drive Identifier;
Timezone – the timezone of the system under examination (**not** the examiner's);
(Optional) Volatility Profile – if not given the Jenkins Volatility job will attempt to work it out;
(Optional) Notes;
(Optional) Keywords;

Barry Anderson, shori@bigpond.net.au

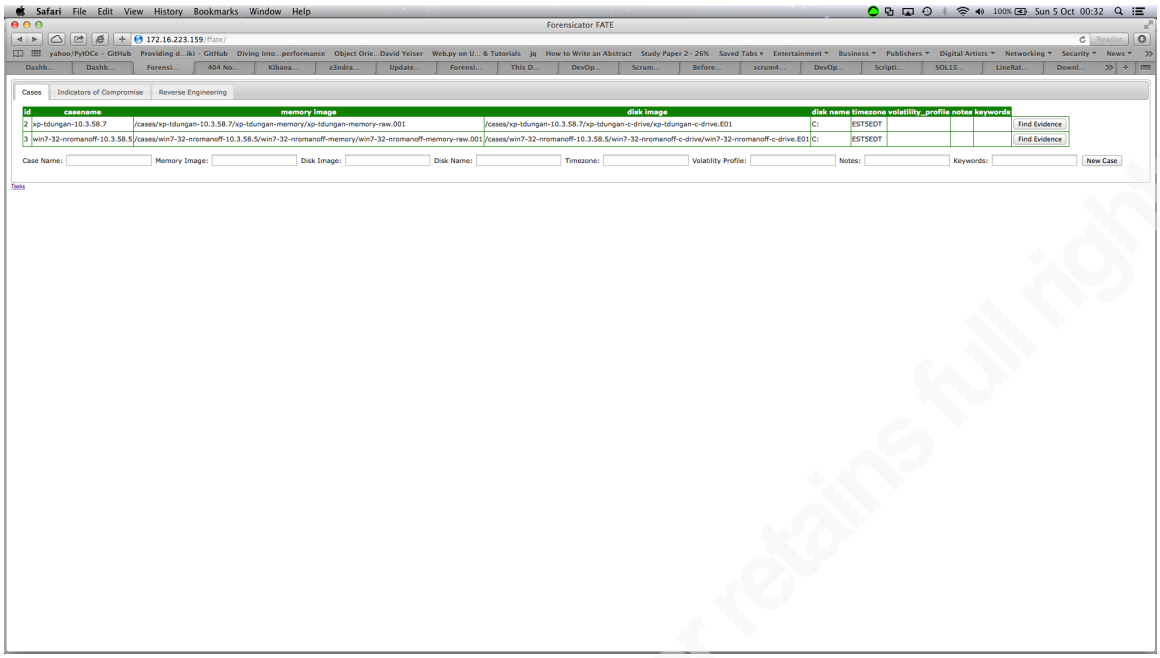

Figure 12, the Forensicator FATE Case Manager with two cases entered

To start processing the evidence for a case, simply click the "Find Evidence" button beside that case. You can bring up the Jenkins interface to keep an eye on the status of jobs in another browser window by clicking the small "Tasks" link under the tabs.

The other two tabs, Indicators of Compromise and Reverse Engineering are for assisted views of the contents of those shares. Unlike /artifacts for example, which you can browse around until you find the sorter output for a case, the names of things in /ioc are less than helpful (we want to know what an IOC is **for**, not what its UUID is).

The contents of /reverse, on the other hand, we want to protect our neophyte engineer from clicking on. All they actually need is the hash of any binaries we're suspicious of in order to be able to search http://fileadvisor.bit9.com or http://virustotal.com and searching those sites by hash doesn't give away our knowledge of attacker TTP's, unlike uploading a discovered binary.

If you don't understand how uploading to Virustotal may tip your hand, consider that if an attacker has used an exploit never before seen in the wild attacking you, all they need to do is poll Virustotal with the hash of their exploit, and as soon as there's a match, they know that their exploit has been discovered, and the defenders are using Virustotal to attempt to analyse it. Conversely, for the defender, search-by-hash allows us to see if we're dealing with known malware without disclosing to an attacker that they've been discovered.

Barry Anderson, shori@bigpond.net.au

4. What we have built – and where to from here?

We've now built a Digital Forensics Laboratory with an automatable, scalable, repeatable process framework, as well as an extremely lightweight Case Manager that allows a neophyte DFIR engineer to process the evidence for a case. What's next?

More automation – Still to come are further automation of the investigation (beyond the evidence processing stage), for example IOC generation; Mac Forensics evidence processing automation (findMacEvidence); Linux Forensics evidence processing automation (findLinuxEvidence); Mobile Forensics Evidence processing automation (findMobileEvidence); Network Forensics evidence processing automation (findNetworkEvidence); and an overarching findEvidence task.

Color – the colorized supertimeline visualization is a highly effective way of seeing patterns in large volumes of data quickly. While ELK is a superior solution for storing huge volumes of data, the colorized view of the timeline is seen as a desirable feature for the Kibana dashboard.

Packets never lie - in this paper, only the most cursory attention has been paid to Network Forensics, the value thereof, and integrating evidence found on the network with evidence found on the systems under investigation.

Cloudy with a chance of DFIR - if your workload is variable, but you don't want to tie up valuable capital or funding provisioning your laboratory for peak workloads, Infrastructure As A Service (IAAS) may be the answer. With IAAS, capacity is scalable on demand (as is essentially the case for most well-configured virtualization solutions), but more importantly you pay only for what you use. The CloudFormation scripts to stand up a virtual DFIR lab on AWS will be provided.

Compartmentalization – the described setup, like the SIFT workstation, makes certain assumptions about the environment in which is deployed. These assumptions may not hold as cooperation is made easier and more widespread. An expert consulted for their expertise on one case may not have clearance to view another case's details. Likewise, an expert consulted for their particular expertise may not have clearance to view all the details even of the case for which they are consulted.

These all require updates to the work done in this paper, and while some (FOR572 Network Forensics Integration for example) may be suitable topics for future Gold Papers, others may simply be done with a view to building on the capability developed herein. Accordingly update instructions may be found at the author's blog at http://c6i.blogspot.com/ or at GitHub in the project repository at https://github.com/z3ndrag0n/forensicator-fate.

Barry Anderson, shori@bigpond.net.au

5. References

Anderson, Barry (2014). Endarkenment. Personal blog site: http://c6i.blogspot.com/

Anderson, Barry (2014). Bringing DevOps to Forensics. Retrieved September 18, 2014, from GitHub Web site: https://github.com/z3ndrag0n/forensicator-fate

Croy, R. Tyler (2014). Jenkins. Retrieved April 14, 2014 from Jenkins CI Web site: https://jenkins-ci.org

Hagen, Phil (2014). The ELK (Elasticsearch, Logstash, Kibana) VM used in SANS FOR572, Advanced Network Forensics and Analysis. Retrieved August 20, 2014, from GitHub Web site: http://bit.ly/for572-logstash-readme

Kristensen, Erik (2014). SANS Investigative Forensics Toolkit Bootstrap Script. Retrieved September 6, 2014, from GitHub Web site: https://github.com/sans-dfir/sift-bootstrap

Lee, Rob (2014). The SIFT Workstation 3.0. Retrieved August 7, 2014, from SANS Digital Forensics Web site: https://digital-forensics31.sans.org/community/download-sift-kit/3.0.

Lee, Rob. *Forensics 508 Advanced Computer Forensic Analysis and Incident Response, 508.1 Enterprise Incident Response*, Bethesda, MD: SANS Institute

Lee, Rob. *Forensics 508 Advanced Computer Forensic Analysis and Incident Response, 508.2 Incident Response and Memory Analysis*, Bethesda, MD: SANS Institute

Lee, Rob. *Forensics 508 Advanced Computer Forensic Analysis and Incident Response, 508.3 Timeline Analysis*, Bethesda, MD: SANS Institute

Lee, Rob. *Forensics 508 Advanced Computer Forensic Analysis and Incident Response, 508.4 Deep-Dive Forensics and Anti-Forensics*, Bethesda, MD: SANS Institute

Lee, Rob. *Forensics 508 Advanced Computer Forensic Analysis and Incident Response, 508.5 (Part 1) Detection & Intrusion Forensics – The Art Of Finding Unknown Malware*, Bethesda, MD: SANS Institute

Lee, Rob. *Forensics 508 Advanced Computer Forensic Analysis and Incident Response, Workbook - Advanced Computer Forensic Analysis and Incident Response*, Bethesda, MD: SANS Institute

Barry Anderson, shori@bigpond.net.au

MANDIANT (2013). OpenIOC – An Open Framework for Sharing Threat Intelligence. Retrieved August 20, 2014, from OpenIOC Web site: http://www.openioc.org/

MI2 Security (2012). The Malware Management Framework. Retrieved September 21, 2014, from Malware Management Framework web site: http://sniperforensicstoolkit.squarespace.com/malwaremanagementframework.

Mueller, Ernest (2011). What Is Devops? | the agile admin. Retrieved September 30, 2014, from the agile admin Web site: http://theagileadmin.com/what-is-devops/.

Nelson-Smith, Stephen (2010). What Is This Devops Thing, Anyway? Retrieved, September 30, 2014, from Just Enough Developed Infrastructure blog: http://www.jedi.be/blog/2010/02/12/what-is-this-devops-thing-anyway/.

Rose, Erik (2012). "python elasticsearch client". Retrieved September 19, 2014 from GitHub web site: https://github.com/rhec/pyelasticsearch

Smart, John Ferguson. *Jenkins: The Definitive Guide*, Sebastopol, CA:O'Reilly

Swartz, Aaron (2012). Welcome to web.py! Retrieved August 20, 2014 from web.py Web site: http://webpy.org/

Barry Anderson, shori@bigpond.net.au

6. Appendices

6.1. Appendix A – The Forensicator FATE bundle installer

The following is the content of forensicator-fate/scripts/ffate-bundle-installer.sh

```
#!/bin/sh

forensicator-fate/scripts/install-pyelasticsearch.sh
forensicator-fate/scripts/create-shares.sh
forensicator-fate/scripts/install-jenkins.sh
forensicator-fate/scripts/install-pg.sh
forensicator-fate/scripts/install-ffate.sh
```

6.2. Appendix B – Install pyelasticsearch

The following is the content of forensicator-fate/scripts/install-pyelasticsearch.sh

```
#!/bin/sh
# Make sure plaso is up-to-date before continuing (thanks BMG)
sudo apt-get update
sudo apt-get install python-plaso
# Download and install python library to interact with Elasticsearch
git clone https://github.com/rhec/pyelasticsearch.git
cd pyelasticsearch
python setup.py build
sudo python setup.py install
```

6.3. Appendix C – additional shares for the SIFT workstation

The following is the content of forensicator-fate/scripts/create-shares.sh

```
#!/bin/sh
sudo mkdir /ioc /blacklist /whitelist /reverse /artifacts
sudo mv /etc/samba/smb.conf /etc/samba/smb.conf-default
sudo su - root -c "cat >/etc/samba/smb.conf-fate" <<EOF

[ioc]
        path = /ioc
        writeable = yes
        browseable = yes
        guest ok = yes

[blacklist]
        path = /blacklist
        writeable = yes
        browseable = yes
        guest ok = yes

[whitelist]
        path = /whitelist
        writeable = yes
        browseable = yes
        guest ok = yes

[reverse]
        path = /reverse
        writeable = yes
        browseable = yes
        guest ok = yes

[artifacts]
        path = /artifacts
```

Barry Anderson, shori@bigpond.net.au

```
        writeable = yes
        browseable = yes
        guest ok = yes
EOF
sudo su - root -c "cat /etc/samba/smb.conf-default /etc/samba/smb.conf-fate
>/etc/samba/smb.conf"
sudo service smbd restart
```


Barry Anderson, shori@bigpond.net.au

6.4. Appendix D – Install Jenkins

The following is the content of forensicator-fate/scripts/install-jenkins.sh

```
#!/bin/sh
sudo apt-get install Jenkins
#update Volatility to 2.4
sudo apt-get install python-volatility
sudo chgrp jenkins /ioc /blacklist /whitelist /reverse /artifacts
sudo chmod g+w /ioc /blacklist /whitelist /reverse /artifacts
sudo su - root -c 'echo "%jenkins ALL=(ALL:ALL) NOPASSWD:ALL" >/etc/sudoers.d/jenkins'
sudo chmod 440 /etc/sudoers.d/jenkins
sudo adduser sansforensics jenkins

#from lucabelmondo on github (comment on https://gist.github.com/rowan-m/1026918)
wget -O default.js http://updates.jenkins-ci.org/update-center.json
sed '1d;$d' default.js > default.json
sudo mkdir /var/lib/jenkins/updates
sudo mv default.json /var/lib/jenkins/updates/
sudo chown -R jenkins:jenkins /var/lib/jenkins/updates
sudo service jenkins restart

#update Jenkins
cd /usr/share/jenkins
sudo mv jenkins.war jenkins-last-install.war
sudo curl -o jenkins.war -L http://updates.jenkins-ci.org/download/war/latest/jenkins.war
cd ~
sudo service jenkins restart

#wait for Jenkins to start
until wget http://localhost:8080/ 2>&1 | grep "response" | grep -c "200 OK"; do echo
Sleeping 15 seconds waiting for Jenkins to start... ; sleep 15 ; done

#install plugins
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 install-
plugin parameterized-trigger
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 install-
plugin conditional-buildstep
sudo service jenkins restart

#wait for Jenkins to start
until wget http://localhost:8080/ 2>&1 | grep "response" | grep -c "200 OK"; do echo
Sleeping 15 seconds waiting for Jenkins to start... ; sleep 15 ; done

#install jobs
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
bulk_extractor_disk <forensicator-fate/jenkins/jobs/bulk_extractor_disk.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
bulk_extractor_memory <forensicator-fate/jenkins/jobs/bulk_extractor_memory.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
bulk_extractor <forensicator-fate/jenkins/jobs/bulk_extractor.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Carving <forensicator-fate/jenkins/jobs/Carving.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
FStimeline <forensicator-fate/jenkins/jobs/FStimeline.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
IOC <forensicator-fate/jenkins/jobs/IOC.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
NSRL <forensicator-fate/jenkins/jobs/NSRL.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
sorter <forensicator-fate/jenkins/jobs/sorter.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Supertimeline <forensicator-fate/jenkins/jobs/Supertimeline.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.0 <forensicator-fate/jenkins/jobs/Volatility.0.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.1 <forensicator-fate/jenkins/jobs/Volatility.1.xml
```

Barry Anderson, shori@bigpond.net.au

```
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.2 <forensicator-fate/jenkins/jobs/Volatility.2.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.3 <forensicator-fate/jenkins/jobs/Volatility.3.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.4 <forensicator-fate/jenkins/jobs/Volatility.4.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.5 <forensicator-fate/jenkins/jobs/Volatility.5.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.6 <forensicator-fate/jenkins/jobs/Volatility.6.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.7 <forensicator-fate/jenkins/jobs/Volatility.7.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility.8 <forensicator-fate/jenkins/jobs/Volatility.8.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
Volatility <forensicator-fate/jenkins/jobs/Volatility.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-job
findWindowsEvidence <forensicator-fate/jenkins/jobs/findWindowsEvidence.xml

#install views
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-view
"Filesystem Analysis" <forensicator-fate/jenkins/views/FSAnalysis.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-view
"Memory Analysis" <forensicator-fate/jenkins/views/MemoryAnalysis.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-view
"Helper Tasks" <forensicator-fate/jenkins/views/HelperTasks.xml
java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s http://localhost:8080 create-view
"Find Evidence" <forensicator-fate/jenkins/views/ForensicatorFATE.xml

sudo cp forensicator-fate/scripts/guess_profile.pl /usr/bin
sudo chown jenkins:jenkins /usr/bin/guess_profile.pl
sudo chmod 755 /usr/bin/guess_profile.pl
```

Barry Anderson, shori@bigpond.net.au

6.5. Appendix E – Jenkins jobs and views

The following command line lists all Jenkins jobs:

```
sansforensics@sift1:~$ java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s
http://localhost:8080 list-jobs
```

This is the complete list of jobs that Jenkins should know about:

```
bulk_extractor
bulk_extractor_disk
bulk_extractor_memory
Carving
findWindowsEvidence
FStimeline
IOC
NSRL
sorter
Supertimeline
Volatility
Volatility.0
Volatility.1
Volatility.2
Volatility.3
Volatility.4
Volatility.5
Volatility.6
Volatility.7
Volatility.8
```

The following command imports the job NSRL from the NSRL.xml file:

```
$ java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s \
http://localhost:8080 create-job NSRL <NSRL.xml
```

The following command updates the job NSRL from the NSRL.xml file (which will come in handy if there have been updates to the Jenkins jobs which you've downloaded):

```
$ java -jar /run/jenkins/war/WEB-INF/jenkins-cli.jar -s \
http://localhost:8080 update-job NSRL <NSRL.xml
```

The contents of NSRL.xml (the Jenkins job that unzips the NSRL zipfile, moves the hashfile to /whitelist and builds the hfind indices as used by sorter) are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Unzip the NSRL zip found in the provided directory;&#xd;
move the resulting file to /whitelist;&#xd;
produce the indices;</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>NSRL_DOWNLOAD_DIRECTORY</name>
          <description>The directory the NSRLFile.txt.zip archive has been downloaded
to.</description>
          <defaultValue>/home/sansforensics/Downloads</defaultValue>
        </hudson.model.StringParameterDefinition>
```

Barry Anderson, shori@bigpond.net.au

```
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>false</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command>cd /tmp &amp;&amp; unzip ${NSRL_DOWNLOAD_DIRECTORY}/NSRLFile.txt.zip
&amp;&amp; mv NSRLFile.txt /whitelist &amp;&amp; cd /whitelist &amp;&amp; hfind -i nsrl-
md5 NSRLFile.txt &amp;&amp; hfind -i nsrl-sha1 NSRLFile.txt</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

The contents of IOC.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Copy downloaded (or created!) Indicators of Compromise found in the
provided directory to /ioc;</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>IOC_DOWNLOAD_DIRECTORY</name>
          <description>The directory the .ioc files have been downloaded to or created
in.</description>
          <defaultValue>/home/sansforensics/Downloads</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>false</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command>cp ${IOC_DOWNLOAD_DIRECTORY}/*.ioc /ioc</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.0.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 0 - Identify the memory image profile</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Path of the memory image for volatility to process.</description>
          <defaultValue>/cases/sift408pc-memory.img</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location for the volatility output</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
vol.py imageinfo &gt;${OUTPUT_LOCATION}/imageinfo_output</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.1.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 1 - Identify Rogue Processes</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Path of the memory image for volatility to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location for volatility output.</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
vol.py pslist &gt;${OUTPUT_LOCATION}/pslist_output
vol.py psscan &gt;${OUTPUT_LOCATION}/psscan_output
vol.py pstree &gt;${OUTPUT_LOCATION}/pstree_output</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.2.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 2 - Analyze Process DLLs and Handles</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Pat of the raw memory image.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of the evidence processing output</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
vol.py dlllist &gt;${OUTPUT_LOCATION}/dlllist_output
vol.py getsids &gt;${OUTPUT_LOCATION}/getsids_output
vol.py handles &gt;${OUTPUT_LOCATION}/handles_output
vol.py filescan &gt;${OUTPUT_LOCATION}/filescan_output
vol.py mutantscan &gt;${OUTPUT_LOCATION}/mutantscan_output
vol.py svcscan &gt;${OUTPUT_LOCATION}/svcscan_output
vol.py cmdscan &gt;${OUTPUT_LOCATION}/cmdscan_output
vol.py consoles &gt;${OUTPUT_LOCATION}/consoles_output</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.3.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 3 - Review Network Artifacts</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Location of the raw memory image.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
if  [[ $VOLATILITY_PROFILE =~ &quot;WinXP&quot; ]] ; then vol.py connections
&gt;${OUTPUT_LOCATION}/connections_output &amp;&amp; vol.py connscan
&gt;${OUTPUT_LOCATION}/connscan_output &amp;&amp; vol.py sockets
&gt;${OUTPUT_LOCATION}/sockets_output &amp;&amp; vol.py sockscan
&gt;${OUTPUT_LOCATION}/sockscan_output; elif [[ $VOLATILITY_PROFILE =~ &quot;Win2K3&quot;
]] ; then vol.py connections &gt;${OUTPUT_LOCATION}/connections_output &amp;&amp; vol.py
connscan &gt;${OUTPUT_LOCATION}/connscan_output &amp;&amp; vol.py sockets
&gt;${OUTPUT_LOCATION}/sockets_output &amp;&amp; vol.py sockscan
&gt;${OUTPUT_LOCATION}/sockscan_output; elif [[ $VOLATILITY_PROFILE =~ &quot;Win&quot; ]]
; then vol.py netscan &gt;${OUTPUT_LOCATION}/netscan_output; else echo &quot;Unsupported
Memory Image Profile to Review Network Artifacts&quot;; fi</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.4.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 4 - Look for Evidence of Code Injection</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Location of the raw memory image</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
vol.py malfind --dump-dir ${OUTPUT_LOCATION} &gt;${OUTPUT_LOCATION}/malfind_output
vol.py ldrmodules -v &gt; ${OUTPUT_LOCATION}/ldrmodules_output</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.5.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 5 - Check for Signs of a Rootkit</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Location of the raw memory image</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used for building output paths</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of the evidence processing output.</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
vol.py psxview &gt;${OUTPUT_LOCATION}/psxview_output
vol.py modscan &gt;${OUTPUT_LOCATION}/modscan_output
vol.py apihooks &gt;${OUTPUT_LOCATION}/apihooks_output
vol.py ssdt &gt;${OUTPUT_LOCATION}/ssdt_output
vol.py driverirp &gt;${OUTPUT_LOCATION}/driverirp_output
vol.py idt &gt;${OUTPUT_LOCATION}/idt_output</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.6.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 6 - Dump Suspicious Processes and Drivers</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Location of the raw memory image.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of the evidence processing output.</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_OUTPUT_LOCATION</name>
          <description></description>
          <defaultValue>${OUTPUT_LOCATION}/volatility-dumps</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
if  [[ ! -d $VOLATILITY_OUTPUT_LOCATION ]] ; then
mkdir $VOLATILITY_OUTPUT_LOCATION
fi
vol.py dlldump --dump-dir ${VOLATILITY_OUTPUT_LOCATION}
&gt;${OUTPUT_LOCATION}/dlldump_output
vol.py moddump --dump-dir ${VOLATILITY_OUTPUT_LOCATION}
&gt;${OUTPUT_LOCATION}/moddump_output
vol.py procdump --dump-dir ${VOLATILITY_OUTPUT_LOCATION}
&gt;${OUTPUT_LOCATION}/procdump_output
vol.py memdump --dump-dir ${VOLATILITY_OUTPUT_LOCATION}
&gt;${OUTPUT_LOCATION}/memdump_output</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.7.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 7 - Registry Analysis</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Location of the raw memory image</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of the evidence processing output.</description>
          <defaultValue>/artifacts/{CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
vol.py hivelist &gt;${OUTPUT_LOCATION}/hivelist_output
vol.py userassist &gt;${OUTPUT_LOCATION}/userassist_output</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.8.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Volatility - Step 8 - Memory Timelining</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Location of the raw memory image.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of the evidence processing output.</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
vol.py timeliner --output=body &gt;${OUTPUT_LOCATION}/timeliner_bodyfile_output</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Volatility.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Perform a Volatility Memory Analysis</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description></description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_LOCATION</name>
          <description>Specify the location of the memory image to analyse.</description>
          <defaultValue>file://${MEMORY_IMAGE_FILE}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>Specify the profile of the memory image to analyse.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description></description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <org.jenkinsci.plugins.conditionalbuildstep.ConditionalBuilder plugin="conditional-
buildstep@1.3.3">
      <runner class="org.jenkins_ci.plugins.run_condition.BuildStepRunner$Run"
plugin="run-condition@1.0"/>
      <runCondition class="org.jenkins_ci.plugins.run_condition.core.ExpressionCondition"
plugin="run-condition@1.0">
        <expression>^$</expression>
        <label>$VOLATILITY_PROFILE</label>
      </runCondition>
      <conditionalbuilders>
        <hudson.tasks.Shell>
          <command>vol.py imageinfo | guess_profile.pl</command>
        </hudson.tasks.Shell>
      </conditionalbuilders>
    </org.jenkinsci.plugins.conditionalbuildstep.ConditionalBuilder>
    <hudson.plugins.parameterizedtrigger.TriggerBuilder plugin="parameterized-
trigger@2.25">
      <configs>
        <hudson.plugins.parameterizedtrigger.BlockableBuildTriggerConfig>
          <configs>
            <hudson.plugins.parameterizedtrigger.CurrentBuildParameters/>
            <hudson.plugins.parameterizedtrigger.FileBuildParameters>
              <propertiesFile>imageinfo.properties</propertiesFile>
              <failTriggerOnMissing>true</failTriggerOnMissing>
```

Barry Anderson, shori@bigpond.net.au

```
              <useMatrixChild>false</useMatrixChild>
              <onlyExactRuns>false</onlyExactRuns>
            </hudson.plugins.parameterizedtrigger.FileBuildParameters>
          </configs>

<projects>Volatility.1,Volatility.2,Volatility.3,Volatility.4,Volatility.5,Volatility.6,V
olatility.7,Volatility.8</projects>
          <condition>ALWAYS</condition>
          <triggerWithNoParameters>false</triggerWithNoParameters>
          <block>
            <buildStepFailureThreshold>
              <name>FAILURE</name>
              <ordinal>2</ordinal>
              <color>RED</color>
              <completeBuild>true</completeBuild>
            </buildStepFailureThreshold>
            <unstableThreshold>
              <name>UNSTABLE</name>
              <ordinal>1</ordinal>
              <color>YELLOW</color>
              <completeBuild>true</completeBuild>
            </unstableThreshold>
            <failureThreshold>
              <name>FAILURE</name>
              <ordinal>2</ordinal>
              <color>RED</color>
              <completeBuild>true</completeBuild>
            </failureThreshold>
          </block>
          <buildAllNodesWithLabel>false</buildAllNodesWithLabel>
        </hudson.plugins.parameterizedtrigger.BlockableBuildTriggerConfig>
      </configs>
    </hudson.plugins.parameterizedtrigger.TriggerBuilder>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Carving.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Runs blkls to collect unallocated storage, then determines the cluster
size from the disk image with fsstat, then runs foremost with the appropriate parameters
to carve out deleted files.</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>DISK_IMAGE_FILE</name>
          <description></description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>UNALLOCATED_STORAGE_FILE</name>
          <description></description>
          <defaultValue>${OUTPUT_LOCATION}/${CASE_NAME}.blkls</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build the paths to output files
to.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of evidence processing artifacts</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>FOREMOST_OUTPUT</name>
          <description></description>
          <defaultValue>${OUTPUT_LOCATION}/foremost</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>false</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
blkls $DISK_IMAGE_FILE &gt;$UNALLOCATED_STORAGE_FILE
CLUSTER_SIZE=`fsstat $DISK_IMAGE_FILE | awk &apos;/Cluster Size: /{print $3}&apos;`
foremost -q -b $CLUSTER_SIZE -o $FOREMOST_OUTPUT $UNALLOCATED_STORAGE_FILE</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of FStimeline.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Create a filesystem timeline with fls and mactime for
analysis</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>DISK_IMAGE_FILE</name>
          <description>Location of the disk image to create the filesystem timeline
from</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>DISK_NAME</name>
          <description>Name of the disk for output</description>
          <defaultValue>C:</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build the output path.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of the timeline output file</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>TIMEZONE</name>
          <description>Timezone of the system under analysis</description>
          <defaultValue>EST5EDT</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
fls -r -m $DISK_NAME $DISK_IMAGE_FILE &gt;${OUTPUT_LOCATION}/bodyfile  &amp;amp;&amp;amp; mactime
-d -b ${OUTPUT_LOCATION}/bodyfile -z $TIMEZONE &gt;${OUTPUT_LOCATION}/fs-
timeline.csv</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of Supertimeline.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Create a Supertimeline with plaso for analysis</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>DISK_IMAGE_FILE</name>
          <description>Location of the disk image to create the supertimeline
from</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build the output path.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of the supertimeline output file</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>TIMEZONE</name>
          <description>Timezone of the system being analysed.</description>
          <defaultValue>EST5EDT</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
log2timeline.py ${OUTPUT_LOCATION}/plaso.dump $DISK_IMAGE_FILE
psort.py -z$TIMEZONE -o L2tcsv -w ${OUTPUT_LOCATION}/supertimeline.csv
${OUTPUT_LOCATION}/plaso.dump</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of bulk_extractor_disk.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description></description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>DISK_IMAGE_FILE</name>
          <description></description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>KEYWORDS_FILE</name>
          <description></description>
          <defaultValue>/cases/${CASE_NAME}/keywords.txt</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>DISK_ARTIFACTS</name>
          <description></description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>false</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
if [[ ! -f $KEYWORDS_FILE ]]; then touch $KEYWORDS_FILE; fi
bulk_extractor -F $KEYWORDS_FILE -o $DISK_ARTIFACTS $DISK_IMAGE_FILE</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of bulk_extractor_memory.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description></description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description></description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>KEYWORDS_FILE</name>
          <description></description>
          <defaultValue>/cases/${CASE_NAME}/keywords.txt</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_ARTIFACTS</name>
          <description></description>
          <defaultValue>${OUTPUT_LOCATION}/bulk_extractor_memory</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description></description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description>Name of the case - used to build output paths.</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>false</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
if [[ ! -f $KEYWORDS_FILE ]]; then touch $KEYWORDS_FILE; fi
bulk_extractor -F $KEYWORDS_FILE -e net -e aes -e wordlist -o $MEMORY_ARTIFACTS
$MEMORY_IMAGE_FILE</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of bulk_extractor.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>Stream Analysis - Runs Simson Garfinkel&amp;apos;s bulk_Extractor against both
Memory and Disk Image</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Location of the raw memory image</description>
          <defaultValue>/cases/xp-tdungan-10.3.58.7/xp-tdungan-memory/xp-tdungan-memory-
raw.001</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>DISK_IMAGE_FILE</name>
          <description>Location of the raw disk image</description>
          <defaultValue>/cases/xp-tdungan-10.3.58.7/xp-tdungan-c-drive/xp-tdungan-c-
drive.E01</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>DISK_ARTIFACTS</name>
          <description></description>
          <defaultValue>/artifacts/xp-tdungan-
10.3.58.7/bulk_extractor_disk</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_ARTIFACTS</name>
          <description></description>
          <defaultValue>/artifacts/xp-tdungan-
10.3.58.7/bulk_extractor_memory</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <hudson.plugins.parameterizedtrigger.TriggerBuilder plugin="parameterized-
trigger@2.25">
      <configs>
        <hudson.plugins.parameterizedtrigger.BlockableBuildTriggerConfig>
          <configs>
            <hudson.plugins.parameterizedtrigger.CurrentBuildParameters/>
          </configs>
          <projects>bulk_extractor_memory,bulk_extractor_disk</projects>
          <condition>ALWAYS</condition>
          <triggerWithNoParameters>false</triggerWithNoParameters>
          <block>
            <buildStepFailureThreshold>
              <name>FAILURE</name>
              <ordinal>2</ordinal>
              <color>RED</color>
              <completeBuild>true</completeBuild>
            </buildStepFailureThreshold>
            <unstableThreshold>
              <name>UNSTABLE</name>
              <ordinal>1</ordinal>
              <color>YELLOW</color>
              <completeBuild>true</completeBuild>
            </unstableThreshold>
            <failureThreshold>
```

Barry Anderson, shori@bigpond.net.au

```
            <name>FAILURE</name>
            <ordinal>2</ordinal>
            <color>RED</color>
            <completeBuild>true</completeBuild>
          </failureThreshold>
        </block>
        <buildAllNodesWithLabel>false</buildAllNodesWithLabel>
      </hudson.plugins.parameterizedtrigger.BlockableBuildTriggerConfig>
    </configs>
  </hudson.plugins.parameterizedtrigger.TriggerBuilder>
 </builders>
 <publishers/>
 <buildWrappers/>
</project>
```


Barry Anderson, shori@bigpond.net.au

The contents of sorter.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description></description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>DISK_IMAGE_FILE</name>
          <description></description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>SORTER_ARTIFACTS</name>
          <description></description>
          <defaultValue>${OUTPUT_LOCATION}/sorter</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>DISK_NAME</name>
          <description>Name of the disk for output.</description>
          <defaultValue>C:</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>BLACKLIST</name>
          <description>A list of hashes of files known to be bad</description>
          <defaultValue>/blacklist/known_bad_files.txt</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>WHITELIST</name>
          <description>A list of hashes of files we trust.</description>
          <defaultValue>/whitelist/known_good_files.txt</defaultValue>
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>false</concurrentBuild>
  <builders>
    <hudson.tasks.Shell>
      <command> if [[ ! -d $OUTPUT_LOCATION ]]; then mkdir $OUTPUT_LOCATION; fi
if  [[ ! -d $SORTER_ARTIFACTS ]] ; then
mkdir $SORTER_ARTIFACTS
fi
if [[ -f /whitelist/NSRLFile.txt ]]; then
sorter -s -h -n /whitelist/NSRLFile.txt -a $BLACKLIST -x $WHITELIST -m ${DISK_NAME} -d
$SORTER_ARTIFACTS $DISK_IMAGE_FILE
else
sorter -s -h -a $BLACKLIST -x $WHITELIST -m ${DISK_NAME} -d $SORTER_ARTIFACTS
$DISK_IMAGE_FILE
fi</command>
    </hudson.tasks.Shell>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

Barry Anderson, shori@bigpond.net.au

The contents of findWindowsEvidence.xml are:

```
<?xml version='1.0' encoding='UTF-8'?>
<project>
  <actions/>
  <description>The Forensicator Pro &amp;quot;Find Evidence&amp;quot; Button as a Jenkins job -
but only for Windows.</description>
  <keepDependencies>false</keepDependencies>
  <properties>
    <hudson.model.ParametersDefinitionProperty>
      <parameterDefinitions>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_IMAGE_FILE</name>
          <description>Location of the raw memory image</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>DISK_IMAGE_FILE</name>
          <description>Location of the raw disk image</description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>CASE_NAME</name>
          <description></description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>OUTPUT_LOCATION</name>
          <description>Location of the evidence processing output</description>
          <defaultValue>/artifacts/${CASE_NAME}</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>DISK_ARTIFACTS</name>
          <description>Location of the evidence bulk_extractor produces stream processing
the disk image</description>
          <defaultValue>${OUTPUT_LOCATION}/bulk_extractor_disk</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>MEMORY_ARTIFACTS</name>
          <description>Location of the evidence bulk_extractor produces stream processing
the memory image</description>
          <defaultValue>${OUTPUT_LOCATION}/bulk_extractor_memory</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>DISK_NAME</name>
          <description>Name of the disk for timeline output</description>
          <defaultValue>C:</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>TIMEZONE</name>
          <description>Timezone of the system under analysis (*NOT* the examiner&amp;apos;s
timezone).</description>
          <defaultValue>EST5EDT</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>VOLATILITY_PROFILE</name>
          <description>The target system type (if known) corresponding to any memory
image supplied for analysis. If not provided, volatility will be invoked with the
imageinfo plugin in an attempt to determine the correct value. </description>
          <defaultValue></defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>WHITELIST</name>
          <description>List of hashes of files we trust.</description>
          <defaultValue>/whitelist/known_good_files.txt</defaultValue>
        </hudson.model.StringParameterDefinition>
        <hudson.model.StringParameterDefinition>
          <name>BLACKLIST</name>
          <description>List of hashes of known bad files.</description>
          <defaultValue>/blacklist/known_bad_files.txt</defaultValue>
```

Barry Anderson, shori@bigpond.net.au

```
        </hudson.model.StringParameterDefinition>
      </parameterDefinitions>
    </hudson.model.ParametersDefinitionProperty>
  </properties>
  <scm class="hudson.scm.NullSCM"/>
  <canRoam>true</canRoam>
  <disabled>false</disabled>
  <blockBuildWhenDownstreamBuilding>false</blockBuildWhenDownstreamBuilding>
  <blockBuildWhenUpstreamBuilding>false</blockBuildWhenUpstreamBuilding>
  <triggers/>
  <concurrentBuild>true</concurrentBuild>
  <builders>
    <org.jenkinsci.plugins.conditionalbuildstep.ConditionalBuilder plugin="conditional-
buildstep@1.3.3">
      <runner class="org.jenkins_ci.plugins.run_condition.BuildStepRunner$DontRun"
plugin="run-condition@1.0"/>
      <runCondition class="org.jenkins_ci.plugins.run_condition.logic.Not" plugin="run-
condition@1.0">
        <condition class="org.jenkins_ci.plugins.run_condition.core.ExpressionCondition">
          <expression>^$</expression>
          <label>${MEMORY_IMAGE_FILE}</label>
        </condition>
      </runCondition>
      <conditionalbuilders>
        <hudson.plugins.parameterizedtrigger.TriggerBuilder plugin="parameterized-
trigger@2.25">
          <configs>
            <hudson.plugins.parameterizedtrigger.BlockableBuildTriggerConfig>
              <configs>
                <hudson.plugins.parameterizedtrigger.CurrentBuildParameters/>
              </configs>
              <projects>Volatility,bulk_extractor_memory</projects>
              <condition>ALWAYS</condition>
              <triggerWithNoParameters>false</triggerWithNoParameters>
              <block>
                <buildStepFailureThreshold>
                  <name>FAILURE</name>
                  <ordinal>2</ordinal>
                  <color>RED</color>
                  <completeBuild>true</completeBuild>
                </buildStepFailureThreshold>
                <unstableThreshold>
                  <name>UNSTABLE</name>
                  <ordinal>1</ordinal>
                  <color>YELLOW</color>
                  <completeBuild>true</completeBuild>
                </unstableThreshold>
                <failureThreshold>
                  <name>FAILURE</name>
                  <ordinal>2</ordinal>
                  <color>RED</color>
                  <completeBuild>true</completeBuild>
                </failureThreshold>
              </block>
              <buildAllNodesWithLabel>false</buildAllNodesWithLabel>
            </hudson.plugins.parameterizedtrigger.BlockableBuildTriggerConfig>
          </configs>
        </hudson.plugins.parameterizedtrigger.TriggerBuilder>
      </conditionalbuilders>
    </org.jenkinsci.plugins.conditionalbuildstep.ConditionalBuilder>
    <org.jenkinsci.plugins.conditionalbuildstep.ConditionalBuilder plugin="conditional-
buildstep@1.3.3">
      <runner class="org.jenkins_ci.plugins.run_condition.BuildStepRunner$DontRun"
plugin="run-condition@1.0"/>
      <runCondition class="org.jenkins_ci.plugins.run_condition.logic.Not" plugin="run-
condition@1.0">
        <condition class="org.jenkins_ci.plugins.run_condition.core.ExpressionCondition">
          <expression>^$</expression>
          <label>${DISK_IMAGE_FILE}</label>
        </condition>
      </runCondition>
      <conditionalbuilders>
```

Barry Anderson, shori@bigpond.net.au

```
        <hudson.plugins.parameterizedtrigger.TriggerBuilder plugin="parameterized-
trigger@2.25">
          <configs>
            <hudson.plugins.parameterizedtrigger.BlockableBuildTriggerConfig>
              <configs>
                <hudson.plugins.parameterizedtrigger.CurrentBuildParameters/>
              </configs>

<projects>FStimeline,Supertimeline,sorter,bulk_extractor_disk,Carving</projects>
              <condition>ALWAYS</condition>
              <triggerWithNoParameters>false</triggerWithNoParameters>
              <block>
                <buildStepFailureThreshold>
                  <name>FAILURE</name>
                  <ordinal>2</ordinal>
                  <color>RED</color>
                  <completeBuild>true</completeBuild>
                </buildStepFailureThreshold>
                <unstableThreshold>
                  <name>UNSTABLE</name>
                  <ordinal>1</ordinal>
                  <color>YELLOW</color>
                  <completeBuild>true</completeBuild>
                </unstableThreshold>
                <failureThreshold>
                  <name>FAILURE</name>
                  <ordinal>2</ordinal>
                  <color>RED</color>
                  <completeBuild>true</completeBuild>
                </failureThreshold>
              </block>
              <buildAllNodesWithLabel>false</buildAllNodesWithLabel>
            </hudson.plugins.parameterizedtrigger.BlockableBuildTriggerConfig>
          </configs>
        </hudson.plugins.parameterizedtrigger.TriggerBuilder>
      </conditionalbuilders>
    </org.jenkinsci.plugins.conditionalbuildstep.ConditionalBuilder>
  </builders>
  <publishers/>
  <buildWrappers/>
</project>
```

The contents of guess_profile.pl are:

```
#!/usr/bin/perl
while(<>) {
    if (/Suggested Profile\(s\) \: (.*)$/) {
        $profiles = $1;
        @profiles = split / |,/, $profiles;
    }
    elsif (/Image Type \(Service Pack\) \: (.*)$/) {
        $profile = $1;
    }
}

open(my $fh, ">", "imageinfo.properties") or die "cannot open > imageinfo.properties:
$!";
print $fh "VOLATILITY_PROFILE = ", grep(/SP$profile/,@profiles), "\n";
close $fh;
```

Barry Anderson, shori@bigpond.net.au

6.6. Appendix F – Installing PostgreSQL, the Python bindings, a database, role and table

The following is the content of forensicator-fate/scripts/install-pg.sh

```
#!/bin/sh
sudo apt-get install postgresql python-psycopg2
sudo service postgresql start
sudo su - postgres -c psql <<EOF
create role webpy;
create database webpy;
grant all on database webpy to webpy;
alter role webpy with login;
EOF
sudo ed /etc/postgresql/9.1/main/pg_hba.conf <<EOF
/^local.*all.*all.*peer$
s/peer/trust/
w
q
EOF
sudo service postgresql restart
psql -U webpy -d webpy <<EOF
create table cases (id SERIAL, casename varchar, memory_image varchar, disk_image
varchar, disk_name varchar, timezone varchar, volatility_profile varchar, notes varchar,
case_keywords varchar);
\q
EOF
```

Barry Anderson, shori@bigpond.net.au

6.7. Appendix G – Installing Apache, WSGI, web.py and Forensicator FATE

The following is the content of forensicator-fate/scripts/install-ffate.sh

```
#!/bin/sh
sudo easy_install web.py
sudo apt-get install apache2
sudo apt-get install libapache2-mod-wsgi
sudo a2enmod rewrite wsgi

sudo service apache2 restart

sudo ed /etc/apache2/sites-available/default <<EOF
/ErrorLog
i
WSGIScriptAlias /ffate /var/www/forensicator-fate/ffate.py
Alias /static /var/www/public_html

<Directory /var/www/forensicator-fate>
  SetHandler wsgi-script
  Options ExecCGI FollowSymLinks
</Directory>

AddType text/html .py

<Location />
#  RewriteEngine on
#  RewriteBase /
#  RewriteCond %{REQUEST_URI} !^/static
#  RewriteCond %{REQUEST_URI} !^(/.*)+ffate.py/
#  RewriteRule ^(.*)$ ffate.py/$1 [PT]
</Location>
.
w
q
EOF

wget http://code.jquery.com/jquery-1.11.1.min.js
wget http://jqueryui.com/resources/download/jquery-ui-1.11.1.zip
unzip jquery-ui-1.11.1.zip
wget http://jqueryui.com/resources/download/jquery-ui-themes-1.11.1.zip
unzip jquery-ui-themes-1.11.1.zip

sudo mkdir /var/www/forensicator-fate
sudo mkdir /var/www/public_html
sudo mv /var/www/index.html /var/www/public_html

sudo mv jquery-1.11.1.min.js /var/www/public_html/jquery.js
sudo mv jquery-ui-1.11.1/jquery-ui.min.js /var/www/public_html/jquery-ui.js
sudo mv jquery-ui-1.11.1/jquery-ui.theme.min.css /var/www/public_html/jquery-ui.theme.css
sudo mv jquery-ui-1.11.1/jquery-ui.min.css /var/www/public_html/jquery-ui.css
sudo mv jquery-ui-1.11.1/jquery-ui.structure.min.css /var/www/public_html/jquery-
ui.structure.css
sudo mv jquery-ui-1.11.1/images/ /var/www/public_html/
sudo mv jquery-ui-themes-1.11.1/themes/ /var/www/public_html/

sudo cp forensicator-fate/webapp/ffate.css /var/www/public_html/
sudo cp forensicator-fate/webapp/ffate.py /var/www/forensicator-fate/
sudo cp -R forensicator-fate/webapp/templates /var/www/templates

sudo service apache2 restart
```

Barry Anderson, shori@bigpond.net.au

6.8. Appendix G – The Forensicator FATE Web Application

The following is the content of forensicator-fate/webapp/ffate.py

```
import web

urls = (
    '/', 'index',
    '/cases', 'cases',
    '/search', 'search',
    '/add','add'
)

db = web.database(dbn='postgres', db='webpy', user='webpy', pw='')

render = web.template.render('/var/www/templates')

web.template.Template.globals.update(dict(
  render = render
))


class index:
    def GET(self):
        jenkins_url = web.ctx.homedomain + ":8080/"
        return render.tabbed("Forensicator FATE", jenkins_url)

class cases:
    def GET(self):
        return render.cases()

class add:
    def POST(self):
        i = web.input()
        n = db.insert('cases',
casename=i.casename,memory_image=i.memory_image,disk_image=i.disk_image,disk_name=i.disk_
name,timezone=i.timezone,volatility_profile=i.volatility_profile,notes=i.notes,case_keywo
rds=i.case_keywords)
        raise web.seeother('/')

class search:
    def POST(self):
        i = web.input()
        print i
        return render.listing(db.select('cases',what='*',order='id',limit=10000))
    def GET(self):
        jenkins_url = web.ctx.homedomain + ":8080/"
        jenkins_job_url = jenkins_url + "job/findWindowsEvidence/buildWithParameters"
        return render.listing(db.select('cases',what='*',order='id',limit=10000),
jenkins_job_url)


if __name__ == "__main__":
    app.run()


app = web.application(urls, globals())
application = app.wsgifunc()
```

Barry Anderson, shori@bigpond.net.au

The following is the content of forensicator-fate/webapp/ffate.css

```
body {
  height: 100%;
  font-family: "Trebuchet MS", "Helvetica", "Arial",  "Verdana", "sans-serif";
  font-size: 62.5%;
}

iframe[name="frame2"] {
  width: 95%;
  height: auto !important;
  min-height: 600;
  //background-color: #666699;
}

table, td, th {
  border: 1px solid green;
  border-collapse: collapse;
}

th {
  background-color: green;
  color: white;
}
```

The following is the content of forensicator-fate/webapp/templates/tabbed.html

```
$def with (title, jenkins_url)
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
  <head>

    <title>
$if title: $title
    </title>

    <link rel="stylesheet" href="/static/themes/smoothness/jquery-ui.css">
    <link href="/static/ffate.css" rel="Stylesheet" type="text/css">
    <script type="text/javascript" src="/static/jquery.js"></script>
    <script type="text/javascript" src="/static/jquery-ui.js"></script>
    <script>
$$(function() {
$$( "#tabs" ).tabs({
 beforeLoad: function( event, ui ) {
    ui.jqXHR.error(function() {
    ui.panel.html(
      "Couldn't load this tab." +
      'Please <a href="https://github.com/z3ndrag0n/forensicator-fate/issues/"
target="new">Report</a> this.' );
  });
 }
  });
});
    </script>


  </head>
  <body>
<div id="tabs">
  <ul>
    <li><a href="search">Cases</a></li>
<!--    <li><a href="/artifacts">Artifacts</a></li>-->
    <li><a href="ioc">Indicators of Compromise</a></li>
    <li><a href="ffate/hashdir.py">Reverse Engineering</a></li>
  </ul>


<!--
  <div id="tabs-1">
```

Barry Anderson, shori@bigpond.net.au

```
    <p>Tab 1</p>
  </div>
-->
</div>

    <a href="$jenkins_url" target="new">Tasks</a>
  </body>
</html>
```

The following is the content of forensicator-fate/webapp/templates/listing.html

```
$def with (cases, jenkins_job_url)
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
  <head>
    <link href="/static/ffate.css" rel="Stylesheet" type="text/css">
  </head>
  <body>

$if cases:
    <table>
    <th>id </th><th>casename </th><th>memory image </th><th>disk image </th><th>disk name
</th><th>timezone </th><th>volatility_profile </th><th>notes </th><th>keywords </th>

    $for case in cases:
        $:render.li(case,jenkins_job_url)
    </table>

<form method="post" action="add">
<p>Case Name: <input type="text" name="casename" />
Memory Image: <input type="text" name="memory_image" />
Disk Image: <input type="text" name="disk_image" />
Disk Name: <input type="text" name="disk_name" />
Timezone: <input type="text" name="timezone" />
Volatility Profile: <input type="text" name="volatility_profile" />
Notes: <input type="text" name="notes" />
Keywords: <input type="text" name="case_keywords" />
<input type="submit" value="New Case" /></p>
</form>

</body>
```

The following is the content of forensicator-fate/webapp/templates/li.html

```
$def with (case, jenkins_job_url)

<tr id="$case.id">
<form action="$jenkins_job_url">

<td>$case.id </td><td><input type="hidden" name="CASE_NAME"
value="$case.casename">$case.casename </td><td><input type="hidden"
name="MEMORY_IMAGE_FILE" value="$case.memory_image">$case.memory_image </td><td><input
type="hidden" name="DISK_IMAGE_FILE" value="$case.disk_image">$case.disk_image
</td><td><input type="hidden" name="DISK_NAME" value="$case.disk_name">$case.disk_name
</td><td><input type="hidden" name="TIMEZONE" value="$case.timezone">$case.timezone
</td><td><input type="hidden" name="VOLATILITY_PROFILE"
value="$case.volatility_profile">$case.volatility_profile </td><td>$case.notes
</td><td>$case.case_keywords </td>
<td><input type="submit" value="Find Evidence"></td>
</form>
</tr>
```

Barry Anderson, shori@bigpond.net.au