Linux IR - Key forensic artifacts for incident responders

Taz Wake | Halkyn Consulting Ltd | Originally published 2024-04-24

Overview

Like Thanos, incident response is an inevitability in modern IT environments. Despite some old ideas about various operating systems being "secure" (whatever that means), the reality is that attackers access Linux systems all the time. This means that anyone trying to deal with an attack - from formal IR professionals to help desk teams just troubleshooting - needs to understand what is a *useful* artifact and what conclusions we can draw from it.

In this article, I will highlight some of the main artifacts we use in Linux IR. This is not a conclusive list of every bit of evidence you can find, largely because the difference across Linux distros would make that a never-ending task. Also, this is because, as an investigator, you need to go where the incident takes you and this means different artifacts might suddenly become useful for you. Ultimately, this article is a starting point and will look at the artifacts we deal with most often, across most distros in most incidents.

Please note, some artifacts are only available on live systems (such as /proc, /dev/shm and other virtual file systems). These will be excluded from this article.

Warning Note for Windows Investigators

If you do DFIR across Windows platforms, you will be very used to the wealth of forensic evidence that gets recorded. We have the luxury of artifacts like Shimcache, AmCache, Prefetch, SRUM, and many more sources of activity. We also have the luxury of generally consistent environments with predictable configurations.

There isn't an equivalent of a lot of this within Linux distros. For example, there is nothing that aligns with the Prefetch as "evidence of execution" and while it is possible to use the package management tools for "evidence of program installation", this does **not** give a complete picture.

When you are doing Linux IR you will need to accept the reality that there is less evidence, the evidence can be harder to find and harder to interpret, and the evidence might be in unexpected places. If you are good with this, then Linux IR is for you!

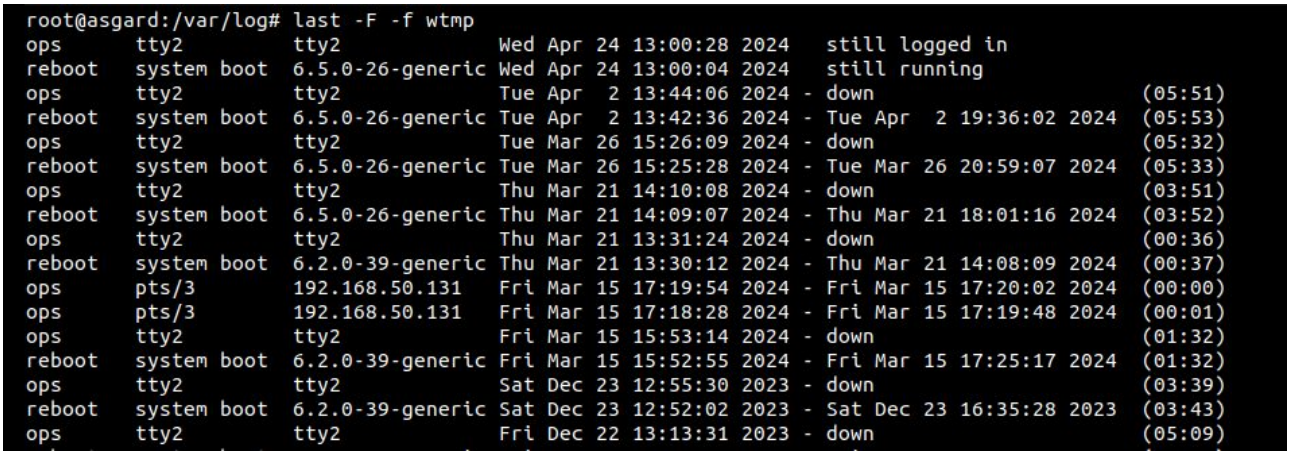
Key Artifacts

Event log data in general

- **Location**: Generally /var/log/*. Some log sources will have their own subfolders (e.g. /var/log/apache2). Some applications will write event logs to other locations - such as Docker using /var/lib/docker/containers/<container_id>.
- **Investigative notes**: Check application logs early in an intrusion, as this is one of the most likely entry points for an attacker. The Operating System logs are often less useful but Authentication and Authorization logs can have value.
- **Possible issues**: Logging collection and retention are driven by the syslog and logrotate daemons. Check their configuration files to see what to expect. Also, remember most logs are plain text and can be manually edited or deleted by any user with root privileges.

Binary authentication event logs

- **Location**: There are three files to consider here - /var/log/wtmp, /var/log/btmp and /var/run/utmp.
- **Investigative notes**. These are binary format files and need to be read with a tool such as last or utmpdump. The wtmp file contains a historical record of all users who have authenticated since the log began. In utmp there is a live record of who is currently logged in, and this is written to wtmp at regular intervals. The btmp file is a record of failed authentication events.
- **Possible issues**: The logon duration and log-off times are not reliable. Some Linux deployments (WSL for example) do not record data in these files as they use the parent OS (Windows for WSL) to record authentication.



```
root@asgard:/var/log# last -F -f wtmp
ops      tty2         tty2             Wed Apr 24 13:00:28 2024   still logged in
reboot   system boot  6.5.0-26-generic Wed Apr 24 13:00:04 2024   still running
ops      tty2         tty2             Tue Apr  2 13:44:06 2024 - down                      (05:51)
reboot   system boot  6.5.0-26-generic Tue Apr  2 13:42:36 2024 - Tue Apr  2 19:36:02 2024  (05:53)
ops      tty2         tty2             Tue Mar 26 15:26:09 2024 - down                      (05:32)
reboot   system boot  6.5.0-26-generic Tue Mar 26 15:25:28 2024 - Tue Mar 26 20:59:07 2024  (05:33)
ops      tty2         tty2             Thu Mar 21 14:10:08 2024 - down                      (03:51)
reboot   system boot  6.5.0-26-generic Thu Mar 21 14:09:07 2024 - Thu Mar 21 18:01:16 2024  (03:52)
ops      tty2         tty2             Thu Mar 21 13:31:24 2024 - down                      (00:36)
reboot   system boot  6.2.0-39-generic Thu Mar 21 13:30:12 2024 - Thu Mar 21 14:08:09 2024  (00:37)
ops      pts/3        192.168.50.131   Fri Mar 15 17:19:54 2024 - Fri Mar 15 17:20:02 2024  (00:00)
ops      pts/3        192.168.50.131   Fri Mar 15 17:18:28 2024 - Fri Mar 15 17:19:48 2024  (00:01)
ops      tty2         tty2             Fri Mar 15 15:53:14 2024 - down                      (01:32)
reboot   system boot  6.2.0-39-generic Fri Mar 15 15:52:55 2024 - Fri Mar 15 17:25:17 2024  (01:32)
ops      tty2         tty2             Sat Dec 23 12:55:30 2023 - down                      (03:39)
reboot   system boot  6.2.0-39-generic Sat Dec 23 12:52:02 2023 - Sat Dec 23 16:35:28 2023  (03:43)
ops      tty2         tty2             Fri Dec 22 13:13:31 2023 - down                      (05:09)
```

Example wtmp data

Authentication and authorization logs

- **Location**: This varies by distro but generally /var/log/auth.log (Debian family) and /var/log/secure (RHEL family).

- **Investigative notes**: These logs contain data around connections, including SSH with data such as remote IP address and a hash of the key used. They also record privilege use, such as sudo and su events.
- **Possible issues**: This log often rotates quickly. If you have a busy system, the data will age out faster than you'd expect. Often only the initial privilege escalation is recorded - for example, if a user runs sudo su -, this will be recorded but once they are root, no additional activity will be logged here.

```
root@asgard:/var/log# zgrep ssh auth.log.3
Mar 15 17:05:25 asgard sshd[5172]: Accepted password for ops from 192.168.50.131 port 49640 ssh2
Mar 15 17:05:25 asgard sshd[5172]: pam_unix(sshd:session): session opened for user ops(uid=1000) by (uid=0)
Mar 15 17:05:26 asgard sshd[5250]: Received disconnect from 192.168.50.131 port 49640:11: disconnected by user
Mar 15 17:05:26 asgard sshd[5250]: Disconnected from user ops 192.168.50.131 port 49640
Mar 15 17:05:26 asgard sshd[5172]: pam_unix(sshd:session): session closed for user ops
Mar 15 17:18:14 asgard sshd[5379]: Connection closed by authenticating user ops 192.168.50.131 port 45670 [preauth]
Mar 15 17:18:28 asgard sshd[5384]: Accepted publickey for ops from 192.168.50.131 port 45504 ssh2: RSA SHA256:mZrJtCX+Zuw11opE7T1k/OcM3Sr7rxSPB
KqnPNrvQ90
Mar 15 17:18:28 asgard sshd[5384]: pam_unix(sshd:session): session opened for user ops(uid=1000) by (uid=0)
Mar 15 17:19:48 asgard sshd[5420]: Received disconnect from 192.168.50.131 port 45504:11: disconnected by user
Mar 15 17:19:48 asgard sshd[5420]: Disconnected from user ops 192.168.50.131 port 45504
Mar 15 17:19:48 asgard sshd[5384]: pam_unix(sshd:session): session closed for user ops
Mar 15 17:19:54 asgard sshd[5440]: Accepted publickey for ops from 192.168.50.131 port 33000 ssh2: RSA SHA256:mZrJtCX+Zuw11opE7T1k/OcM3Sr7rxSPB
KqnPNrvQ90
```

Example of ssh connections in /var/log/auth.log

Audit logs

- **Location**: This defaults to /var/log/audit/audit.log.
- **Investigative notes**: A well-configured audit daemon is an excellent way to get visibility into activity on a Linux system. Unfortunately, auditd is not installed by default on Debian family systems, and while it is enabled on Red Hat family systems, the default configuration is sub-optimal. While it is better than nothing, unless measures have been taken to tailor the audit.conf file, the log will miss significant activity.
- **Possible issues**: While it is a plain text log, it is very difficult to read manually. There are likely to be multiple lines in the log per event and correlation is possible through the message number field. Data is also stored in a range of obfuscated ways - for example, the command line entry on a PROCTITLE is stored as hex-encoded, null-separated data. The easiest ways to read the audit logs are to pass them into a SIEM (ELK, Splunk etc) or to use the aureport and ausearch tools. These work best if you have implemented tagging though.

```
root@asgard:/var/log/audit# head audit.log
type=DAEMON_START msg=audit(1676897810.306:2882): op=start ver=3.0.7 format=enriched kernel=5.15.0-60-generic auid=4294967295 pid=3614 uid=0 se
s=4294967295 subj=unconfined  res=successAUID="unset" UID="root"
type=CONFIG_CHANGE msg=audit(1676897810.374:104): op=set audit_backlog_limit=8192 old=64 auid=4294967295 ses=4294967295 subj=unconfined res=1AU
ID="unset"
type=SYSCALL msg=audit(1676897810.374:104): arch=c000003e syscall=44 success=yes exit=60 a0=3 a1=7ffc4d624b00 a2=3c a3=0 items=0 ppid=3617 pid=
3627 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="auditctl" exe="/usr/sbin/auditctl"
 subj=unconfined key=(null)ARCH=x86_64 SYSCALL=sendto AUID="unset" UID="root" GID="root" EUID="root" SUID="root" FSUID="root" EGID="root" SGID=
"root" FSGID="root"
type=PROCTITLE msg=audit(1676897810.374:104): proctitle=2F7362696E2F617564697463746C002D52002F6574632F61756469742F61756469742E72756C6573
type=CONFIG_CHANGE msg=audit(1676897810.374:105): op=set audit_failure=1 old=1 auid=4294967295 ses=4294967295 subj=unconfined res=1AUID="unset"
type=SYSCALL msg=audit(1676897810.374:105): arch=c000003e syscall=44 success=yes exit=60 a0=3 a1=7ffc4d624b00 a2=3c a3=0 items=0 ppid=3617 pid=
3627 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="auditctl" exe="/usr/sbin/auditctl"
 subj=unconfined key=(null)ARCH=x86_64 SYSCALL=sendto AUID="unset" UID="root" GID="root" EUID="root" SUID="root" FSUID="root" EGID="root" SGID=
"root" FSGID="root"
type=PROCTITLE msg=audit(1676897810.374:105): proctitle=2F7362696E2F617564697463746C002D52002F6574632F61756469742F61756469742E72756C6573
type=CONFIG_CHANGE msg=audit(1676897810.378:106): op=set audit_backlog_wait_time=60000 old=15000 auid=4294967295 ses=4294967295 subj=unconfined
 res=1AUID="unset"
```

Example audit log being read as plain text

```
root@asgard:/var/log/audit# aureport

Summary Report
======================
Range of time in logs: 20/02/23 12:56:50.306 - 24/04/24 13:17:14.032
Selected time for report: 20/02/23 12:56:50 - 24/04/24 13:17:14.032
Number of changes in configuration: 2011
Number of changes to accounts, groups, or roles: 28
Number of logins: 2
Number of failed logins: 1
Number of authentications: 161
Number of failed authentications: 17
Number of users: 4
Number of terminals: 34
Number of host names: 7
Number of executables: 40
Number of commands: 21
Number of files: 0
Number of AVC's: 0
Number of MAC events: 0
Number of failed syscalls: 0
Number of anomaly events: 18
Number of responses to anomaly events: 0
Number of crypto events: 0
Number of integrity events: 0
Number of virt events: 0
Number of keys: 0
Number of process IDs: 1448
Number of events: 20801
```

Example of aureport output

```
root@asgard:/var/log/audit# ausearch -m USER_AUTH
----
time->Mon Feb 20 12:57:58 2023
type=USER_AUTH msg=audit(1676897878.888:130): pid=3791 uid=0 auid=1000 ses=5 subj=unconfined msg='op=PAM:authentication
grantors=pam_rootok acct="root" exe="/usr/bin/su" hostname=? addr=? terminal=/dev/pts/1 res=success'
----
time->Tue Feb 21 00:00:39 2023
type=USER_AUTH msg=audit(1676937639.932:148): pid=1247 uid=0 auid=4294967295 ses=4294967295 subj=unconfined msg='op=PAM:
authentication grantors=pam_permit acct="gdm" exe="/usr/libexec/gdm-session-worker" hostname=asgard addr=? terminal=/dev
/tty1 res=success'
----
time->Tue Feb 21 00:02:07 2023
type=USER_AUTH msg=audit(1676937727.755:207): pid=1870 uid=0 auid=4294967295 ses=4294967295 subj=unconfined msg='op=PAM:
authentication grantors=pam_succeed_if,pam_permit,pam_cap,pam_gnome_keyring acct="ops" exe="/usr/libexec/gdm-session-wor
ker" hostname=asgard addr=? terminal=/dev/tty1 res=success'
----
time->Tue Feb 21 00:02:43 2023
type=USER_AUTH msg=audit(1676937763.112:245): pid=2667 uid=1000 auid=1000 ses=3 subj=unconfined msg='op=PAM:authenticati
on grantors=pam_permit,pam_cap acct="ops" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/0 res=success'
```

Example of ausearch looking for USER_AUTH messages

Task scheduler evidence

- **Location**: There are a few areas to review for this starting with /etc/crontab, which is the system-wide task scheduler. This contains the scheduled tasks built from /etc/cron.d/*. There are also task scheduler entries in /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly, and /etc/cron.monthly, which have a different format to the crontab in that they are more scripts that run when the task scheduler calls them. Then each user account has its own crontab at /var/spool/cron/ (or /var/spool/cron/crontabs).

- **Investigative notes**: Attackers attempt persistence in all of these locations, so every location needs to be checked. Writing to the system crontab requires elevated privileges and only root accounts can make a crontab entry which runs as root.
- **Possible issues**: The crontabs are also plain text files. Attackers will sometimes have the scheduled task delete itself after running. However, be aware that if this happens, the task is no longer scheduled, so it does break persistence.

```
root@asgard:/etc# cat crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
# You can also override PATH, but by default, newer versions inherit it from the environment
#PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

Example of the /etc/crontab entry

User accounts

- **Location**: /etc/passwd
- **Investigative notes**: This records all of the possible accounts on the system, including service accounts and user accounts. Checking things like the MTIME will help identify the last time a user was added or removed. The password file specifies which users can log in and where the user's home folder is stored.
- **Possible issues**: It is a plain text file, readable by every account on the system. There is likely to be a lot of noise, but you can reduce this by focusing on the login shells to identify accounts that can log in interactively.

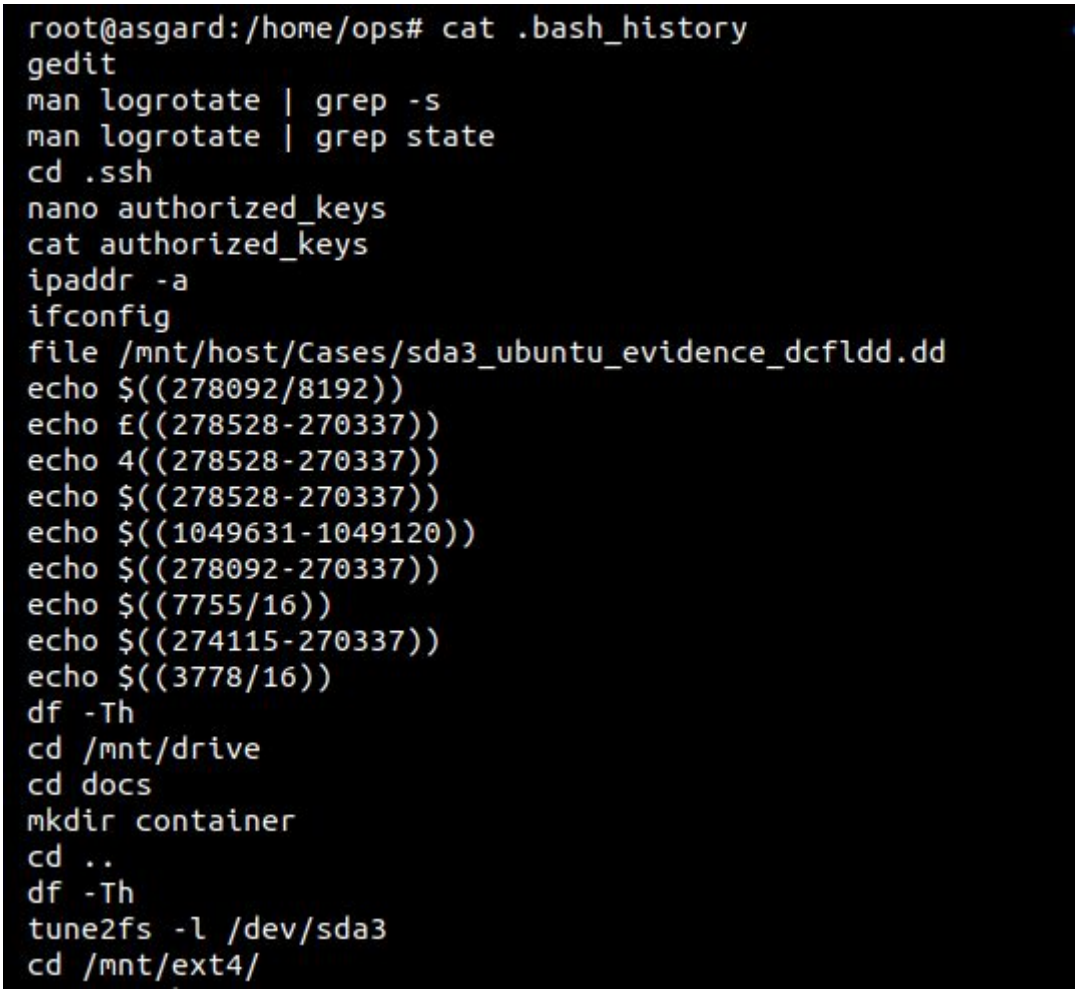
```
root@asgard:/etc# cat passwd | grep sh$
root:x:0:0:root:/root:/bin/bash
ops:x:1000:1000:server,,,:/home/ops:/bin/bash
ftper:x:1001:1001:ftp,ftp,ftp,ftp,ftp:/home/ftper:/bin/bash
samba:x:1002:1002:Samba,,,:/home/samba:/bin/bash
test:x:1003:1004:,,,:/home/test:/bin/bash
```

A passwd file showing users with a login shell.

Command history

- **Location**: /home/username/.bash_history or /home/username/.zsh_history are the most common. Also, check /root/.bash_history and /root/.zsh_history.

- **Investigative notes**: This records commands typed in a bash or zsh shell. The defaults vary by distro but are often the last 500 or so commands. The file does not store timestamps by default and this is generally something that has to be enabled before any intrusions for it to be effective.
- **Possible issues**: Users can avoid storing commands in this history file by entering a space character at the start of their command line. It is also a plain text file that can be edited by the user, so attackers can selectively modify it to remove evidence of their activity. Additionally, there are no inbuilt protections around the history file so it is trivial to wipe or unset it.



Example of a .bash_history file.

SSH Authorized Keys

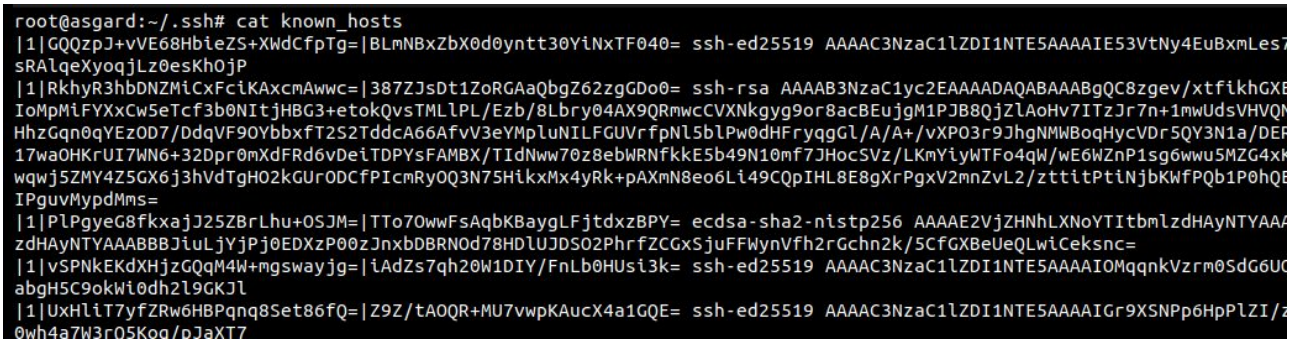
- **Location**: /home/username/.ssh/authorized_keys or /root/.ssh/authorized_keys
- **Investigative notes**: This file contains the public key for any SSH keys which are able to authenticate as the user account. This is often a good place to identify where attackers have inserted their own public keys and are attempting to maintain persistence.

- **Possible issues**: Auditing this data can be time/resource-consuming. An attacker can easily supply a legitimate-looking identity for the key making it hard to spot the odd one out.

Example of Authorized Keys file showing an obvious unusual entry.

SSH Known Hosts file

- **Location**: /home/username/.ssh/known_hosts or /root/.ssh/known_hosts
- **Investigative notes**: The known_hosts file records the details of remote systems that the device has connected to. It stores the IP address and public key of the server and can be used to check lateral movement.
- **Possible issues**: Unfortunately most Linux distros hash the remote IP address for privacy and security reasons. This makes it challenging to confirm the remote IP address. The hashed data is recorded as two base64 encoded strings - the first is the salt, and the second is the hashed IP address. The easiest way to resolve this is to build a list of known-good internal systems, which may allow you to track internal lateral movement. It is unrealistic to maintain a list of external systems unless your environment is heavily controlled.



Example of a Known Hosts file with hashed IP addresses.

Wget HTTPS records

- **Location**: /home/username/.wget-hsts or /root/.wget-hsts
- **Investigative notes**: This file records data when wget is been used to download content from HTTPS connections where an HSTS header has been sent. This allows

you to identify some, but definitely not all, of the remote sites which have been accessed by a user account.

- **Possible issues**: It is a plain-text, user-modifiable file and it does not record every activity.

```
root@asgard:~# cat .wget-hsts
# HSTS 1.0 Known Hosts database for GNU Wget.
# Edit at your own risk.
# <hostname>    <port>  <incl. subdomains>      <created>       <max-age>
repo.zenk-security.com  0       1       1681304546      31536000
laurikari.net   0       1       1683892688      31536000
github.com      0       1       1710521056      31536000
codeload.github.com     0       0       1683892811      31536000
```

Example of .wget-hsts file.

Conclusion

This is a fairly quick journey through some of the key forensic artifacts which contain value for most of the Linux intrusions we have worked. As mentioned previously, there are lots of others that might have value depending on how your environment is configured and what the case you are working on requires.

Please do not view this as a definitive list, instead, it should be seen as a starting point to help guide your investigation. For example, we haven't looked at the Systemd Journal at all - but that is going to become a very significant artifact as distro's mature.

If this is interesting to you, please keep the discussion going - let me know what your favourite artifacts are, and why! Also, if you want to spend a week going over Linux Incident Response in much greater depth while working to analyze a realistic advanced threat actor intrusion, then have a look at FOR577 - https://sans.org/for577.

#linux #forensics #cybersecurity #informationsecurity #dfir #cyber #infosec #cybersec #incidentresponse #ir #SANS #professionaldevelopment

---

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858