Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Incident Identification Through Long Tail Analysis

GIAC (GCUX) Gold Certification

Author: Joshua Lewis, Joshua.d.Lewis@gmail.com
Advisor: Rob VandenBrink
Accepted: February 1st, 2016

Template Version September 2014

Abstract

Incident identification is one of the most challenging steps in the incident response process. If an incident cannot be detected, it cannot be contained. Long tail analysis is an effective incident detection method to supplement existing incident detection techniques. The long tail analysis framework utilizes a simple assumption that compromised hosts represent a subset of the enterprise asset population. In a homogenous environment, where all assets are nearly identical, differences between systems can be used to identify potentially compromised assets. Realistically, most organizations have a heterogeneous environment and struggle with configuration drift, resulting in a limited ability to easily compare systems. This research extends the Kansa Windows PowerShell incident response framework to Linux operating systems, and utilizes long tail analysis to detect outliers across homogenous and heterogeneous environments.

© 2015 The SANS Institute Author retains full rights.

1. Background

Standardization of enterprise IT infrastructure provides significant benefits such as a consistent user experience, reduced maintenance effort, and lower support costs. Utilizing an enterprise IT architecture to obtain standardization, business use cases can be consistently translated into reusable implementation patterns. Unfortunately, many enterprise IT teams struggle to find the resources necessary to build a comprehensive architecture, keep pace with the relentless technology evolution, remain connected to constantly changing business use cases, and ensure consistent adoption of the technology standards. Additional IT resources or cloud based deployment models, such as Platform as a Service, may partially alleviate some of the standardization challenges. However, the reality is that a significant number of heterogeneous systems will continue to exist for the foreseeable future.

The lack of standardization of the IT environment not only complicates administration, it impairs the enterprise security posture. One of the most significant side effects of a non-standard IT environment is a reduced ability to detect security incidents. In a homogenous environment, anomaly analysis tools can be reliably deployed to monitor deviations from the baseline. In a heterogeneous environment, there is no baseline or consistent normal state, causing anomaly analysis tools to produce significant false positives.

Beyond anomaly analysis tools, the operating system security tool ecosystem generally consists of three categories of capabilities: configuration management, file integrity monitoring, and signature based detection. Configuration management tools such as Puppet and Chef have been developed to assist with the application of IT and security standards. These tools apply standards based on an established baseline and can eliminate configuration drift. However, some organizations may not have the necessary financial resources or understanding of each business system to identify the implications of the configuration changes. The second operating system security capability category focuses on file integrity monitoring. File integrity monitoring solutions such as AIDE and Tripwire monitor critical system configuration files and binaries. These tools provide the ability to alert on sophisticated zero-day exploits or stealthy configuration changes by comparing the operational state to a known checksum baseline. However,

© 2015 The SANS Institute Author retains full rights.

implementing file integrity monitoring post system deployment in a non-standard environment may result in the whitelisting of potentially malicious configurations or binaries. The third operating system security capability category focuses on signature detection. Signature based detection identifies malicious actions by recognizing a set of previously seen characteristics or patterns that were determined to be malicious. Signature based detection is typically very reliable and can help classify an incident. However, signature based detection is easily foiled, does not alert on patterns that have not been previously analyzed, and is difficult to obtain a comprehensive signature coverage for all malicious operating system and application permutations in a heterogeneous environment.

Configuration management, file integrity monitoring, and signature detection tools provide a substantial ability to detect sophisticated incidents. The use of these capabilities in non-standard environments reduces the effectiveness and can be difficult to administrate. However, long tail analysis can be utilized to establish a baseline for heterogeneous environments and verify the baseline for homogeneous environments. With an accurate baseline, anomaly analysis tools can be reliably utilized to supplement the existing incident detection approaches. Long tail analysis operates on the premise that malicious configurations or binaries should be less frequent than benign configurations or binaries. Therefore, forensic artifacts can be collected across the enterprise environment, sorted based on the number of occurrences, and analyzed to determine outliers that may indicate an incident. Long tail analysis is an effective method to determine outliers, however a security analyst must have a firm grasp of the forensic artifacts that are being collected to truly confirm if the outlier is a false positive or an incident. The identification of outliers through long tail is helpful for organizations that do not have a strong configuration management capability, and can identify a malicious state that has been whitelisted. From an attacker's standpoint, long tail analysis can be thwarted by compromising large portions of the environment, but this will result in an increased footprint and greater chance of detection through other incident identification techniques.

Long tail anomaly analysis is a good supplement, not a replacement, for existing incident identification approaches. The Kansa Incident Response Framework (Hull,

© 2015 The SANS Institute Author retains full rights.

2014) pioneered the use of long tail analysis to identify outliers. This framework uses a series of PowerShell scripts to collect forensic artifacts from Windows operating systems. Each artifact can be collected in parallel leveraging PowerShell remoting, which enables simultaneous artifact collection across massive host populations. After the artifacts are collected, the results of each artifact are merged together and can be opened in Microsoft Excel or imported into a database for long tail sorting and outlier identification. At the time this research was conducted, the author was unaware of any open source or commercial products that provided a similar capability for Linux or Unix operating systems.

The goal of this research is to champion the use of long tail analysis as a simple and effective incident detection and investigation prioritization technique. Additionally, this research extends the Kansa Incident Response Framework capabilities to Linux and Unix operating systems. Specifically, this research is laser focused on parallel OS and application forensic artifact collection, parsing, and outlier identification for Red Hat and Cent OS. Red Hat and Cent OS were chosen based on their prominent enterprise market share and popularity. However, this framework can be easily ported to other Linux or Unix distributions. Additionally, network based long tail analysis will not be covered within the scope of this research.

2. Long tail analysis framework

2.1. Artifact collection phase: conceptual design

The long tail analysis framework is divided into two phases, the artifact collection (section 2.1) and the artifact analysis phase (section 2.2). Each phase of the framework is described in the context of a conceptual design to introduce the high level processes and considerations. Beyond the conceptual design, proof-of-concept scripts are included appendix 5.1.2 - 5.1.5. Reference Figure 1 for a graphical view of the overall long tail analysis framework.

© 2015 The SANS Institute Author retains full rights.

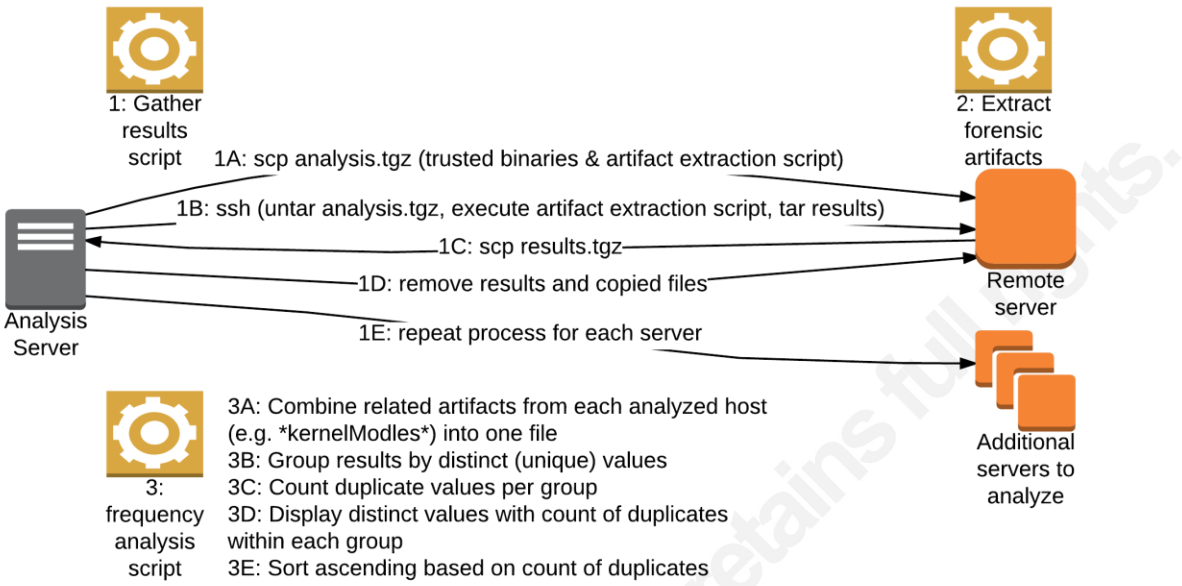

Figure 1 - Long tail analysis framework overview

The artifact collection phase is supported by two processes, the artifact extraction (2.1.1) and the artifact gathering process (2.1.2). These processes are designed to extract forensic artifacts from a population of remote systems and gather the extracted artifacts from each remote host for analysis on a central server.

2.1.1. Artifact extraction process

The first step in the artifact extraction process is to determine what forensic artifacts should be collected. The initial selection of forensic artifacts should be based on the operating systems and applications (e.g. Sendmail, Apache, Bind, etc.) used in the organization. Within each operating system and application, artifacts should be identified that indicate exploitation, persistence, or privilege escalation. Some forensic artifacts may not be suitable for long tail analysis (e.g. the analysis of logic within a script). Therefore, long tail analysis should supplement, not replace, existing incident detection techniques. Additionally, long tail analysis is not intended for incident investigation, and should not be utilized to collect evidence such as memory or disk images. An example of forensic artifacts for Red Hat and Cent OS operating systems are listed in Table 1. Appendix 5.1.1 provides an explanation for each artifact listed in Table 1, to include commands utilized to extract the artifact, and what to look for in the output.

© 2015 The SANS Institute Author retains full rights.

Table 1: Example forensic artifacts for long tail analysis on Red Hat and Cent OS

Forensic artifact	Category of actions detected
List of listening ports	Persistence
List of running processes	Exploitation, persistence
List of services and service state	Persistence
List of kernel modules	Persistence
List of setuid and setgid files	Privilege escalation
List of open but unlinked files	Persistence
List of regular files in the /dev directory	Persistence
List of files that start with dots or spaces	Persistence
List of directories that start with dots	Persistence
List and hash of files in bin directories	Privilege escalation, persistence
List of immutable files	Persistence
List of crontab entries	Persistence
List and hash of scripts executed by run-parts	Persistence
List of users allowed to make changes to cron	Persistence
List of user accounts	Persistence
List of user accounts with null passwords	Persistence
List of uid 0 user accounts	Persistence
List of SSH authorized_keys for the root account	Persistence
Hash of sudoers file	Persistence, privilege escalation
Pam configuration	Persistence
SSHd configuration	Persistence
List of .rhost files	Persistence
List of hosts.equiv files	Persistence
List of X*.hosts files	Persistence
List default run level and init entries	Persistence
List and hash of run level 3&5 startup and kill scripts	Persistence

© 2015 The SANS Institute Author retains full rights.

Hash of rc.local file	Persistence
RPM package verification	Exploitation, persistence, privilege escalation

After the forensic artifacts have been selected for the long tail analysis investigation, the next step is to ensure that the output of these artifacts can be trusted. Attackers have been known to replace operating system binaries that are used for incident identification (e.g. the ps command) or modify binary shared system libraries to hide their malicious artifacts. Rootkits that replace binaries or system libraries can be subverted by temporarily copying statically or dynamically linked binaries from the analysis server to the remote endpoint to run the artifact extraction commands. These statically or dynamically linked binaries are intended to be used solely by the artifact extraction script, do not overwrite the native operating system binaries, and are deleted after the results are extracted. Statically linked binaries can be created using the gcc –static flag (Wright, 2010). However statically linked binaries are operating system and kernel version specific, and would require that each binary be statically compiled with each operating system and kernel version in an enterprise (Pogue, Altheide, & Haverkos, 2008). Depending on the operating system and kernel version standardization in the environment, two options can be utilized. The first option is to statically compile all forensic artifact extraction binaries for each operating system and kernel version used in the enterprise environment. This option may be feasible for homogenous environments, and can provide a higher level of assurance. The second option is to compile dynamically linked forensic artifact extraction binaries for each operating system within the enterprise environment. This option may be more feasible for heterogeneous environments, but provides a lower level of assurance since the shared libraries on the system being analyzed will be used in conjunction with the trusted binary. However, both options provide a higher level of assurance than utilizing the binaries and the shared libraries on the systems being analyzed. For example, the binaries need to extract the forensic artifacts from Table 1 are listed in Table 2.

© 2015 The SANS Institute Author retains full rights.

Table 2: Example binaries for forensic artifact extraction

Binaries need to extract forensic artifacts from Table 1	
netstat	cat
egrep	sha256sum
sed	lsmod
ps	rpm
chkconfig	lsattr
hostname	awk
find	tr
grep	echo
sh	sort
tar	rm

After the trusted dynamically or statically linked binaries have been used to obtain the artifacts, the last step in the artifact extraction process is to parse the results. The comparison of the forensic artifacts across multiple hosts requires the output files to utilize a consistent format. Leverage native command output options or tools, such as grep and awk, to remove unnecessary fields and headers that will cause long tail calculation errors. Additionally, utilize tools such as sed and tr to store the output in a consistent delimited format, such as a CSV, to easily compare the results and enable integration with other tools. Finally, append the hostname and artifact extraction run time to each line in the results. The hostname and run time columns can be ignored during the long tail calculations, but they can be used to quickly trace an outlier back to the original host that generated the artifact.

Based on the artifact selection, use of trusted binaries, and artifact parsing logic, the final step in this process is to automate the artifact extraction. Leverage a script to execute the trusted binaries, parse the output, and write the results of each artifact to a local or remote file. Writing files to local storage may overwrite forensic evidence in unallocated space. Consider writing the extracted artifact files to a remote share or server. Reference appendix 5.1.2 for an example artifact extraction script.

© 2015 The SANS Institute Author retains full rights.

2.1.2. Artifact gathering process

The second process within the artifact collection phase is the artifact gathering process. The artifact gathering process sets up the authentication to the remote endpoint, establishes the investigation methodology, and outlines the logic to automate the process with a script.

The first step in the artifact gathering process is to setup the authentication for the remote endpoints. Copy and remote command execution operations are orchestrated by SCP and SSH with strong public and private key authentication. The SCP and SSH public and private keys can be created using the ssh-keygen command. During the key generation process, ensure that a password is used to encrypt the private key at rest. After the ssh-keygen command completes, edit the public key to add restrictions for how the key can be used. For example, the public key will not be used for X11 forwarding, so the "no-X11-forwarding" option can be set to prevent this activity. After the public key has the appropriate options set, copy the key into the root account authorized_keys file on each remote endpoint that will be evaluated with the long tail analysis framework. Finally, leverage ssh-agent on the analysis server to load the decrypted private key in memory and automatically perform the authentication for each command. The ssh-agent program provides the ability to utilize encrypted private keys on disk, without continuously re-entering the password each time a command is executed. However, since ssh-agent stores a decrypted copy of the SSH private key in memory, ensure that the analysis server is a single role server that is appropriately hardened and has limited administrator access. Reference section 5.1.3 for an example implementation walkthrough for setting up SSH authentication and ssh-agent.

Once the authentication for the copy and remote command execution is setup, the next step is to determine the investigation methodology. The long tail analysis framework can be executed based on a targeted or macro methodology. Running the long tail analysis framework against a targeted set of similar assets (e.g. Web servers) provides the ability to spot service specific logic outliers. For example, a running Apache service may have a high rate of occurrence across the macro enterprise environment, however the Apache service should be disabled on DNS servers as part of the configuration hardening process. A low rate of occurrence of running Apache services

© 2015 The SANS Institute Author retains full rights.

across the DNS server population indicates a misconfiguration or a potential compromise. Inversely, running the long tail analysis framework on the macro enterprise assets provides a larger population that smooths out legitimate outliers. For example, an SSH client patch may be released and slowly applied to the enterprise assets. The long tail analysis framework will highlight the patched and unpatched SSH binary hashes as frequent, while a trojanized SSH binary on a compromised server will have a hash with a low rate of occurrence. Targeted and macro long tail analysis methodologies should both be used in conjunction to effectively spot outliers.

After the authentication and investigation methodology have been established, the final step is to automate the artifact gathering process with a script. Within the artifact gathering script, source the ssh-agent environment variables to be able to utilize ssh-agent to authenticate the SSH and SCP commands. Since the long tail analysis framework may be ran on a massive enterprise host population, the parallel-ssh (Chun & McNabb, 2012) program is used to invoke multiple instances of SSH and SCP at the same time. Parallel-ssh reads a text file containing a list of target host names, and can be configured to simultaneously connect to an arbitrary number of hosts. Next, ensure that a directory structure (e.g. /results/current) is setup on the analysis server to capture the extracted artifacts from each host. If the current directory already contains extracted artifacts from the host that is being evaluated, move the previous artifact results into a historical directory (e.g. /results/historical/<insert host name>/<insert last run time of analysis>/). Next, copy the trusted binaries and the artifact extraction files from the analysis server to the remote host. Execute the artifact extraction script on the remote host and copy the results back to the current results directory on the analysis server (e.g. /results/current). The last step in the artifact gathering script is to remove the artifact extraction script, trusted binaries, and results from the remote host. Copying the trusted binaries and artifact execution script to the remote host and deleting them each time the long tail analysis framework is run, reduces the probability that an attacker will be alerted that analysis activities are being conducted. Even upon detection, the short duration of this analysis limits their ability to modify the binaries or scripts in response. Reference section 5.1.4 for an example script that implements the results gathering process.

© 2015 The SANS Institute Author retains full rights.

2.2. Artifact analysis phase: conceptual design

The second phase of the long tail analysis framework focuses on analyzing the results of the extracted artifacts gathered from each of the remote endpoints. Using the analysis server, the first step in this process is to concatenate like artifacts (e.g. host1.localdomain_01-18-2016.08.25_kernelModules.csv, host2.localdomain_01-18-2016.08.25_kernelModules.csv, etc.) from the results of each host into a combined file (e.g. currentKernelModulesCombined.csv). Using the combined file, display the appropriate columns that should be used for the long tail calculation. The rate of occurrence of an artifact is calculated based the full contents of a line. Each extracted artifact line contains fields that should not be displayed for the long tail calculation (e.g. hostname, date, etc.), and are intended to provide the ability to trace an outlier back to a host for further investigation. After the appropriate artifacts are displayed, the data should be grouped based on duplicate lines. Next, the number of lines within the duplicate grouping is counted, the duplicated lines are removed, and unique lines are prefixed by the count of the lines within each duplicate group. Finally, the data can be sorted in ascending order to display the least frequent occurrences first. Since the combined artifact file is in a consistently delimited format (e.g. CSV), the rate of occurrence calculation can be performed with a script or using pivot tables in Microsoft Excel. Reference section 5.1.5 for an example script that implements the long tail analysis process.

After the artifact rate of occurrence has been calculated, the final step in the analysis phase is to interpret the results. Artifacts that have a low rate of occurrence do not immediately indicate a compromise. A low artifact rate of occurrence can stem from a small sample size that is not representative of the larger asset population, a configuration error, a one-off system, or a compromise. Long tail analysis is helpful to quickly spot outliers, where a large number of artifacts are present (e.g. kernel modules). However, long tail analysis merely helps focus investigation efforts, and an analyst needs to understand the underlying artifacts to differentiate a false positive from a compromise. Reference section 5.1.1 for walk through of common Red Hat and Cent OS artifact explanations.

© 2015 The SANS Institute Author retains full rights.

2.3. Long tail analysis limitations

The long tail analysis framework adds a significant capability to supplement existing incident detection methods. However, long tail analysis is not well suited for some types of artifacts such as script logic. Additionally, long tail analysis will not detect rootkits that load a kernel module to mask their existence from the forensic artifact collection tools used in this framework. The benefits and limitations of this framework should be considered based on the needs of the enterprise and the incident detection strategy.

3. Future research

The long tail analysis framework presented in this research establishes a parallel outlier analysis capability for Red Hat and Cent OS operating systems that can be scaled to massive enterprise asset populations. The example implementation scripts in appendix 5.1.2 - 5.1.5 make extensive use of binary path variables to limit the effort required to port this capability to other Linux or Unix variants. Additionally, each forensic artifact is captured in a CSV delimited format and enables data to easily be imported into a database for scalability and increased usability with a web application. Future research should focus on enhancing the long tail analysis framework by adding the capability to analyze multiple Linux or Unix distributions, integrating the capability with a database, and using the the output from multiple long tail analysis investigations to build a historical analysis capability to spot deviations over time.

©2016 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.

4. References

Chun, B., & McNabb, A. (2012, February 2). *parallel-ssh*. Retrieved from code.google.com: https://code.google.com/p/parallel-ssh/

Hull, D. (2014, June 2). *Kansa*. Retrieved from GitHub: https://github.com/davehull/Kansa

Katherine, W., & Joseph, M. (2001). *Mastering Unix.* Alameda, CA: Sybex Inc. Retrieved from http://csweb.cs.wfu.edu/~torgerse/Kokua/SGI/007-2862-005/sgi_html/ch05.html

Kerrisk, M. (2015, July 23). *Linux Programmer's Manual*. Retrieved from man7.org: http://man7.org/linux/man-pages/man5/hosts.equiv.5.html

Murdoch, D. (2014). *Blue Team Handbook: Incident Response Edition* (2.0 ed.).

Pogue, C., Altheide, C., & Haverkos, T. (2008). *Unix and Linux Forensic Analysis DVD Toolkit.* Burlington, MA: Syngress Publishing, Inc.

Pomeranz, H. (2015). *Hardening Linux/Unix Systems.* Eugene, OR: The SANS Intitute.

Red Hat. (n.d.). *13.2.4 Using Key-Based Authentication*. Retrieved from redhat.com: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/s2-ssh-configuration-keypairs.html

Red Hat. (n.d.). *19.10. Managing Hosts*. Retrieved from redhat.com: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/System_Administration_Guide/s1-network-config-hosts.html

Joshua Lewis, Joshua.D.Lewis@gmail.com

© 2015 The SANS Institute Author retains full rights.

Red Hat. (n.d.). *2.3. Configuring sudo Access*. Retrieved from redhat.com: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/2/html/Getting_Started_Guide/ch02s03.html

Red Hat. (n.d.). *Chapter 39. Automated Tasks*. Retrieved from redhat.com: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/ch-autotasks.html

Red Hat. (n.d.). *Chapter 40. Kernel Modules*. Retrieved from redhat.com: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/System_Administration_Guide/ch-kernel-modules.html

Red Hat. (n.d.). *Red Hat Enterprise Linux 3: Introduction to System Administration*. Retrieved from redhat.com: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Introduction_to_System_Administration/s1-acctsgrps-rhlspec.html

Red Hat. (n.d.). *Red Hat Enterprise Linux 3: Reference Guide*. Retrieved from redhat.com: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Reference_Guide/s1-boot-init-shutdown-sysv.html

rpm.org. (n.d.). *Maximum RPM: Taking the Red Hat Package Manager to the Limit*. Retrieved from rpm.org: http://www.rpm.org/max-rpm/s1-rpm-verify-what-to-verify.html

Joshua Lewis, Joshua.D.Lewis@gmail.com

© 2015 The SANS Institute Author retains full rights.

SANS Institute. (n.d.). *Intrusion Discovery Cheat Sheet for Linux*. Retrieved from sans.org: https://pen-testing.sans.org/resources/downloads

Wright, C. (2010, March 9). *Building a UNIX/Linux Incident response / Forensic Disk*. Retrieved from digital-forensics.sans.org: https://digital-forensics.sans.org/blog/2010/03/09/building-a-unixlinux-incident-response-forensic-disk/

Zeltser, L. (2015, January). *Security Incident Survey Cheat Sheet for Server Administrators*. Retrieved from zeltser.com: https://zeltser.com/security-incident-survey-cheat-sheet/

©2016 SANS Institute, Author retains full rights.

Joshua Lewis, Joshua.D.Lewis@gmail.com

© 2015 The SANS Institute Author retains full rights.

5. Appendix

5.1.1. Artifact explanation

The artifacts captured in the table below are derived from the authors previous experience responding to incidents. These artifacts are not meant to capture all relevant forensic data points, but are provided as a sample for the types of analysis that should be conducted. Organizational specific artifacts can easily be added into the long tail analysis example implementation scripts.

Artifact Category	Artifact	Command	Artifact Rationale and Considerations
Ports	List listening ports	netstat -ln	Listening network sockets that are outside the normal parameters for the asset function (e.g. HTTP listening on a DNS server) may indicate a configuration error or the presence of backdoor utilities. Additionally, obscure listening ports (e.g. 4444) may indicate unauthorized or malicious software. netstat displays the listening sockets (-l) based on port number (-n).
Processes	List running processes	ps –Ao "comm,pid,ppid,start,user"	Processes can be used to find malicious programs or administration activities. Process anomalies can be discovered by looking for processes that utilize tricky process names (e.g. kthread vs. kthreadd), processes started in the wrong user context (e.g. init started by any user except root), uncommon processes (e.g. rktdaemon), and process with the wrong parent process ID (e.g. init started by the sendmail process). ps displays all processes (-A) with a format (-o) of command, process ID, parent process ID, time process was started, and user ("comm,pid,ppid,start,user").

©2016 SANS Institute, Author retains full rights.

Services	List services and state (on/off)	chkconfig --list	Services are used to run scripts and applications in the background, and can be utilized by attackers for persistence. Run level 3 (full multiuser with networking) and run level 5 (full multiuser with networking and X windows) are the most commonly used run levels. chkconfig lists the services and the service status (on/off) for each run level (--list). Reference the start and stop scripts for additional information on the functionality of each service (e.g. head –n 15 /etc/rc3.d/S24nfslock).
Kernel modules	List loaded kernel modules	lsmod	Kernel modules provide the Linux kernel with a modular design and enable the kernel to be dynamically or manually modified at runtime by loading and unloading modules (Red Hat, n.d.). Attackers can leverage kernel modules to perform low level rootkit functionality or obtain low level access (e.g. access to all memory). The lsmod command lists the loaded kernel modules and what each kernel module is used by.
Files	List setuid and setgid files	find / -perm -2000 –o –perm -4000	setuid and setgid files are permissions granted to executables that allow executable owners or groups to temporarily run privileged functions as unprivileged users. Attackers commonly upload setuid or setgid files to escalate privileges or modify existing binaries for stealthy persistence. The find command is used to locate files in the root directory (/) with the setuid (-perm -4000) or (-o) setgid (-perm -2000).
	Open but unlinked files	lsof +L1 (SANS Institute, n.d.)	Each file within an operating system should normally have one or more links. In the past, attackers have started a program then immediately deleted the binary from disk to make the recovery of the original binary more difficult. However, unlinked binaries can be recovered from the /proc virtual file system. lsof lists open files with a link count of zero (+L1).

©2016 SANS Institute, Author retains full rights.

	Regular files in the /dev directory	find /dev –type f	Character and block files are typically found in the /dev directory (Pomeranz, 2015). A limited number of regular files (-type f) normally exist in the /dev directory. Regular files in the /dev directory should be fairly consistent between systems. The find command is used to locate regular files (-type f) in the dev directory (/dev).
	Files that start with dots or spaces	find / \(-name " *" –o –name ".*" \) (SANS Institute, n.d.)	Attackers commonly utilize file names that start with a space or a dot for obfuscation. Files that start with a space may allow a file to co-exist and blend in with a similar named file. Similarly, files that start with a dot are hidden from the default ls command (utilize the –a flag to display all entries). The find command locates all files in the rood directory (/) with a name that begins with a space (-name " *") or (-o) a name that begins with a dot (-name ".*").
	Directories that start with dots	find / -type d –name .*	Attackers commonly utilize directories that start with a dot for obfuscation. Directories that start with a dot are hidden from the default ls command (utilize the –a flag to display all entries). The find command locates all files in the root directory (/) that are directories (-type d) and begin with a dot (.*).
	List and hash of files in the bin directories	sha256sum /bin/* /sbin/* /usr/bin/*	The /bin, /sbin, and /usr/bin directories contain core binaries and libraries. These files should be relatively consistent across systems (e.g. files are in these directories and the file contents). In the past, attackers have trojanized commonly used binaries (e.g. ls) or added additional binaries to these directories. The sha256sum command creates a SHA 256 hash of each file in the /bin/*, /sbin*, and /usr/bin/* directories. This can be utilized to find unauthorized or trojanized binaries.

©2016 SANS Institute, Author retains full rights.

	List of immutable files	lsattr –R /	In previous incidents, attackers have set the immutable file permission to prevent a malicious file from being deleted by administrators that are unfamiliar with immutable files. Even with root permissions, files that have the immutable bit set cannot be deleted without removing the immutable bit. Immutable permissions occur infrequently in a normal operating system build. Immutable files should be considered anomalous unless an organization uses this as a common system administration technique. The lsattr command displays a list file attributes, recursively (-R) for the root directory (/). Files that have the immutable permission set are listed as "----i----------".
Tasks	List crontab entries	cat /etc/crontab /etc/cron.d/*	Cron is a method to schedule commands and scripts to run at periodic intervals (Red Hat, n.d.). Attackers commonly use cron for persistence. The cat command is used to read the crontab (/etc/crontab) and all shell scripts located in the cron.d (/etc/cron.d/*) directory. This can be utilized to find unauthorized cron jobs or shell scripts.
	List and hash scripts executed by run-parts	sha256sum /etc/cron.daily/* /etc/cron.hourly/* /etc/cron.monthly/*	Within the crontab, Red Hat includes a "run-parts" cron job that executes any shell script under the /etc/cron.daily, /etc/cron.weekly, and /etc/cron.monthly directories (Red Hat, n.d.). The sha256sum command creates a SHA 256 hash of each file within the /etc/cron.daily/*, /etc/cron.weekly/*, and /etc/cron.monthly/* directories. This can be utilized to find unauthorized cron shell scripts.
	List users allowed to make changes to cron	cat /etc/cron.allow	If a cron.allow file exists, then users who are allowed to run cron jobs must be listed in this file. Users that are not in the cron.allow file are not allowed to run cron jobs. If a cron.allow file does not exist, Linux and Unix operating systems look for a cron.deny file. Any user listed in the cron.deny file is not

© 2017 SANS Institute, Author retains full rights.

			allowed to run cron jobs. Users not listed in the cron.deny file are permitted to run cron jobs. If a cron.allow or cron.deny file does not exist, then all users are allowed to run cron jobs by default (Red Hat, n.d.). The presence of cron.allow, cron.deny or entries within each of these files may indicate anomalous activity. The cat command is used to read the contents of the /etc/cron.allow file if it exists. This can be utilized to find unauthorized cron.allow files or entries.
	List users denied from making changes to cron	cat /etc/cron.deny	If a cron.allow file exists, then users who are allowed to run cron jobs must be listed in this file. Users that are not in the cron.allow file are not allowed to run cron jobs. If a cron.allow file does not exist, Linux and Unix operating systems look for a cron.deny file. Any user listed in the cron.deny file is not allowed to run cron jobs. Users not listed in the cron.deny file are permitted to run cron jobs. If a cron.allow or cron.deny file does not exist, then all users are allowed to run cron jobs by default (Red Hat, n.d.). The presence of cron.allow, cron.deny or entries within each of these files may indicate anomalous activity. The cat command is used to read the contents of the /etc/cron.deny file if it exists. This can be utilized to find unauthorized cron.deny files or entries.
Authentication	List user accounts	cat /etc/passwd	User accounts provide access to an operating system or application. In previous incidents, attackers have created additional user accounts for persistent access. The first column in the /etc/passwd file contains a list of all operating system user accounts. The cat command is used to read the password (/etc/passwd) file to obtain the current list of user accounts. This can be used to find unauthorized user accounts.

©2016 SANS Institute, Author retains full rights.

	List user accounts with a null password	awk -F: '($2 == "") {print $1}' /etc/shadow (Murdoch, 2014)	Blank (null) account passwords should not normally occur within an enterprise managed operating system. The presence of a user account with a null password may indicates anomalous activity and should be investigated. The awk command is used to read the /etc/shadow file (/etc/shadow), by separating the fields by a colon (-F:), and print the username in field one ({print $1}) where the password in field two is blank {$2== “”). A password field with a value of “*” indicates an account is disabled, and a password field with a value of “!!” indicates that a password has not been set and the account cannot be used to login.
	SSH authorized_keys for the root account	cat /root/.ssh/authorized_keys	The /<user account>/.ssh/authorized_keys file lists public keys that are allowed to utilize SSH on an operating system (Red Hat, n.d.). In previous incidents, attackers have added their own public keys to the authorized_keys files for persistent access. The cat command is utilized to list the contents for the authorized SSH public keys for the root account (/root/.ssh/authorized_keys). This can be utilized to find unauthorized public keys that are allowed to authentication to the operating system.
	Hash of sudoers file	sha256sum /etc/sudoers	Sudo provides the ability to delegate the use of privileged commands to unprivileged users without the use of the root account (Red Hat, n.d.). The /etc/sudoers configuration file is used to determine the privileged commands that can be ran by users. In an enterprise environment, this file tends to be consistent across related systems or business units. Attackers could utilize the sudoers file to maintain privileged access with an unprivileged account. Deviations in the sudoers files could indicate malicious activity and should be investigated. The sha256sum command is used to create a SHA 256 hash of the

©2016 SANS Institute, Author retains full rights.

			/etc/sudoers file. This can be utilized to find configuration changes made to the sudoers file.
	List of .rhost files	find / -name .rhosts	The .rhosts and file is used for insecure host/IP based authentication (Pomeranz, 2015). .rhosts files should not be used in an enterprise environment. Attackers may setup rhost authentication to hide in plain sight, or leverage a compromised machine with rhost authentication to move laterally across the network. The find command is used to locate .rhost files (-name .rhosts) in the root directory (/). This can be utilized to find unauthorized .rhosts files.
	List of hosts.equiv files	find /etc -name hosts.equiv (Murdoch, 2014)	The hosts.equiv file allows users and hosts to utilize legacy commands such as rcp, rsh, and rlogin (Kerrisk, 2015). The legacy r* commands do not utilize encryption. The presence of the hosts.equiv file may indicate a configuration error or an attacker attempting to hide in plain sight. The find command is utilized to locate hosts.equiv files (-name hosts.equiv) in the /etc directory (/etc). This can be utilized to find unauthorized hosts.equiv files.
	List of X0.hosts files	find /etc -name X*.hosts (Murdoch, 2014)	The X*.hosts file is used for X server host authentication (Katherine & Joseph, 2001). Host based authentication can by trivially defeated using DNS poisoning. Attackers may setup X*.hosts authentication to hide in plain sight, or leverage a compromised machine with X*.hosts to move laterally across the network. Alternative methods should be used to remotely access X serveries, such as X11 forwarding over SSH. The find command is utilized to locate X*.hosts files (-name X*.hosts) in the /etc directory (/etc). This can be utilized to find unauthorized X*.hosts files.

©2016 SANS Institute, Author retains full rights.

Startup	List the default runlevel	cat /etc/inittab	Red Hat uses the SysV runlevel system to determine which start and kill scripts are ran by init (Red Hat, n.d.). The typical run level for Red Hat systems is run level three or run level five. Systems that use alternate run levels may be caused by a configuration error, system maintenance, or an attacker trying to hide in plain site by running services from a run that is not monitored by system administrators. The cat command is used to read the inittab file (/etc/inittab) to determine the current run level (e.g. id:5:initdefault:). This can be used to identify the default run level of the operating system.
	List and hash of run level 3&5 startup and kill scripts	sha256sum /etc/rc3.d/* /etc/rc5.d/*	Red Hat uses the SysV runlevel system to determine which start and kill scripts are ran by init (Red Hat, n.d.). The init runlevel scripts reside in the /etc/rc*.d/* directories. Init scripts that start with a "K" are used to kill a service, and scripts that begin with a "S" are used to start a service. Scripts that do not begin with a "K" or an "S" are ignored. The sha256sum command creates a SHA 256 hash of each script in the /etc/rc3.d/* and /etc/rc5.d/* directories. This can be used to identify new scripts that an attacker has added or made changes to an existing script.
DNS	List the local DNS host file entries	cat /etc/hosts (Zeltser, 2015)	The hosts file is used by the operating system for local DNS resolution. This file is checked prior to using DNS servers for name resolution (Red Hat, n.d.). An attacker may attempt to modify this file to perform man-in-the-middle or denial of service attacks. The cat command is used to read the hosts file (/etc/hosts). This can be used to identify unauthorized name resolution entries.
	List DNS servers used	cat /etc/resolv.conf	The resolv.conf file lists the current DNS servers. An attacker may attempt to configure alternate DNS servers to perform man-in-the-middle or denial of service attacks. The cat command is

© 2016 SANS Institute, Author retains full rights.

			used to read the contents of the resolv.conf file (/etc/resolv.conf). This can be used to identify the use of unauthorized DNS servers.
Packages	Verify RPM packages	rpm –Va (SANS Institute, n.d.)	The rpm package manager can be used to verify an installed package with information about the package from the rpm database. Specifically, rpm can be used to verify package file attributes (size, hash, permission, type, owner, group), package dependencies, and run a verification script (rpm.org, n.d.). The rpm command can be ran with the verification option (-V) for all packages (-a). This can be utilized to find package files that have changed since they were built. However, this may output a large number of changes that may not be malicious. Each change should be verified on a case by case basis.

©2016 SANS Institute, Author retains full rights.

5.1.2. Artifact extraction script

The artifact extraction script automates the extraction of the defined long tail analysis artifacts listed in section 5.1.1. This script should be copied over to the remote endpoint that is being analyzed and ran locally. Additional organizational specific application or operating system specific artifacts can easily be added to this script.

```
#!/bin/bash
# Copyright Joshua Lewis, 2016.
# Version 1.0
# Permission to use and distribute this script permitted as long as the
copyright is preserved and referenced
# Use of this script at your own risk, no warrany is implied

#Define path to trusted binaries
BINPATH="/root/analysis/tools"
#Define path for output
OUTPATH="/root/analysis/results"
#Set date and time the analysis was ran
DATETIME=$(date +%m-%d-%Y.%H.%M)

#Make directory results
mkdir $OUTPATH

################### Ports ###################
#$BINPATH/echo "### Capture listenging ports "
$BINPATH/netstat -ln | $BINPATH/egrep -v '^unix|Active|Proto' |
$BINPATH/sed -e 's@^@'"$($BINPATH/hostname)"'\ '@' | $BINPATH/awk '{
print $5,$2,$1}' | $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_netstat.csv

################### Process ###################
#$BINPATH/echo "### Capture running processes "
$BINPATH/ps -Ao "comm,pid,ppid,start,user" | $BINPATH/grep -v ^COMMAND
|$BINPATH/sed -e 's@^@'"$($BINPATH/hostname)"'\ '@' | $BINPATH/awk '{
print $2,$3,$4,$5,$6,$1}' | $BINPATH/tr -s " " "," >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_processListing.csv

################### Services ###################
#$BINPATH/echo "### Capture run level 3 & 5 services "
$BINPATH/chkconfig --list | $BINPATH/awk '{print $1,$5,$7}'|
$BINPATH/sed -e 's@$@'\ "$(hostname)"'@'| $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_runLevel3And5Services.csv

################### Kernel modules ###################
#$BINPATH/echo "### Capture kernel modules "
$BINPATH/lsmod | $BINPATH/awk '{print $1,$2}'| $BINPATH/tr -s ' ' ','|
$BINPATH/sed -e 's@$@',"$($BINPATH/hostname)"'@' >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_kernelModules.csv

################### Files ###################
#$BINPATH/echo "### Capture setuid and setgid files "
```

© 2015 The SANS Institute Author retains full rights.

```
$BINPATH/find / -perm -2000 -o -perm -4000 -exec $BINPATH/sh -c 'echo
"$0"' {} \; 2>/dev/null | $BINPATH/sed -e 's@$@'\
"$($BINPATH/hostname)"'@'| $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_setUidAndSetGidFiles.csv

#$BINPATH/echo "### Capture open but unlinked files "
# The idea to collect this artifact derived from the SANS Intrusion
Discovery Cheat Sheet for Linux (https://pen-
testing.sans.org/resources/downloads)
$BINPATH/lsof +L1 | $BINPATH/awk '{print $1,$2,$3,$10}'|$BINPATH/tr -s
' ' ','| $BINPATH/sed -e 's@$@',"$($BINPATH/hostname)"'@' |
$BINPATH/grep -v 'COMMAND,PID,USER,NAME,'>
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_open_unlinked_files.csv

#$BINPATH/echo "### Capture regular files in the /dev directory "
# The idea to collect this artifact derived from SANS 506 Securing
Linux/Unix by Hal Pomeranz
$BINPATH/find /dev -type f | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' >$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_regularFilesInDev.csv

#$BINPATH/echo "### Capture files that start with a dots or spaces "
# The idea to collect this artifact derived from the SANS Intrusion
Discovery Cheat Sheet for Linux (https://pen-
testing.sans.org/resources/downloads)
$BINPATH/find / \( -name " *" -o -name ".*" \) -exec sh -c 'echo "$0"'
{} \; | $BINPATH/sed -e 's@$@',"$($BINPATH/hostname)"'@'
>$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_dotAndSpaceFiles.csv

#$BINPATH/echo "### Capture directories that start with a dots "
$BINPATH/find / -type d -name .\* | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' >$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_dotDirectories.csv

#$BINPATH/echo "### Capture list and hash of files in the bin
directories "
$BINPATH/sha256sum /bin/* /sbin/* /usr/bin/* 2>/dev/null | $BINPATH/sed
-e 's@$@',"$($BINPATH/hostname)"'@' | $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_hashOfFilesInBinDirectories.csv

#$BINPATH/echo "### Capture list of immutable files "
$BINPATH/lsattr -R / 2>/dev/null | $BINPATH/grep -e '----i-' |
$BINPATH/sed -e 's@$@',"$($BINPATH/hostname)"'@' | $BINPATH/tr -s ' '
',' > $OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_chattrFiles.csv

################### Tasks ###################
#$BINPATH/echo "### Capture crontab "
$BINPATH/cat /etc/crontab /etc/cron.d/* | $BINPATH/grep -Ev
'^SHELL=/bin/bash|^PATH=/sbin|^MAILTO=root|^HOME=/|^#' | $BINPATH/sed -
s '/^$/d'| $BINPATH/sed -e 's/$/,/'| $BINPATH/sed -e
's@$@'"$($BINPATH/hostname)"'@' >$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_crontabAndCron.d.csv

#$BINPATH/echo "### Capture shell scripts executed by run-parts daily,
hourly, monthly "
```

Joshua Lewis, Joshua.D.Lewis@gmail.com

© 2015 The SANS Institute Author retains full rights.

```
$BINPATH/sha256sum /etc/cron.daily/* /etc/cron.hourly/*
/etc/cron.monthly/* | $BINPATH/sed -e 's@$@'\
"$($BINPATH/hostname)"'@'| $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_run-
partsDailyHourlyMonthly.csv

#$BINPATH/echo "### Capture cron allow settings "
$BINPATH/cat /etc/cron.allow 2>/dev/null | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' | $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_cronAllow.csv

#$BINPATH/echo "### Capture cron deny settings "
$BINPATH/cat /etc/cron.deny 2>/dev/null | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' | $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_cronDeny.csv

################### Authentication ###################
#$BINPATH/echo "### Capture user accounts "
$BINPATH/cat /etc/passwd | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@'>$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_userAccts.csv

# The idea to collect this artifact derived from the "Blue Team
Handbook: Incident Response Edition" by Don Murdoch
#$BINPATH/echo "### Capture user accounts with a null password "
$BINPATH/awk -F: '($2 == "") {print $1}' /etc/shadow | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' | $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_accountsWithNullPasswd.csv

#$BINPATH/echo "### Capture SSH authorized_keys for the root account "
$BINPATH/cat /root/.ssh/authorized_keys | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' >$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_rootSshAuthKeys.csv

#$BINPATH/echo "### Capture hash of sudoers file "
$BINPATH/sha256sum /etc/sudoers | $BINPATH/sed -e 's@$@'\
"$($BINPATH/hostname)"'@'| $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_sudoersFile.csv

#$BINPATH/echo "### Capture the list of .rhost files "
$BINPATH/find / -name .rhosts | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' > $OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_rhostFiles.csv

# The idea to collect this artifact derived from the "Blue Team
Handbook: Incident Response Edition" by Don Murdoch
#$BINPATH/echo "### Capture the list of hosts.equiv files "
$BINPATH/find /etc -name hosts.equiv | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' > $OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_hosts.equivFiles.csv

# The idea to collect this artifact derived from the "Blue Team
Handbook: Incident Response Edition" by Don Murdoch
#$BINPATH/echo "### Capture the list of X0.hosts files "
```


© 2015 The SANS Institute Author retains full rights.

```
$BINPATH/find /etc -name X0.hosts | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' > $OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_X0.hostsFiles.csv

#TODO add UID 0 accounts in /etc/passwd
#TODO add list of groups
#TODO add Pam configuration
#TODO add /etc/ssh/sshd_config

################### Startup ###################
#$BINPATH/echo "### Capture default runlevel "
$BINPATH/cat /etc/inittab | $BINPATH/grep -Ev '^#' | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' >$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_inittab.csv

#$BINPATH/echo "### Capture the list and hash of run level 3 & 5
startup and kill scripts "
$BINPATH/sha256sum /etc/rc3.d/* /etc/rc5.d/* | $BINPATH/sed -e 's@$@'\
"$($BINPATH/hostname)"'@'| $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_runlevel3And5StartKillScripts.csv

##TODO add /etc/rc.local

################### DNS ###################
# The idea to collect this artifact derived from the Security Incident
Survey Cheat Sheet for Server Administrators
(https://zeltser.com/security-incident-survey-cheat-sheet/)
#$BINPATH/echo "### Capture host file entries "
$BINPATH/cat /etc/hosts | $BINPATH/grep -ve $($BINPATH/hostname) |
$BINPATH/grep -v '::1' | $BINPATH/awk '{print $1,$2}' | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' | $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_etcHosts.csv

# The idea to collect this artifact derived from the Security Incident
Survey Cheat Sheet for Server Administrators
(https://zeltser.com/security-incident-survey-cheat-sheet/)
#$BINPATH/echo "### Capture DNS servers used "
$BINPATH/cat /etc/resolv.conf | $BINPATH/grep -Ev '^#' | $BINPATH/sed -
e 's@$@',"$($BINPATH/hostname)"'@' | $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo $DATETIME)_resolv.confEntries.csv

################### Packages ###################
# The idea to collect this artifact derived from the SANS Intrusion
Discovery Cheat Sheet for Linux (https://pen-
testing.sans.org/resources/downloads)
#$BINPATH/echo "### Capture RPM package verification "
$BINPATH/rpm -Va 2>/dev/null | $BINPATH/sort | $BINPATH/grep -v
'prelink' | $BINPATH/awk '{ print $1,$3}' | $BINPATH/sed -e
's@$@',"$($BINPATH/hostname)"'@' | $BINPATH/tr -s ' ' ',' >
$OUTPATH/$($BINPATH/hostname)_$(echo
$DATETIME)_rpmPackageVerification.csv
```


© 2015 The SANS Institute Author retains full rights.

5.1.3. Setup public and private key authentication

Public and private keys provide a strong authentication mechanism to authenticate to a remote endpoint. Building and using the keys can be setup on the analysis server using the following steps:

- **Step 1**: Generate public and private keys
  - Run ssh-keygen –t rsa –b 4096
  - Ensure a bit length of 4096+ is utilized

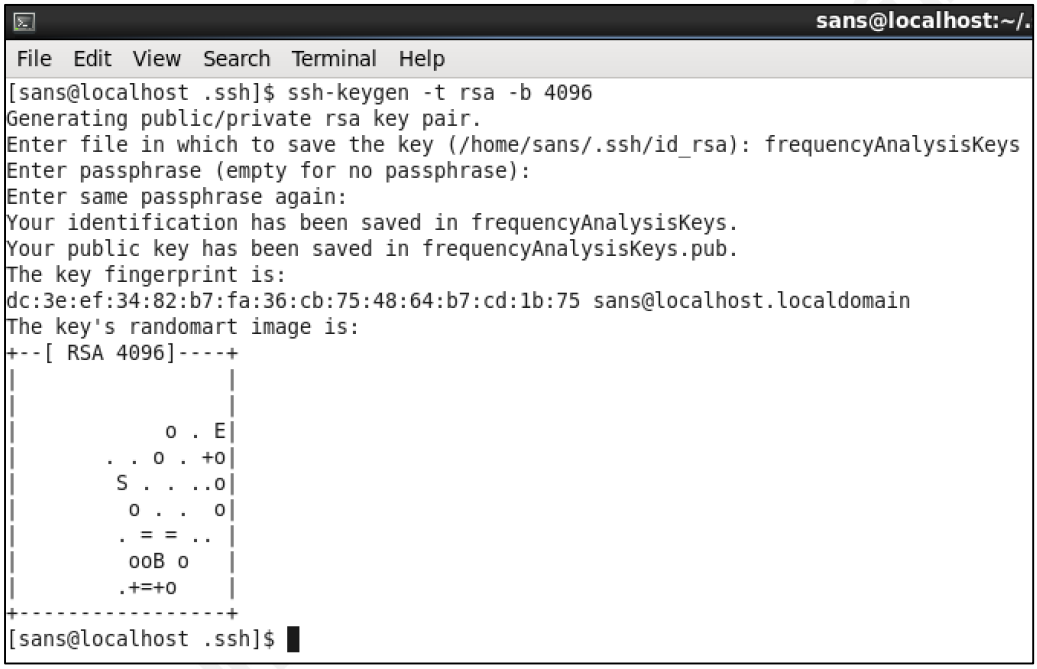

Figure 2 - Generate public and private keys

- **Step 2**: Restrict how the public key can be utilized (Pomeranz, 2015)
  - no-pty: prevents an interactive shell
  - no-port-forwarding: prevents TCP port forwarding
  - no-X11-forwarding: prevents forwarding of the X11 protocol

© 2015 The SANS Institute Author retains full rights.

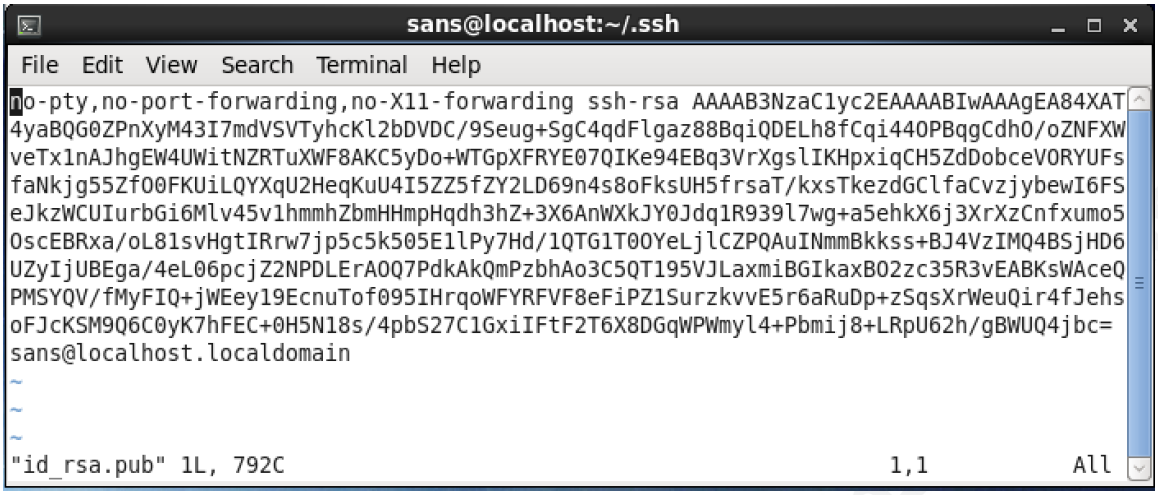

Figure 3 - Set restrictions on how public keys can be utilized

- **Step 3**: Copy the public key to the authorized_keys file for each remote host to be evaluated with the long tail analysis framework
  - scp /home/sans/.ssh/frequencyAnalysisKeys.pub root@webserver:/root/.ssh
  - ssh root@webserver "cat /root/.ssh/frequencyAnalysisKeys.pub >> /root/.ssh/authorized_keys"

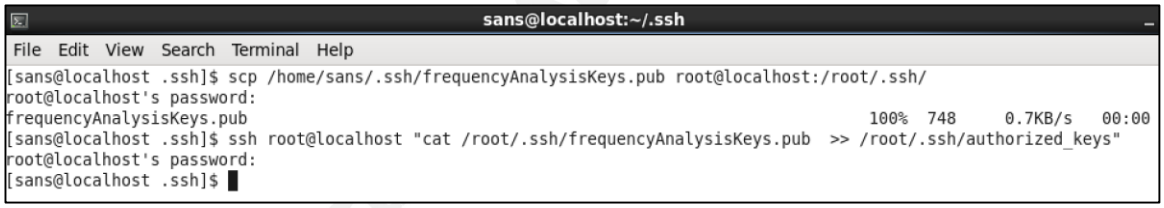

Figure 4 - Copy public key to remote hosts

- **Step 4**: Make a copy of the private key and rename as id_rsa
  - cp frequencyAnalysisKeys id_rsa

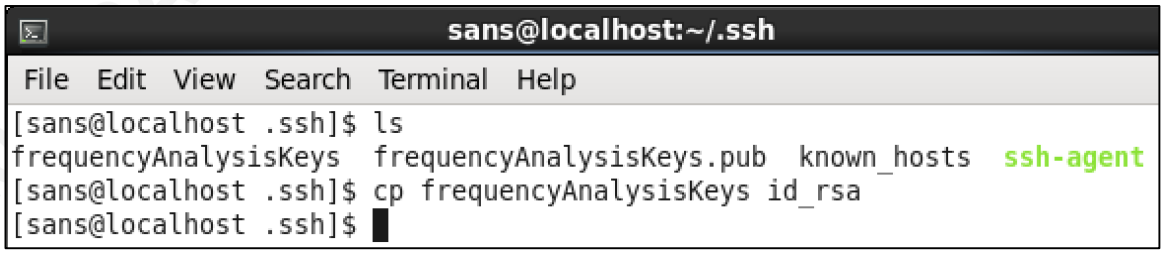

Figure 5 - Configure the private key

- **Step 5**: Setup ssh-agent
  - ssh and scp natively communicate with ssh-agent through the SSH_AUTH_SOCK variable (Pomeranz, 2015)

© 2015 The SANS Institute Author retains full rights.

- Utilize the script below to run ssh-agent to output the shellcode for the SSH_AUTH_SOCK and SSH_AGENT_PID environment variables to a file, then run ssh-add to load the encrypted private keys in memory
- Run this script each time the analysis server is restarted

```
#!/bin/bash
#This script re-sets the ssh-agent environment variables and loads the
encrypted private keys into memory.
#RUN THIS SCRIPT AFTER EACH RESTART

#Script derived from SANS 506 Securing Linux/Unix by Hal Pomeranz

#remove previous versions of the ssh-agent file
rm -f $HOME/.ssh/ssh-agent

#run the ssh-agent command to get shellcode the SSH_AUTH_SOCK and
SSH_AGENT_PID, then output this to a file
ssh-agent | grep -v echo > $HOME/.ssh/ssh-agent

#set the permissions on the ssh-agent file
chmod 700 $HOME/.ssh/ssh-agent

#source this file so that the environment variables remain available
. $HOME/.ssh/ssh-agent

#force ssh-agent to load the private keys in memory; will be prompted
to enter passwords for encrypted private keys
ssh-add
```

- **Step 6**: Run updateSSH.sh script to update the ssh-agent file created in step 5

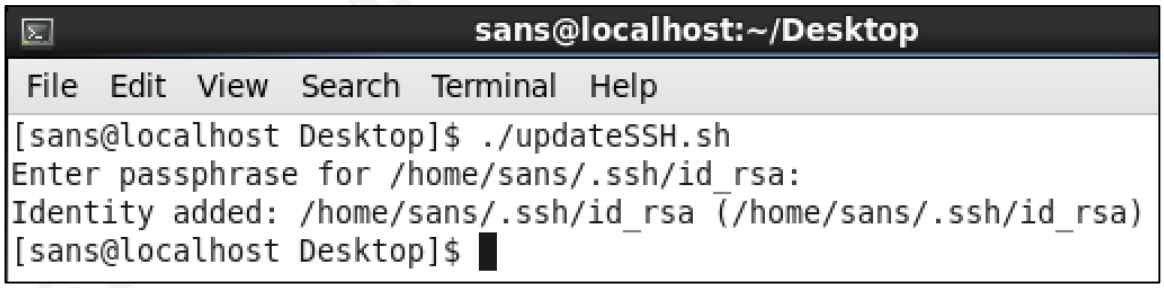

Figure 6 - Update ssh-agent file to be sourced in scripts

- **Step 7**: Source ssh-agent file in analysis server scripts that require ssh-agent authentication

© 2015 The SANS Institute Author retains full rights.

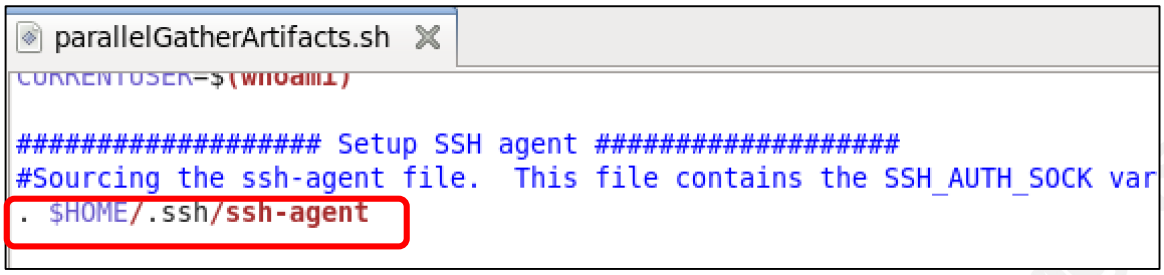

Figure 7 - Source ssh-agent file in scripts

5.1.4. Gather artifacts script

The gather artifacts script automates the parallel execution of the remote artifact extraction script and the parallel collection of the artifacts extracted from the remote endpoints. Parallel SSH and SCP are achieved using the parallel-ssh (Chun & McNabb, 2012) program. Download parallel-ssh from https://code.google.com/p/parallel-ssh/, and place the unzipped folder in the current working directory defined in the script variables. Additionally, ensure that the trusted binaries and artifact extraction script is placed in an analysis.tgz tarball in the current working directory (reference section 5.1.6 for additional details). Finally, ensure that a host.txt file exists in the current working directory that lists each remote endpoint that should be analyzed (format the file: root@<insert host name>, one entry per line).

```
#!/bin/bash
# Copyright Joshua Lewis, January 2016.
# Version 1.0
# Permission to use and distribute this script permitted as long as the
copyright is preserved and referenced
# Use of this script at your own risk, no warrany is implied

################### Setup variables ###################
#Define path to local binaries
LOCALPSCP=/home/sans/Desktop/pssh-2.3.1/bin/pscp
LOCALPSSH=/home/sans/Desktop/pssh-2.3.1/bin/pssh
LOCALSSH=/usr/bin/ssh
LOCALMKDIR=/bin/mkdir
LOCALTAR=/bin/tar
LOCALRM=/bin/rm
LOCALECHO=/bin/echo
LOCALLS=/bin/ls
LOCALMV=/bin/mv
LOCALGREP=/bin/grep
LOCALAWK=/bin/awk
LOCALSORT=/bin/sort
LOCALUNIQ=/usr/bin/uniq
```


© 2015 The SANS Institute Author retains full rights.

```
LOCALHEAD=/usr/bin/head
LOCALTEE=/usr/bin/tee
LOCALIFCONFIG=/sbin/ifconfig
LOCALCUT=/bin/cut
LOCALSSHKEYSCAN=/usr/bin/ssh-keyscan
LOCALCHOWN=/bin/chown

#Define working directories and file variables
LOCALWORKINGDIR=/home/sans/Desktop
SCRIPANDTOOLSTAR=analysis.tgz
TARGETS=hosts.txt
FREQUENCYANALYSISKNOWNHOST=frequencyAnalysisKnownHost
FREQUENCYANALYSISRUNLOG=frequencyAnalysisRunLog.txt
ARTIFACTCOLLECTIONSCRIPT=artifactExtraction.sh
LOCALRESULTSDIR=/home/sans/Desktop/results
LOCALHISTORICALDIR=/home/sans/Desktop/results/historical
LOCALCURRENTDIR=/home/sans/Desktop/results/current
REMOTEROOTDIR=/root
REMOTEWORKINGDIR=/root/analysis

#Error handling to make sure the hosts.txt file exists
if [ ! -f "$LOCALWORKINGDIR/$TARGETS" ]; then
    $LOCALECHO "## <error>"
    $LOCALECHO "host.txt file not present in $LOCALWORKINGDIR.  Create
this file and add \"roo@host\" entries, one per line"
    $LOCALECHO "## </error>"
    exit 255
fi

#Define network interface used to connect to remote machines
LOCALNETIF=eth3

#Define other runtime variables
DATETIME=$(date +%m-%d-%Y.%H.%M)
REMOTERESULTSTAR="${DATETIME}_results.tgz"
CURRENTUSER=$(whoami)

################### Setup SSH agent ###################
#Sourcing the ssh-agent file.  This file contains the SSH_AUTH_SOCK
variable to allow ssh-agent to pass the keys that are loaded in memory
. $HOME/.ssh/ssh-agent

################### Clean up local results directory to prep for
historical analysis ###################
# Check for previous host evaluation results, move any previous results
to <local working directory>/results/historical/<host>/<last run time
of analysis>
$LOCALMKDIR $LOCALRESULTSDIR 2>/dev/null
$LOCALMKDIR $LOCALCURRENTDIR 2>/dev/null
cd $LOCALCURRENTDIR
if [[ $($LOCALLS | $LOCALGREP .csv) ]] ; then
    while $LOCALLS *.csv &>/dev/null; do
        HOSTLASTRUN=$($LOCALLS | $LOCALGREP .csv | $LOCALAWK -F "_"
'{print $1}' | $LOCALHEAD -1)
        DATELASTRUN=$($LOCALLS | $LOCALGREP .csv | $LOCALAWK -F "_"
'{print $2}' | $LOCALHEAD -1)
        $LOCALMKDIR $LOCALHISTORICALDIR 2>/dev/null
```


© 2015 The SANS Institute Author retains full rights.

```
        $LOCALMKDIR $LOCALHISTORICALDIR/$($LOCALECHO $HOSTLASTRUN)
2>/dev/null
        $LOCALMKDIR $LOCALHISTORICALDIR/$($LOCALECHO
$HOSTLASTRUN)/$DATELASTRUN 2>/dev/null
        $LOCALMV $($LOCALECHO $HOSTLASTRUN)*.csv
$LOCALHISTORICALDIR/$HOSTLASTRUN/$DATELASTRUN
    done
fi

################### Copy tar artifact collection and tools to remote
host ###################
cd $LOCALWORKINGDIR
#Copy tar of artifact collection script and tools
#Using parallel scp (pscp) -h (lists of hosts to run scp in parallel) -
p (amount of parallel hosts at the same time); output results to
console and to a log file
$LOCALPSCP -h $LOCALWORKINGDIR/$TARGETS -p 25
$LOCALWORKINGDIR/$SCRIPANDTOOLSTAR $REMOTEROOTDIR/$SCRIPANDTOOLSTAR

#Copy statically or dynamically linked, trusted tar, cp, and rm
binaries to handle the extraction of the artifact collection script
#Using parallel scp (pscp) -h (lists of hosts to run scp in parallel) -
p (amount of parallel hosts at the same time); output results to
console and to a log file
$LOCALPSCP -h $LOCALWORKINGDIR/$TARGETS -p 25 $LOCALWORKINGDIR/tar
$REMOTEROOTDIR/tar
$LOCALPSCP -h $LOCALWORKINGDIR/$TARGETS -p 25 $LOCALWORKINGDIR/rm
$REMOTEROOTDIR/rm

################### Run artifact collection script on remote host
###################
#Untar artifact collection script and tools, run the script, tar the
results
#Using parallel ssh (pssh) -h (lists of hosts to run scp in parallel) -
p (amount of parallel hosts at the same time); output results to
console and to a log file
$LOCALPSSH -h $LOCALWORKINGDIR/$TARGETS -p 25 -t 600 "(cd
$REMOTEROOTDIR; $REMOTEROOTDIR/tar -xzpf $SCRIPANDTOOLSTAR; cd
$REMOTEWORKINGDIR; ./$ARTIFACTCOLLECTIONSCRIPT 2>/dev/null; cd
$REMOTEWORKINGDIR/results; $REMOTEROOTDIR/tar -czf
\$(hostname)_$REMOTERESULTSTAR *)"

################### Copy the artifact collection results back to the
local host and cleanup ###################
#Copy remote host results back to the local host
$LOCALMKDIR $LOCALHISTORICALDIR 2>/dev/null

#pssh-2.3.1 does not support the copy of files from a remote machine to
a local machine, this section is a temp workaround
#Get the IP address of the interface used to connect to the remote
machines, build a entry to be added to the known_hosts file on remote
machines
$LOCALSSHKEYSCAN $($LOCALIFCONFIG $LOCALNETIF | $LOCALGREP 'inet addr:'
| $LOCALCUT -d: -f2 | $LOCALAWK '{ print $1}') >
$LOCALWORKINGDIR/$FREQUENCYANALYSISKNOWNHOST
#copy the entry to be added to the known_hosts file to the remote
machines
```


© 2015 The SANS Institute Author retains full rights.

```
$LOCALPSCP -h $LOCALWORKINGDIR/$TARGETS -p 25
$LOCALWORKINGDIR/$FREQUENCYANALYSISKNOWNHOST $REMOTEROOTDIR/.ssh
#add the entry to the known_hosts file on the remote machines
$LOCALPSSH -h $LOCALWORKINGDIR/$TARGETS -p 25 "cat
$REMOTEROOTDIR/.ssh/$FREQUENCYANALYSISKNOWNHOST >>
$REMOTEROOTDIR/.ssh/known_hosts"
#ssh into the remote host using ssh-agent forwarding, then scp the
results back to the analysis server
$LOCALPSSH -h $LOCALWORKINGDIR/$TARGETS -X -A -p 25 "scp
$REMOTEWORKINGDIR/results/*_$REMOTERESULTSTAR root@$($LOCALIFCONFIG
$LOCALNETIF | $LOCALGREP 'inet addr:' | $LOCALCUT -d: -f2 | $LOCALAWK
'{ print $1}'):$LOCALCURRENTDIR"
#reset file permissions for the /results/current/* files
$LOCALSSH root@127.0.0.1 "$LOCALCHOWN $CURRENTUSER
/$LOCALCURRENTDIR/*.tgz"

#Cleanup files on the remote host
#Note that deleting and re-creating these files/directories overwrites
unallocated space. The tradeoff of leaving these
# files in place is that an attacker may be able to see what artifacts
are being collected and/or modify the local scripts.
$LOCALPSSH -h $LOCALWORKINGDIR/$TARGETS -p 25 "($REMOTEROOTDIR/rm -rf
$REMOTEWORKINGDIR; $REMOTEROOTDIR/rm -f $SCRIPANDTOOLSTAR;
$REMOTEROOTDIR/rm -f tar; $REMOTEROOTDIR/rm -f rm)" | $LOCALTEE -a
$LOCALWORKINGDIR/$FREQUENCYANALYSISRUNLOG

#Untar local results and delete tar
cd $LOCALCURRENTDIR
for i in *.tgz; do $LOCALTAR -xzf $i &>/dev/null; done
$LOCALRM -f $LOCALCURRENTDIR/*.tgz
```

5.1.5. Long tail analysis script

The long tail analysis script automates the parsing and rate of occurrence calculation for the collected artifacts from remote endpoints. This script is ran on the analysis server after all of the artifact gathering scripts have completed.

```
#!/bin/bash
# Copyright Joshua Lewis, 2016.
# Version 1.0
# Permission to use and distribute this script permitted as long as the
copyright is preserved and referenced
# Use of this script at your own risk, no warrany is implied

################### Setup variables ###################
#Define path to local binaries
LOCALECHO=/bin/echo
LOCALCLEAR=/usr/bin/clear
LOCALAWK=/bin/awk
LOCALCAT=/bin/cat
LOCALMKDIR=/bin/mkdir
LOCALSORT=/bin/sort
LOCALUNIQ=/usr/bin/uniq

#Define working directories
```


© 2015 The SANS Institute Author retains full rights.

```
LOCALCURRENTRESULTSDIR=/home/sans/Desktop/results/current
LOCALFREQUENCYDIR=/home/sans/Desktop/results/frequency_analysis

#Reusable commands
##TODO Clean up the case statements by substituting some of the common
sequences of multiple commands with a variable
#################### Determine artifact to perform frequency analysis
####################
$LOCALCLEAR
$LOCALECHO "Select a number 1-27 to perform artifact frequency
analysis"
$LOCALECHO "[1]  Frequency of listening ports"
$LOCALECHO "[2]  Frequency of running processes"
$LOCALECHO "[3]  Frequency of run level 3 & 5 services"
$LOCALECHO "[4]  Frequency of kernel modules"
$LOCALECHO "[5]  Frequency of setuid and setgid files"
$LOCALECHO "[6]  Frequency of open but unlinked files"
$LOCALECHO "[7]  Frequency of regular files in the /dev directory"
$LOCALECHO "[8]  Frequency of files that start with a dots or spaces"
$LOCALECHO "[9]  Frequency of directories that start with a dots"
$LOCALECHO "[10] Frequency of list and hash of files in the bin
directories"
$LOCALECHO "[11] Frequency of list of immutable files"
$LOCALECHO "[12] Frequency of cron entries"
$LOCALECHO "[13] Frequency of run-parts daily, hourly, monthly entries"
$LOCALECHO "[14] Frequency of cron allow settings"
$LOCALECHO "[15] Frequency of cron deny settings"
$LOCALECHO "[16] Frequency of user accounts"
$LOCALECHO "[17] Frequency of user accounts with a null password"
$LOCALECHO "[18] Frequency of SSH authorized_keys for the root account"
$LOCALECHO "[19] Frequency of hash of sudoers file"
$LOCALECHO "[20] Frequency of the list of .rhost files"
$LOCALECHO "[21] Frequency of the list of hosts.equiv files"
$LOCALECHO "[22] Frequency of the list of X0.hosts files"
$LOCALECHO "[23] Frequency of default runlevels"
$LOCALECHO "[24] Frequency of the list and hash of run level 3 & 5
startup and kill scripts"
$LOCALECHO "[25] Frequency of host file entries"
$LOCALECHO "[26] Frequency of DNS servers used"
$LOCALECHO "[27] Frequency of RPM package verification"
$LOCALECHO -n "Enter number, the press [ENTER]: "
read ARTIFACTTOEVALNUM

##TODO Add parsing for additional artifacts
# 1) UID 0 accounts
# 2) List of groups
# 3) Pam configuration
# 4) SSHd_config
# 4) /etc/rc.local

#################### Analyze artifacts based on user input
####################
$LOCALMKDIR $LOCALCURRENTRESULTSDIR 2>/dev/null
cd $LOCALCURRENTRESULTSDIR
$LOCALMKDIR $LOCALFREQUENCYDIR 2>/dev/null
case $ARTIFACTTOEVALNUM in
1)
```

Joshua Lewis, Joshua.D.Lewis@gmail.com

© 2015 The SANS Institute Author retains full rights.

```
    #Find the frequency for listening ports
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*netstat*.csv >
$LOCALFREQUENCYDIR/currentNetstatCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currentNetstatCombined.csv | $LOCALSORT | $LOCALUNIQ
-c | $LOCALSORT -n > $LOCALFREQUENCYDIR/netstatFrequency
    less $LOCALFREQUENCYDIR/netstatFrequency
    ;;
2)
    ##TODO Capture the PPID in this as well so that process name, user
and PPID can help identify start anomalies for processes trying to hide
in plane site
    #Find the frequency for running processes
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*processListing*.csv >
$LOCALFREQUENCYDIR/currentProcessListingCombined.csv
    $LOCALAWK -F ',' '{print $1,$5}'
$LOCALFREQUENCYDIR/currentProcessListingCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/processListingFrequency
    less $LOCALFREQUENCYDIR/processListingFrequency
    ;;
3)
    #Find the frequency for run level 3 & 5 services
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*runLevel3And5Services*.csv >
$LOCALFREQUENCYDIR/currentRunLevel3And5ServicesCombined.csv
    $LOCALAWK -F ',' '{print $1,$2,$3}'
$LOCALFREQUENCYDIR/currentRunLevel3And5ServicesCombined.csv |
$LOCALSORT | $LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/runLevel3And5ServicesFrequency
    less $LOCALFREQUENCYDIR/runLevel3And5ServicesFrequency
    ;;
4)
    #Find the frequency for kernel modules
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*kernelModules*.csv >
$LOCALFREQUENCYDIR/currentKernelModulesCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currentKernelModulesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/kernelModulesFrequency
    less $LOCALFREQUENCYDIR/kernelModulesFrequency
    ;;
5)
    #Find the frequency for setuid and setgid files
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*setUidAndSetGidFiles*.csv >
$LOCALFREQUENCYDIR/currentsetUidAndSetGidFilesCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentsetUidAndSetGidFilesCombined.csv | $LOCALSORT
| $LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/setUidAndSetGidFilesFrequency
    less $LOCALFREQUENCYDIR/setUidAndSetGidFilesFrequency
    ;;
6)
```


© 2015 The SANS Institute Author retains full rights.

```
    #Find the frequency for open but unlinked files
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*open_unlinked_file*.csv >
$LOCALFREQUENCYDIR/currentOpenUnlinkedFilesCombined.csv
    $LOCALAWK -F ',' '{print $1,$4}'
$LOCALFREQUENCYDIR/currentOpenUnlinkedFilesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/openUnlinkedFilesFrequency
    less $LOCALFREQUENCYDIR/openUnlinkedFilesFrequency
    ;;
7)
    #Find the frequency for regular files in the /dev directory
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*regularFilesInDev*.csv >
$LOCALFREQUENCYDIR/currentregularFilesInDevCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentregularFilesInDevCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/regularFilesInDevFrequency
    less $LOCALFREQUENCYDIR/regularFilesInDevFrequency
    ;;
8)
    #Find the frequency for files that start with a dots or spaces
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*dotAndSpaceFiles*.csv >
$LOCALFREQUENCYDIR/currentdotAndSpaceFilesCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentdotAndSpaceFilesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/dotAndSpaceFilesFrequency
    less $LOCALFREQUENCYDIR/dotAndSpaceFilesFrequency
    ;;
9)
    #Find the frequency for directories that start with a dots
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*dotDirectories*.csv >
$LOCALFREQUENCYDIR/currentdotDirectoriesCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentdotDirectoriesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/dotDirectoriesFrequency
    less $LOCALFREQUENCYDIR/dotDirectoriesFrequency
    ;;
10)
    #Find the frequency for list and hash of files in the bin
directories
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*hashOfFilesInBinDirectories*.csv
> $LOCALFREQUENCYDIR/currenthashOfFilesInBinDirectoriesCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currenthashOfFilesInBinDirectoriesCombined.csv |
$LOCALSORT | $LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/hashOfFilesInBinDirectoriesFrequency
    less $LOCALFREQUENCYDIR/hashOfFilesInBinDirectoriesFrequency
    ;;
11)
    #Find the frequency for list of immutable files
```


© 2015 The SANS Institute Author retains full rights.

```
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*chattrFiles*.csv >
$LOCALFREQUENCYDIR/currentchattrFilesCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currentchattrFilesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n > $LOCALFREQUENCYDIR/chattrFilesFrequency
    less $LOCALFREQUENCYDIR/chattrFilesFrequency
    ;;
12)
    #Find the frequency for cron entries
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*crontabAndCron*.csv >
$LOCALFREQUENCYDIR/currentcrontabAndCronCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentcrontabAndCronCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/crontabAndCronFrequency
    less $LOCALFREQUENCYDIR/crontabAndCronFrequency
    ;;
13)
    #Find the frequency for run-parts daily, hourly, monthly entries
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*run-partsDailyHourlyMonthly*.csv
> $LOCALFREQUENCYDIR/currentrun-partsDailyHourlyMonthlyCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}' $LOCALFREQUENCYDIR/currentrun-
partsDailyHourlyMonthlyCombined.csv | $LOCALSORT | $LOCALUNIQ -c |
$LOCALSORT -n > $LOCALFREQUENCYDIR/run-partsDailyHourlyMonthlyFrequency
    less $LOCALFREQUENCYDIR/run-partsDailyHourlyMonthlyFrequency
    ;;
14)
    #Find the frequency for cron allow settings
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*cronAllow*.csv >
$LOCALFREQUENCYDIR/currentcronAllowCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentcronAllowCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n > $LOCALFREQUENCYDIR/cronAllowFrequency
    less $LOCALFREQUENCYDIR/cronAllowFrequency
    ;;
15)
    #Find the frequency for cron deny settings
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*cronDeny*.csv >
$LOCALFREQUENCYDIR/currentcronDenyCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentcronDenyCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n > $LOCALFREQUENCYDIR/cronDenyFrequency
    less $LOCALFREQUENCYDIR/cronDenyFrequency
    ;;
16)
    #Find the frequency for user accounts
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*userAccts*.csv >
$LOCALFREQUENCYDIR/currentuserAcctsCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentuserAcctsCombined.csv | $LOCALSORT |
```


© 2015 The SANS Institute Author retains full rights.

```
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/cronuserAcctsFrequency
    less $LOCALFREQUENCYDIR/cronuserAcctsFrequency
    ;;
17)
    #Find the frequency for user accounts with a null password
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*accountsWithNullPasswd*.csv >
$LOCALFREQUENCYDIR/currentaccountsWithNullPasswdCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentaccountsWithNullPasswdCombined.csv |
$LOCALSORT | $LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/accountsWithNullPasswdFrequency
    less $LOCALFREQUENCYDIR/accountsWithNullPasswdFrequency
    ;;
18)
    #Find the frequency for SSH authorized_keys for the root account
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*rootSshAuthKeys*.csv >
$LOCALFREQUENCYDIR/currentrootSshAuthKeysCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentrootSshAuthKeysCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/rootSshAuthKeysFrequency
    less $LOCALFREQUENCYDIR/rootSshAuthKeysFrequency
    ;;
19)
    #Find the frequency for hash of sudoers file
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*sudoersFile*.csv >
$LOCALFREQUENCYDIR/currentsudoersFileCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currentsudoersFileCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n > $LOCALFREQUENCYDIR/sudoersFileFrequency
    less $LOCALFREQUENCYDIR/sudoersFileFrequency
    ;;
20)
    #Find the frequency for list of .rhost files
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*rhostFiles*.csv >
$LOCALFREQUENCYDIR/currentrhostFilesCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentrhostFilesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n > $LOCALFREQUENCYDIR/rhostFilesFrequency
    less $LOCALFREQUENCYDIR/rhostFilesFrequency
    ;;
21)
    #Find the frequency for the list of hosts.equiv files
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*hosts.equivFiles*.csv >
$LOCALFREQUENCYDIR/currenthosts.equivFilesCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currenthosts.equivFilesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/hosts.equivFilesFrequency
    less $LOCALFREQUENCYDIR/hosts.equivFilesFrequency
    ;;
```


© 2015 The SANS Institute Author retains full rights.

```
22)
    #Find the frequency for the list of X0.hosts files
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*X0.hostsFiles*.csv >
$LOCALFREQUENCYDIR/currentX0.hostsFilesCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentX0.hostsFilesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/X0.hostsFilesFrequency
    less $LOCALFREQUENCYDIR/X0.hostsFilesFrequency
    ;;
23)
    #Find the frequency for default runlevels
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*inittab*.csv >
$LOCALFREQUENCYDIR/currentinittabCombined.csv
    $LOCALAWK -F ',' '{print $1}'
$LOCALFREQUENCYDIR/currentinittabCombined.csv | $LOCALSORT | $LOCALUNIQ
-c | $LOCALSORT -n > $LOCALFREQUENCYDIR/inittabFrequency
    less $LOCALFREQUENCYDIR/inittabFrequency
    ;;
24)
    #Find the frequency for the list and hash of run level 3 & 5
startup and kill scripts
    $LOCALCLEAR
    $LOCALCAT
$LOCALCURRENTRESULTSDIR/*runlevel3And5StartKillScripts*.csv >
$LOCALFREQUENCYDIR/currentrunlevel3And5StartKillScriptsCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currentrunlevel3And5StartKillScriptsCombined.csv |
$LOCALSORT | $LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/runlevel3And5StartKillScriptsFrequency
    less $LOCALFREQUENCYDIR/runlevel3And5StartKillScriptsFrequency
    ;;
25)
    #Find the frequency for the host file entries
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*etcHosts*.csv >
$LOCALFREQUENCYDIR/currentetcHostsCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currentetcHostsCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n > $LOCALFREQUENCYDIR/etcHostsFrequency
    less $LOCALFREQUENCYDIR/etcHostsFrequency
    ;;
26)
    #Find the frequency for the DNS servers used
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*resolv.confEntries*.csv >
$LOCALFREQUENCYDIR/currentresolv.confEntriesCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currentresolv.confEntriesCombined.csv | $LOCALSORT |
$LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/resolv.confEntriesFrequency
    less $LOCALFREQUENCYDIR/resolv.confEntriesFrequency
    ;;
27)
    #Find the frequency for RPM package verification
```


© 2015 The SANS Institute Author retains full rights.

```
    $LOCALCLEAR
    $LOCALCAT $LOCALCURRENTRESULTSDIR/*rpmPackageVerification*.csv >
$LOCALFREQUENCYDIR/currentrpmPackageVerificationCombined.csv
    $LOCALAWK -F ',' '{print $1,$2}'
$LOCALFREQUENCYDIR/currentrpmPackageVerificationCombined.csv |
$LOCALSORT | $LOCALUNIQ -c | $LOCALSORT -n >
$LOCALFREQUENCYDIR/rpmPackageVerificationFrequency
    less $LOCALFREQUENCYDIR/rpmPackageVerificationFrequency
    ;;
*)
    echo "Value entered outside of options 1-27.  Try again"
    ;;

esac
```

5.1.6. Putting it all together

- **Step 1**: Create a tar file called "analysis" that contains the trusted binaries and artifact extraction script that will be copied over to the remote endpoint
    - mkdir -p /home/sans/Desktop/analysis /home/sans/Desktop/analysis/tools
    - cd /home/sans/desktop
    - copy statically linked or trusted binaries and scripts into the analysis folder
        - /bin/netstat
        - /bin/egrep
        - /bin/sed
        - /bin/ps
        - /sbin/chkconfig
        - /bin/hostname
        - /bin/find
        - /bin/cat
        - /usr/bin/sha256sum
        - /sbin/lsmod
        - /usr/sbin/lsof
        - /bin/rpm
        - /usr/bin/lsattr
        - /bin/awk
        - /usr/bin/tr
        - /bin/grep
        - /bin/echo
        - /bin/sh

© 2015 The SANS Institute Author retains full rights.

- /bin/sort
- /bin/tar
- /bin/rm

  - tar –czf analysis.tgz analysis

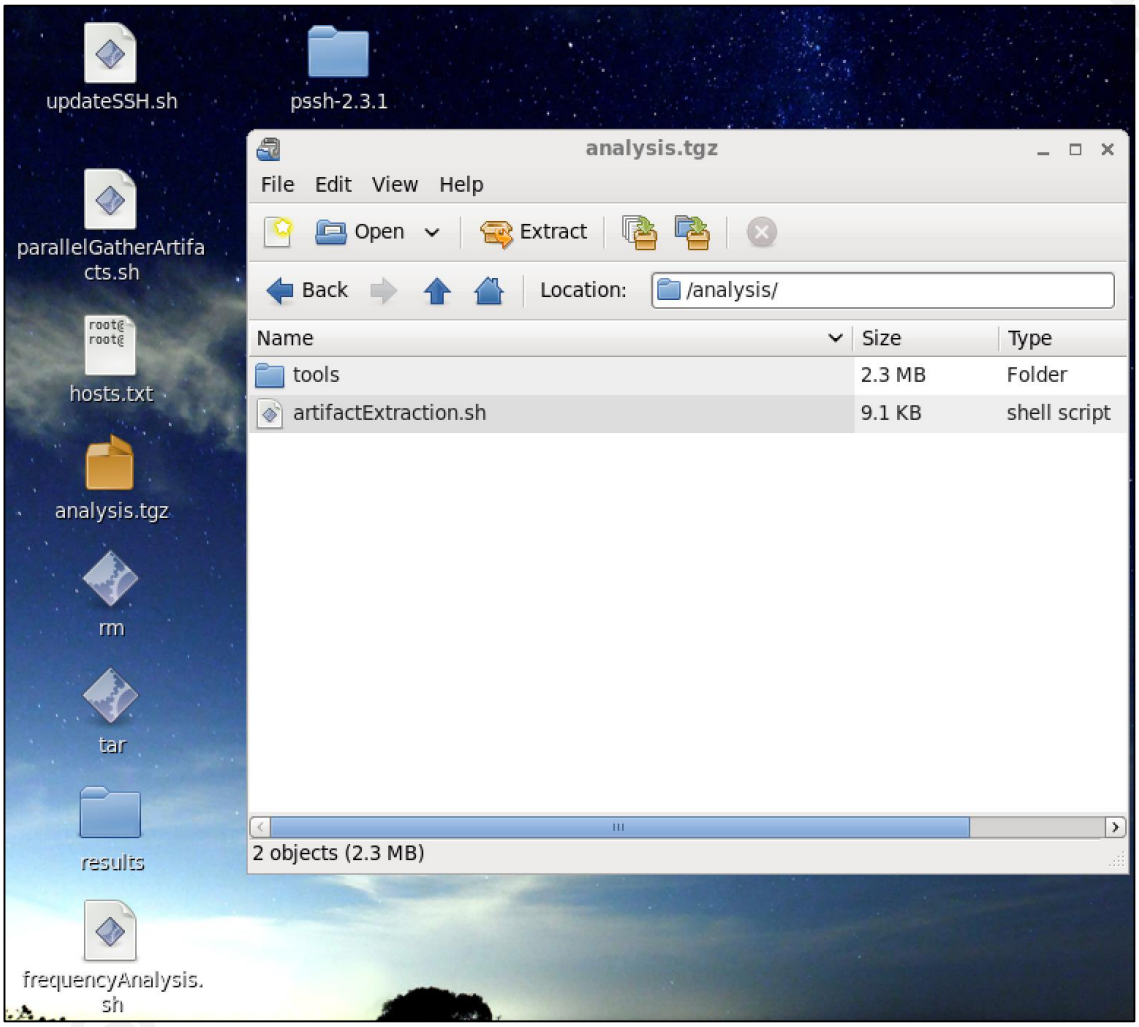

Figure 8 – Long tail analysis framework environment setup

- **Step 2**: Run the updateSSH script

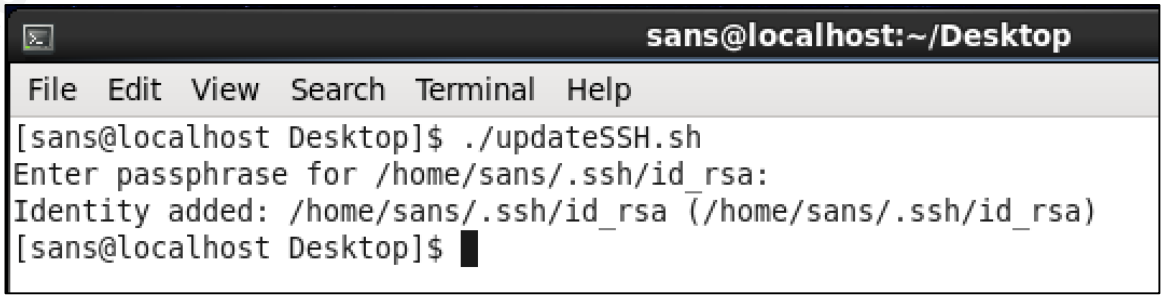

Figure 9 - Update ssh-agent file

© 2015 The SANS Institute Author retains full rights.

- **Step 3**: Test ssh-agent with a remote endpoint
  - Source the ssh-agent file in the current terminal window
    - . /home/sans/.ssh/ssh-agent
  - ssh root@webserver1 uptime

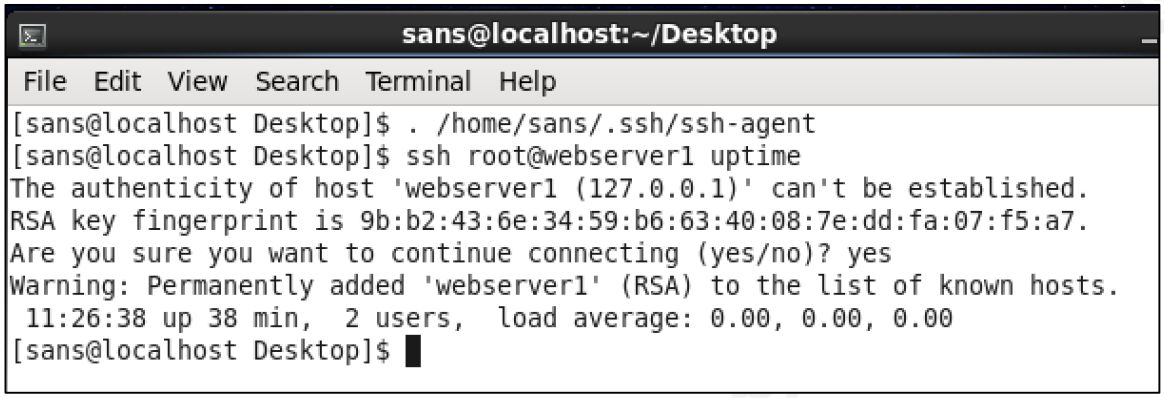

Figure 10 - Test ssh-agent authentication

- **Step 4**: Run the gather artifacts script

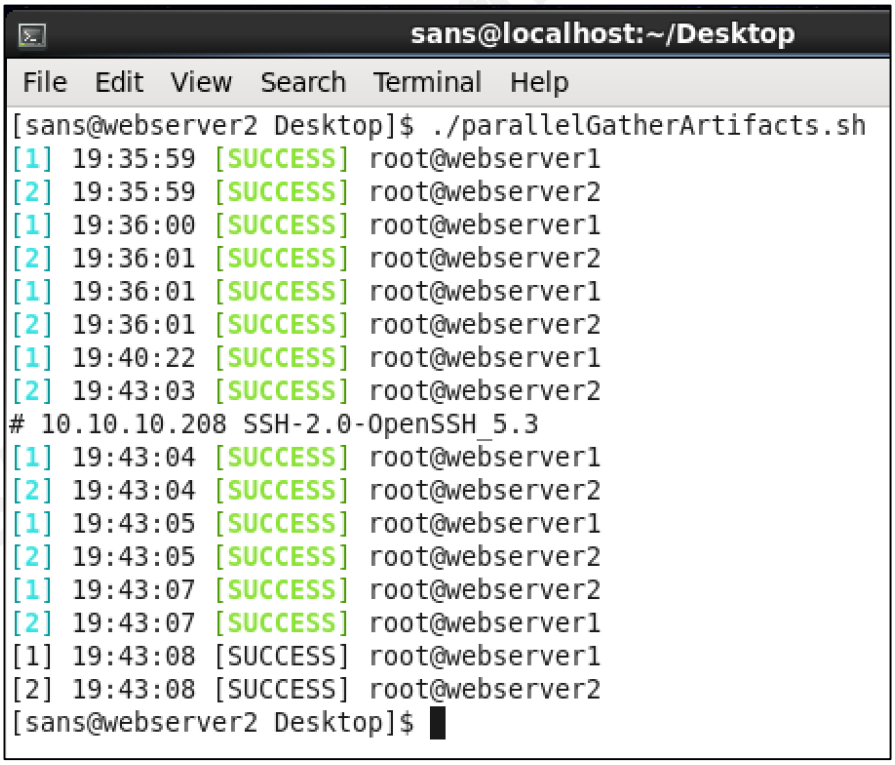

Figure 11 - Run the gather artifacts script

- **Step 5:** Review the updated directory structure

© 2015 The SANS Institute Author retains full rights.

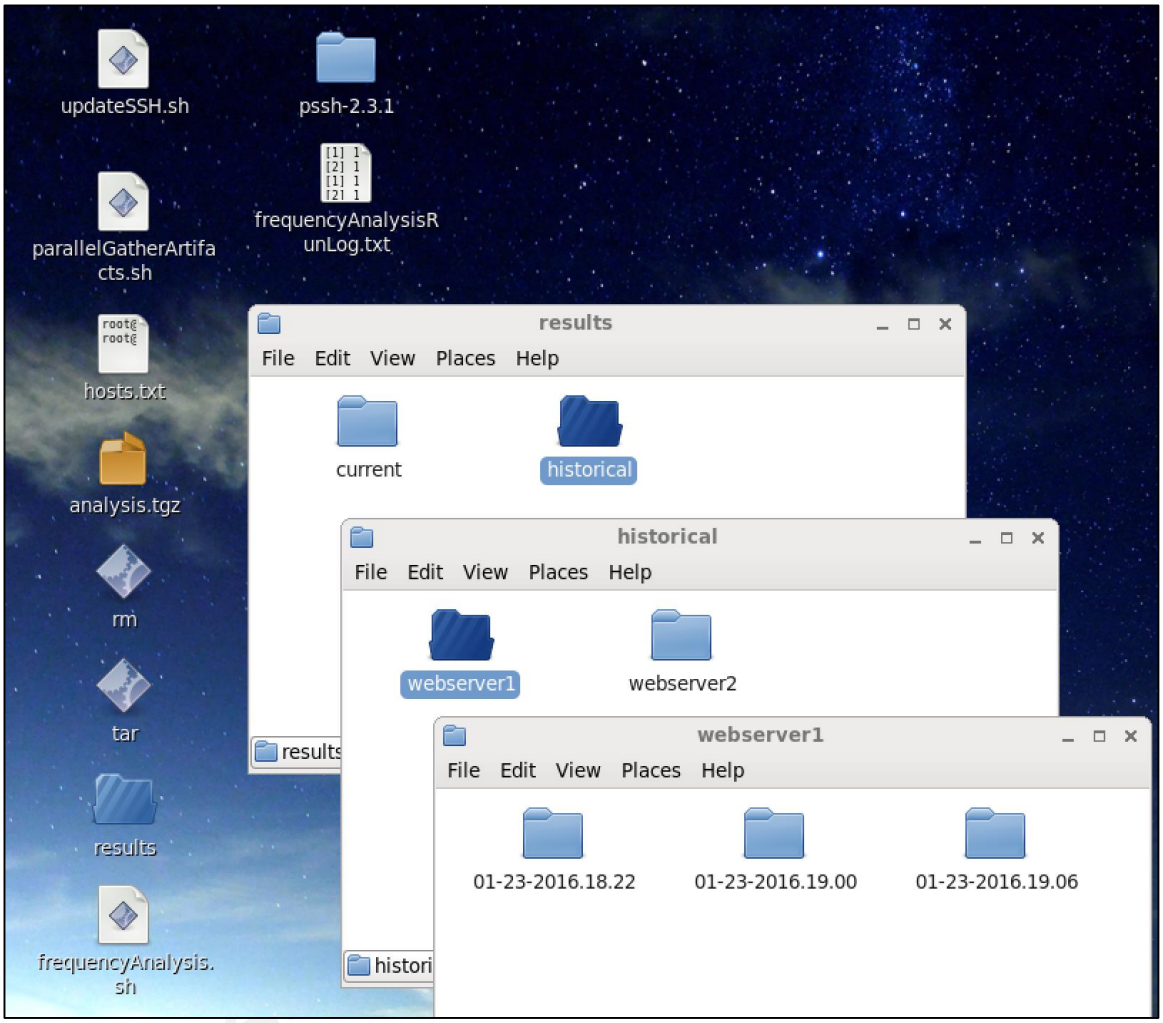

Figure 12: Updated directory structure

- **Step 6**: Run the long tail analysis script

©2016 SANS Institute

© 2015 The SANS Institute Author retains full rights.

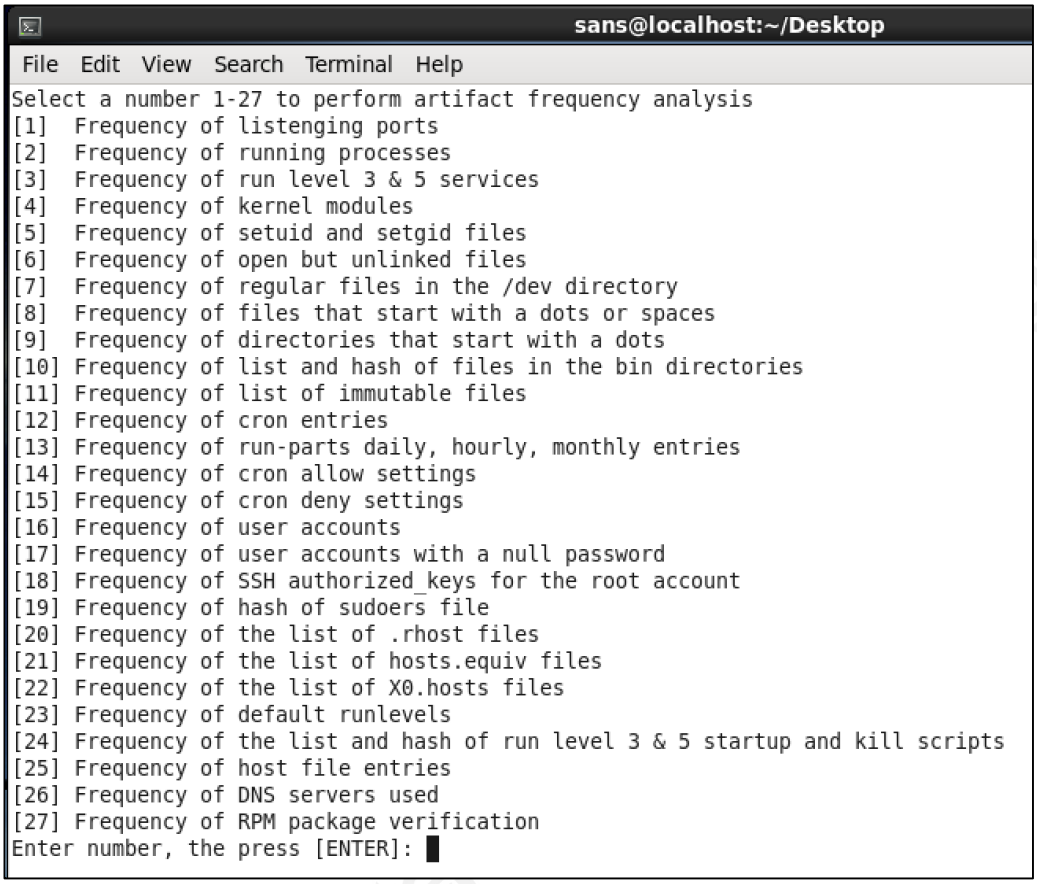

Figure 13 - Running the long tail analysis script

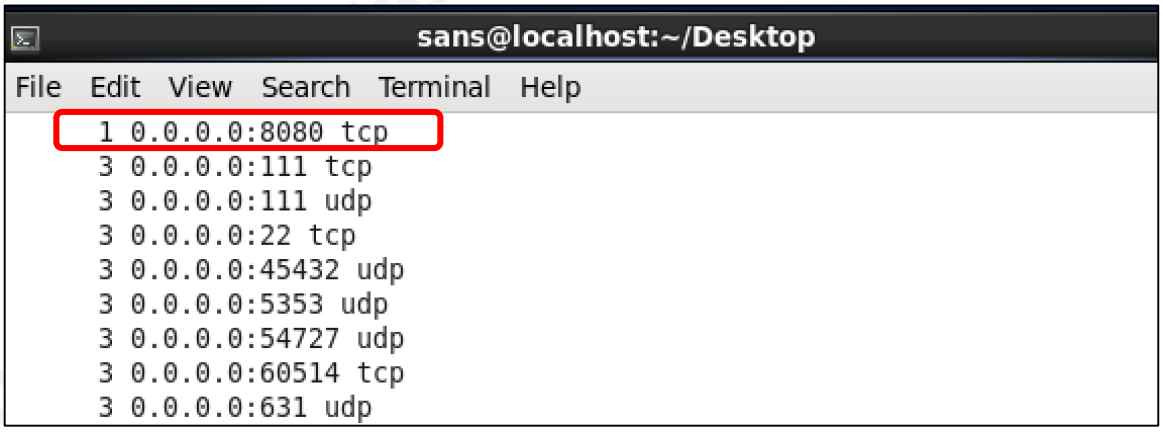

Figure 14 - Identifying outliers for investigation

- **Step 7**: Tracing an outlier back to a host

© 2015 The SANS Institute Author retains full rights.

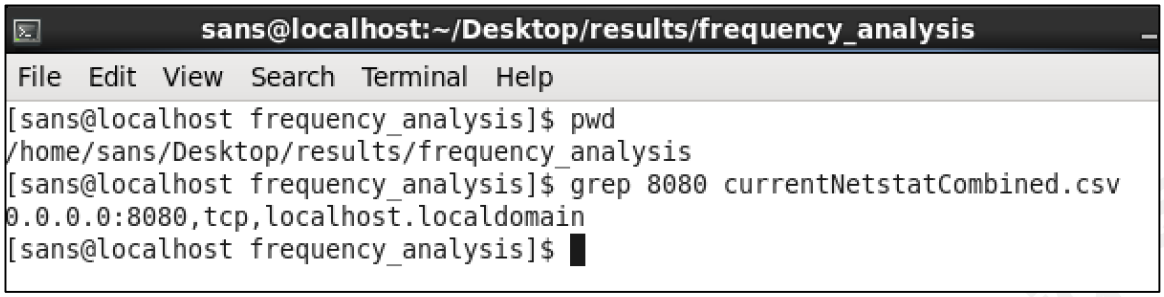

Figure 15 - Tracing an outlier back to a host

©2016 SANS Institute, Author retains full rights.

© 2015 The SANS Institute Author retains full rights.