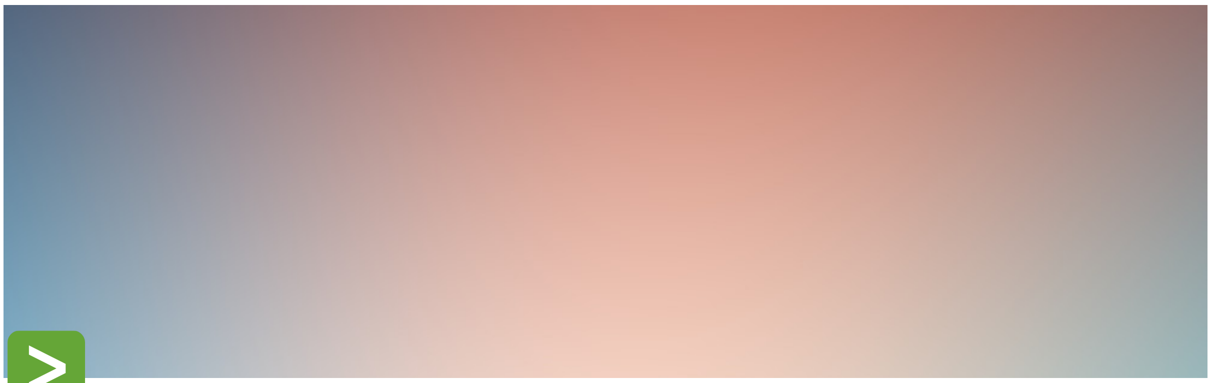


# Linux Security Monitoring with Splunk

## The Current Landscape

### Sysmon for Linux

Add-on for Linux Sysmon | Splunkbase

The Splunk Add-on for Linux Sysmon extract fields from syslog data. Add-On map events for CIM data models: Endpoint, Network Resolution (DNS), Network Traffic, Change. The Splunk Add-on for Linux Sysmon provides the parsing and CIM-compatible knowledge to use with other Splunk apps, such as Splunk Enterprise Security and the Splunk App for PCI Compliance.

https://splunkbase.splunk.com/app/6176/

### Application Focused

Monitoring Linux - Metrics and Logs Forwarding | Splunkbase

Outcold Solutions provide solutions for monitoring Kubernetes, OpenShift, Docker and Linux clusters in Splunk Enterprise and Splunk Cloud. We offer Splunk applications, which give you insights across all environments. We are helping businesses to reduce complexity related to logging and monitoring by

https://splunkbase.splunk.com/app/4768/

## Observability Oriented

Splunk Add-on for Linux | Splunkbase

The Splunk Add-on for Linux allows a Splunk software administrator to collect Linux performance metrics using HTTP Event Collector (HEC) or TCP. The Splunk Add-on for Linux collects data includes: * CPU metrics.* Memory metrics.* Swap metrics.* Mountpoint usage/FS usage.* Network interface traffic.* Disk utilization.* System load.* Process information.* Network protocols information.* IRQ metrics.* TCP connections information.* Thermal information.* System uptime statistics.

https://splunkbase.splunk.com/app/3412/

## Observability & Security

Splunk Add-on for Unix and Linux | Splunkbase

Important: Read upgrade Instructions and test add-on update before deploying to production ***There are changes to default indexes and .conf changes in version 6.0 of Splunk Add-on for Unix and Linux that can break an existing installation if upgrade instructions are not followed in detail.

https://splunkbase.splunk.com/app/833/

## Security Focused

Linux Secure Technology Add-On | Splunkbase

This app provides field extractions and normalisation to the Common Information Model for /var/log/secure and /var/log/auth.log (linux_secure sourcetype). It is intended to replace the security-relevant aspects of the Splunk Add-on for Unix and Linux (Splunk_TA_nix) and as such it's strongly recommended that the Splunk_TA_nix app be removed from your search head before installing this app as they may conflict.

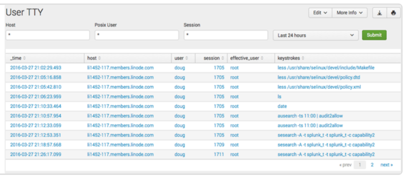
https://splunkbase.splunk.com/app/3476/

## Linux Auditd

Linux Auditd | Splunkbase

Splunk .conf 2018 Edition!

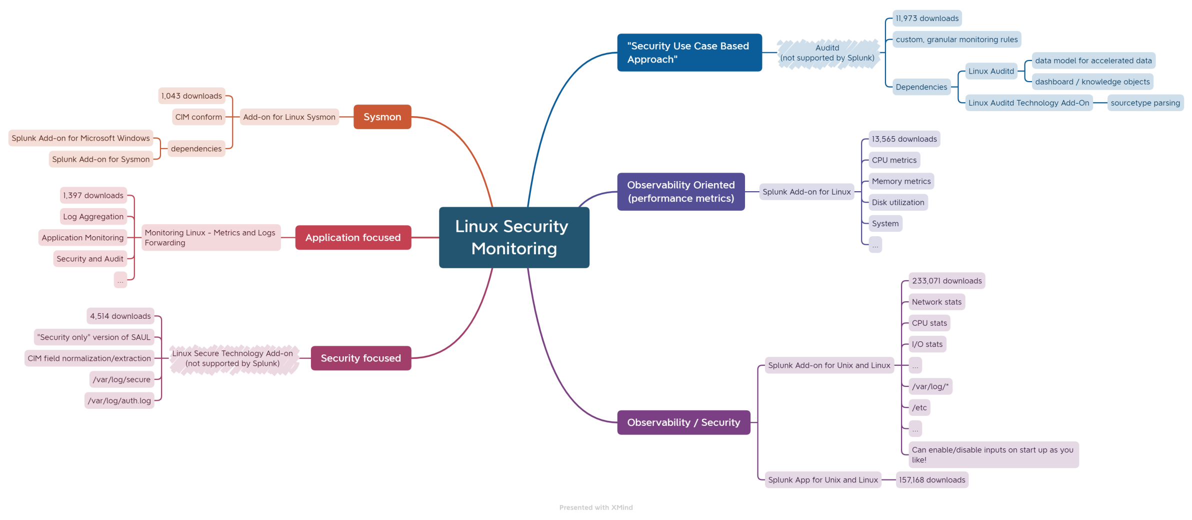
https://splunkbase.splunk.com/app/2642/



| _time | host | user | session | effective_user | keystrokes |
|---|---|---|---|---|---|
| 2016-03-27 21:02:29.493 | li1452-117.members.linode.com | doug | 1705 | root | less /usr/share/selinux/devel/include/Makefile |
| 2016-03-27 21:05:16.858 | li1452-117.members.linode.com | doug | 1705 | root | less /usr/share/selinux/devel/policy.dtd |
| 2016-03-27 21:05:42.810 | li1452-117.members.linode.com | doug | 1705 | root | less /usr/share/selinux/devel/policy.xml |
| 2016-03-27 21:06:23.959 | li1452-117.members.linode.com | doug | 1705 | root | ls |
| 2016-03-27 21:10:33.464 | li1452-117.members.linode.com | doug | 1705 | root | date |
| 2016-03-27 21:10:57.954 | li1452-117.members.linode.com | doug | 1705 | root | ausearch -ts 11:00 \| audit2allow |
| 2016-03-27 21:12:33.059 | li1452-117.members.linode.com | doug | 1705 | root | ausearch -ts 11:00 \| audit2allow |
| 2016-03-27 21:12:53.351 | li1452-117.members.linode.com | doug | 1705 | root | sesearch -A -t splunk_t -t splunk_t -c capability2 |
| 2016-03-27 21:18:57.668 | li1452-117.members.linode.com | doug | 1709 | root | sesearch -A -t splunk_t -t splunk_t -c capability2 |
| 2016-03-27 21:26:17.099 | li1452-117.members.linode.com | doug | 1711 | root | sesearch -A -t splunk_t -t splunk_t -c capability2 |

« prev 1 2 next »

## Feature Comparison


Linux Security Monitoring
Sysmon
Add-on for Linux Sysmon
1,043 downloads
CIM conform
dependencies
Splunk Add-on for Microsoft Windows
Splunk Add-on for Sysmon
Application focused
Monitoring Linux - Metrics and Logs Forwarding
1,397 downloads
Log Aggregation
Application Monitoring
Security and Audit
...
Security focused
Linux Secure Technology Add-on (not supported by Splunk)
4,514 downloads
"Security only" version of SAUL
CIM field normalization/extraction
/var/log/secure
/var/log/auth.log
"Security Use Case Based Approach"
Auditd (not supported by Splunk)
11,973 downloads
custom, granular monitoring rules
Dependencies
Linux Auditd
data model for accelerated data
dashboard / knowledge objects
Linux Auditd Technology Add-On
sourcetype parsing
Observability Oriented (performance metrics)
Splunk Add-on for Linux
13,565 downloads
CPU metrics
Memory metrics
Disk utilization
System
...
Observability / Security
Splunk Add-on for Unix and Linux
233,071 downloads
Network stats
CPU stats
I/O stats
...
/var/log/*
/etc
...
Can enable/disable inputs on start up as you like!
Splunk App for Unix and Linux
157,168 downloads
Presented with XMind

*Which solution should we use?*

# Demo Use Cases

## T1016 - System Network Configuration Discovery

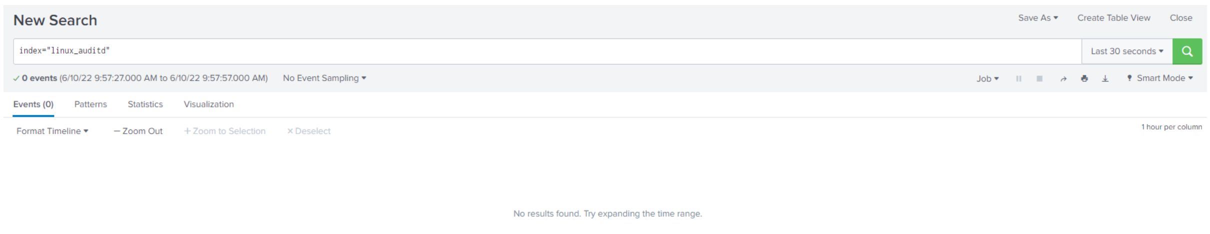
Adversaries my use certain utilities to discover network configurations and settings. This may include IP and MAC addresses, interfaces configurations, open ports, etc.

For this use case we will use the following script to conduct network configuration discovery and trigger alerts in Splunk:

```
# .sh script
if [ -x "$(command -v arp)" ]; then arp -a; else echo "arp is missing from the machine. skipping..."; fi;
if [ -x "$(command -v ifconfig)" ]; then ifconfig; else echo "ifconfig is missing from the machine. skipping..."; fi;
if [ -x "$(command -v ip)" ]; then ip addr; else echo "ip is missing from the machine. skipping..."; fi;
if [ -x "$(command -v netstat)" ]; then netstat -ant | awk '{print $NF}' | grep -v '[a-z]' | sort | uniq -c; else echo "netstat is mis
```

### Auditd Monitoring

After executing the script, we see that there are no log events generated:

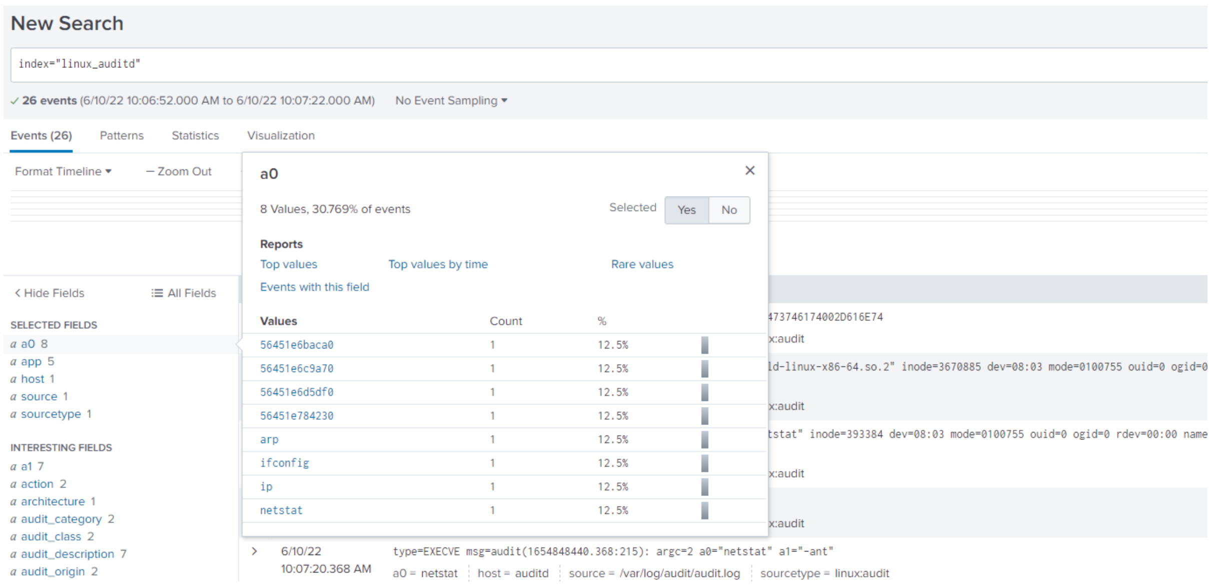


```
index="linux_auditd"
✓ 0 events (6/10/22 9:57:27.000 AM to 6/10/22 9:57:57.000 AM)   No Event Sampling
No results found. Try expanding the time range.
```

This is because we have not specified in the audit rules any conditions for auditd to actively monitor. Opening the file `/etc/audit/rules.d/audit.rules`, we can add the following:

```
-w /usr/sbin/arp -p x -k T1016_network_discovery
# -w /sbin/ip has symlink: /sbin/ip -> /bin/ip*
-w /bin/ip -p x -k T1016_network_discovery
-w /bin/netstat -p x -k T1016_network_discovery
-w /sbin/ifconfig -p x -k T1016_network_discovery
```

Note here that `/sbin/ip` has a symlink to another location, although the `which` command specified this binary. The final binary must be given in the `audit.rules`. Executing the script again, we now see many Splunk events and the name of each command given the `a0` field:



```
index="linux_auditd"
✓ 26 events (6/10/22 10:06:52.000 AM to 6/10/22 10:07:22.000 AM)   No Event Sampling
```

a0

8 Values, 30.769% of events

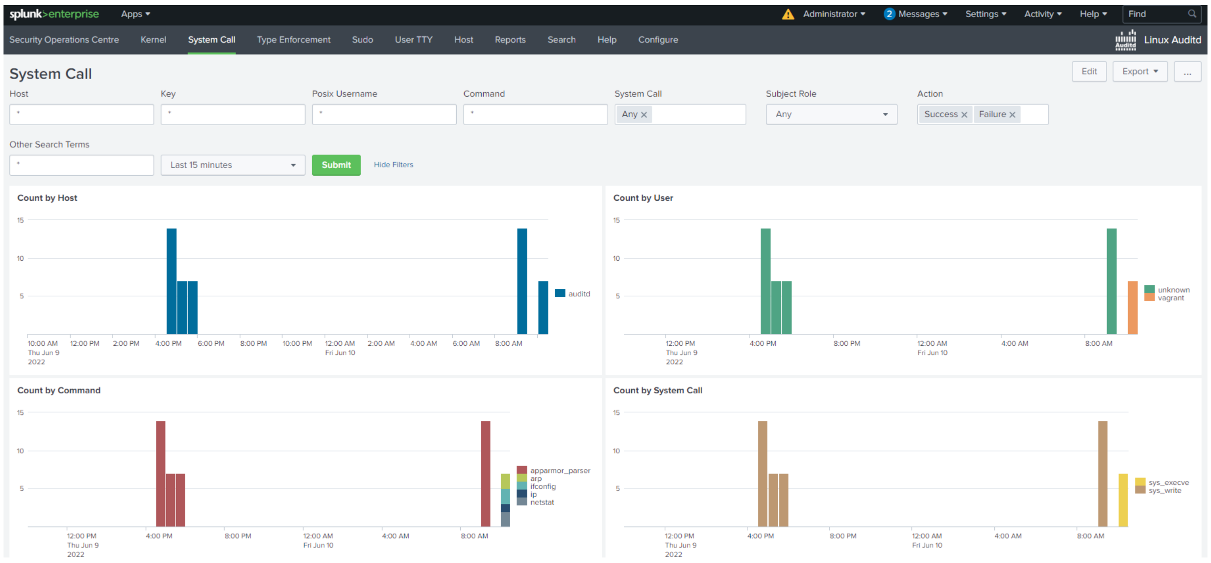
| Values | Count | % |
|---|---|---|
| 56451e6baca0 | 1 | 12.5% |
| 56451e6c9a70 | 1 | 12.5% |
| 56451e6d5df0 | 1 | 12.5% |
| 56451e784230 | 1 | 12.5% |
| arp | 1 | 12.5% |
| ifconfig | 1 | 12.5% |
| ip | 1 | 12.5% |
| netstat | 1 | 12.5% |

SELECTED FIELDS
a a0 8
a app 5
a host 1
a source 1
a sourcetype 1

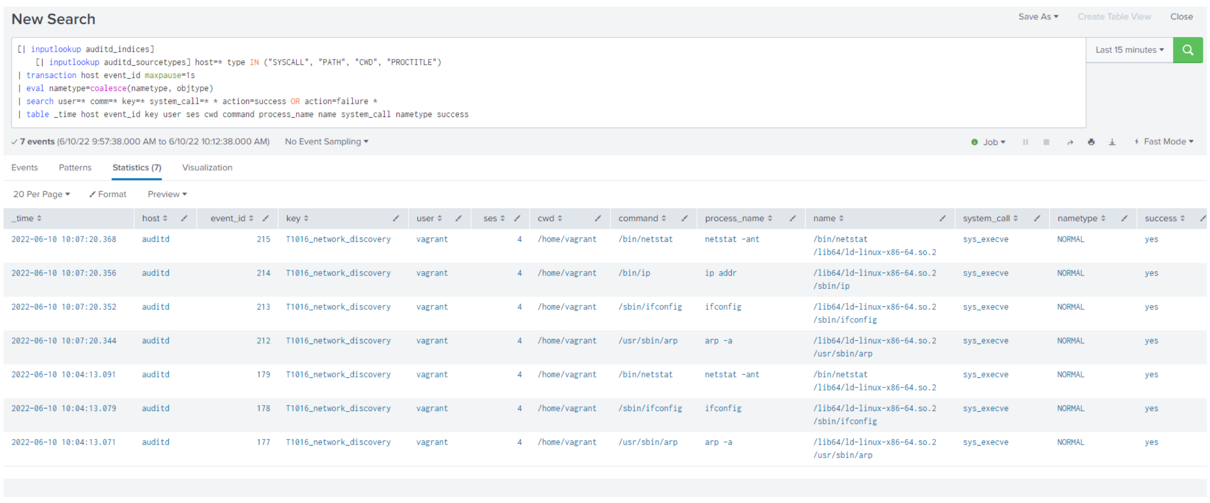
INTERESTING FIELDS
a a1 7
a action 2
a architecture 1
a audit_category 2
a audit_class 2
a audit_description 7
a audit_origin 2

```
473746174002D616E74
x:audit
ld-linux-x86-64.so.2" inode=3670885 dev=08:03 mode=0100755 ouid=0 ogid=0
x:audit
tstat" inode=393384 dev=08:03 mode=0100755 ouid=0 ogid=0 rdev=00:00 name
x:audit
x:audit
> 6/10/22          type=EXECVE msg=audit(1654848440.368:215): argc=2 a0="netstat" a1="-ant"
  10:07:20.368 AM  a0 = netstat | host = auditd | source = /var/log/audit/audit.log | sourcetype = linux:audit
```

Heading over to the Auditd App we can examine some interesting dashboards for these events. For example, in the System Call dashboard we see some visualizations to these recent events:



When using a different SIEM app, such as Splunk Enterprise Security, the Correlation Events table is the most important visualization. It not only gives a great overview of what has happened on system which Auditd has successfully logged, but it also provides a basis for the correlation search logic which will be built in the Splunk ES context:



```
[| inputlookup auditd_indices]
   [| inputlookup auditd_sourcetypes] host=* type IN ("SYSCALL", "PATH", "CWD", "PROCTITLE")
| transaction host event_id maxpause=1s
| eval nametype=coalesce(nametype, objtype)
| search user=* comm=* key=* system_call=* * action=success OR action=failure *
| table _time host event_id key user ses cwd command process_name name system_call nametype success
```

✓ 7 events (6/10/22 9:57:38.000 AM to 6/10/22 10:12:38.000 AM) No Event Sampling

| _time | host | event_id | key | user | ses | cwd | command | process_name | name | system_call | nametype | success |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2022-06-10 10:07:20.368 | auditd | 215 | T1016_network_discovery | vagrant | 4 | /home/vagrant | /bin/netstat | netstat -ant | /bin/netstat /lib64/ld-linux-x86-64.so.2 | sys_execve | NORMAL | yes |
| 2022-06-10 10:07:20.356 | auditd | 214 | T1016_network_discovery | vagrant | 4 | /home/vagrant | /bin/ip | ip addr | /lib64/ld-linux-x86-64.so.2 /sbin/ip | sys_execve | NORMAL | yes |
| 2022-06-10 10:07:20.352 | auditd | 213 | T1016_network_discovery | vagrant | 4 | /home/vagrant | /sbin/ifconfig | ifconfig | /lib64/ld-linux-x86-64.so.2 /sbin/ifconfig | sys_execve | NORMAL | yes |
| 2022-06-10 10:07:20.344 | auditd | 212 | T1016_network_discovery | vagrant | 4 | /home/vagrant | /usr/sbin/arp | arp -a | /lib64/ld-linux-x86-64.so.2 /usr/sbin/arp | sys_execve | NORMAL | yes |
| 2022-06-10 10:04:13.091 | auditd | 179 | T1016_network_discovery | vagrant | 4 | /home/vagrant | /bin/netstat | netstat -ant | /bin/netstat /lib64/ld-linux-x86-64.so.2 | sys_execve | NORMAL | yes |
| 2022-06-10 10:04:13.079 | auditd | 178 | T1016_network_discovery | vagrant | 4 | /home/vagrant | /sbin/ifconfig | ifconfig | /lib64/ld-linux-x86-64.so.2 /sbin/ifconfig | sys_execve | NORMAL | yes |
| 2022-06-10 10:04:13.071 | auditd | 177 | T1016_network_discovery | vagrant | 4 | /home/vagrant | /usr/sbin/arp | arp -a | /lib64/ld-linux-x86-64.so.2 /usr/sbin/arp | sys_execve | NORMAL | yes |

Further audit rules for this use case that could be helpful:

```
-w /usr/bin/ping -p x -k recon
-w /usr/bin/curl -p x -k recon
-w /usr/sbin/dhclient -p x -k recon
-w /usr/sbin/ethtool -p x -k recon
-w /usr/sbin/ifstat -p x -k recon
-w /usr/sbin/ifconfig -p x -k recon
-w /usr/sbin/ifup -p x -k recon
-w /usr/sbin/ifdown -p x -k recon
-w /usr/bin/ip -p x -k recon
# /sbin/ip -> /bin/ip
-w /usr/bin/routel -p x -k recon
-w /usr/bin/routef -p x -k recon
-w /usr/bin/ss -p x -k recon
# ss in /usr/bin/
-w /usr/bin/ssh -p x -k recon
-w /usr/bin/tracepath -p x -k recon
-w /usr/bin/netstat -p x -k recon
-w /usr/sbin/traceroute -p x -k recon
# traceroute -> /etc/alternatives/traceroute.sbin
-w /etc/alternatives/traceroute.sbin -p x -k recon
# /etc/alternatvies/traceroute.sbin -> /usr/bin/traceroute.db
```

```
-w /usr/bin/traceroute.db -p x -k recon
-w /usr/bin/mtr -p x -k recon
-w /usr/sbin/arp -p x -k recon
```

atomic-red-team/T1016.md at master · redcanaryco/atomic-red-team

Adversaries may look for details about the network configuration and settings, such as IP and/or MAC addresses, of systems they access or through information discovery of remote systems. Several operating system administration utilities exist that can be used to gather this information.

https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1016/T1016.md#atomic-test-3---system-network-configuration-discovery

redcanaryco/**atomic-red-team**

Small and highly portable detection tests based on MITRE's ATT&CK.

246 Contributors · 10 Issues · 6k Stars · 2k Forks

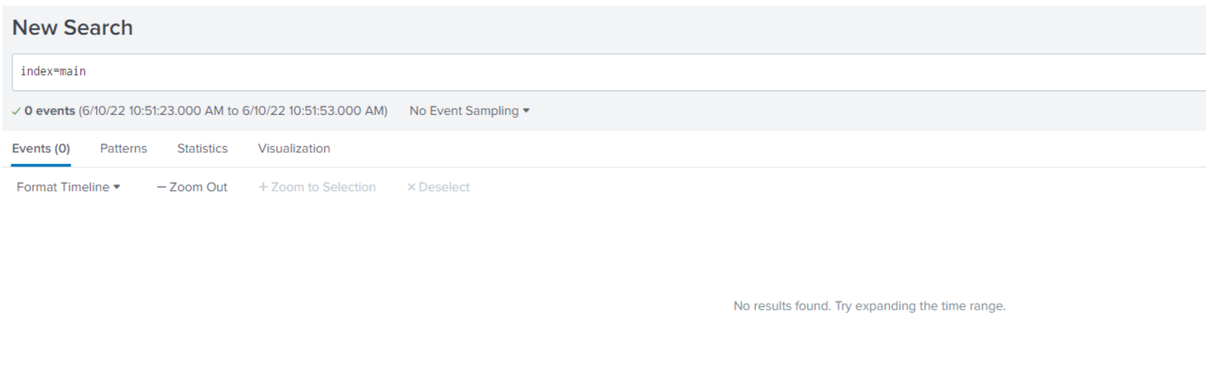
System Network Configuration Discovery

Adversaries may look for details about the network configuration and settings, such as IP and/or MAC addresses, of systems they access or through information discovery of remote systems. Several operating system administration utilities exist that can be used to gather this information. Examples include Arp,

https://attack.mitre.org/techniques/T1016/

## SAUL Monitoring

When the above script is executed now on a system running SAUL and not Auditd, there are no results for the following search:

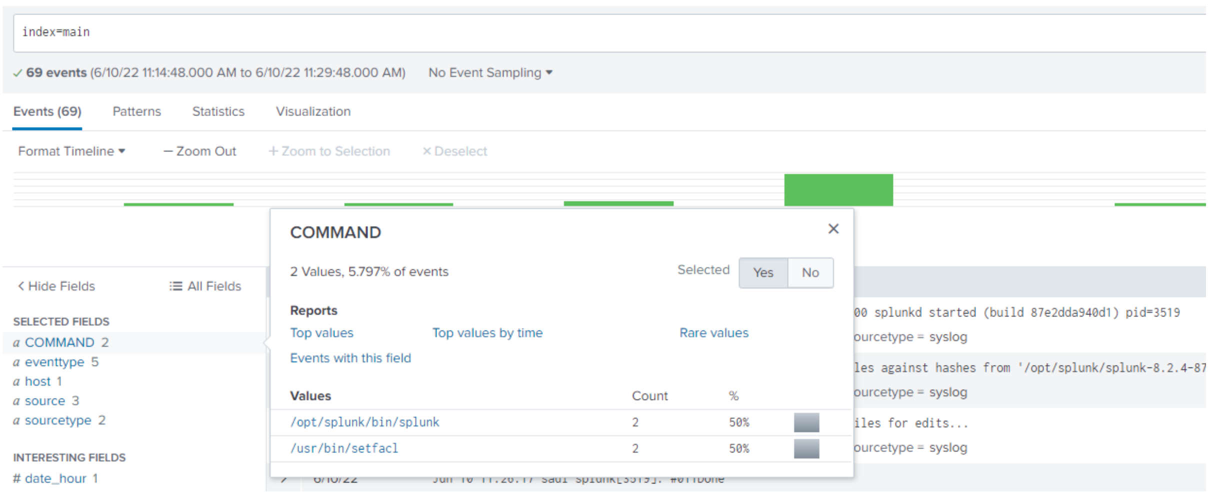

New Search

```
index=main
```

✓ 0 events (6/10/22 10:51:23.000 AM to 6/10/22 10:51:53.000 AM) No Event Sampling ▾

No results found. Try expanding the time range.

Instead, in the last fifteen minutes, only the use of setfacl and restarting Splunk have been logged, but nothing to do with the discovery script that was executed:


```
index=main
```

✓ 69 events (6/10/22 11:14:48.000 AM to 6/10/22 11:29:48.000 AM) No Event Sampling ▾

SELECTED FIELDS
a COMMAND 2
a eventtype 5
a host 1
a source 3
a sourcetype 2

INTERESTING FIELDS
# date_hour 1

COMMAND

2 Values, 5.797% of events — Selected: [✓ Yes] No

Reports: Top values · Top values by time · Rare values · Events with this field

| Values | Count | % |
|---|---|---|
| /opt/splunk/bin/splunk | 2 | 50% |
| /usr/bin/setfacl | 2 | 50% |

```
00 splunkd started (build 87e2dda940d1) pid=3519
ourcetype = syslog
les against hashes from '/opt/splunk/splunk-8.2.4-87
ourcetype = syslog
iles for edits...
ourcetype = syslog
```

Thus, if we want to detect system network configuration discovery, it seems this TA alone will not be sufficient.

## T1003.008 - /etc/passwd and /etc/shadow

Dumping the contents of these files can allow for password cracking. In the case of hashed passwords, one could could a rainbow table to discover the password by finding the corresponding hash from a known base of clear text passwords. John the Ripper also contains the capability for password cracking, using `unshadow` with `/etc/passwd` and `/etc/shadow`.

OS Credential Dumping: /etc/passwd and /etc/shadow

Adversaries may attempt to dump the contents of /etc/passwd and /etc/shadow to enable offline password cracking. Most modern Linux operating systems use a combination of /etc/passwd and /etc/shadow to store user account information including password hashes in /etc/shadow. By default, /etc/shadow is only

https://attack.mitre.org/techniques/T1003/008/

atomic-red-team/T1003.008.md at master · redcanaryco/atomic-red-team

Adversaries may attempt to dump the contents of /etc/passwd and /etc/shadow to enable offline password cracking. Most modern Linux operating systems use a combination of /etc/passwd and /etc/shadow to store user account information including password hashes in /etc/shadow.

https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1003.008/T1003.008.md

redcanaryco/**atomic-red-team**

Small and highly portable detection tests based on MITRE's ATT&CK.

246 Contributors · 10 Issues · 6k Stars · 2k Forks

## Auditd Monitoring

In order to detect read or write to the these files, the following audit rules can be added:

```
-w /etc/passwd -p r -k passwd_read
-w /etc/passwd -p wa -k passwd_write
-w /etc/shadow -p r -k shadow_read
-w /etc/shadow -p wa -k shadow_write
```

We can monitor the events generated and ingested in Splunk for the following tests:

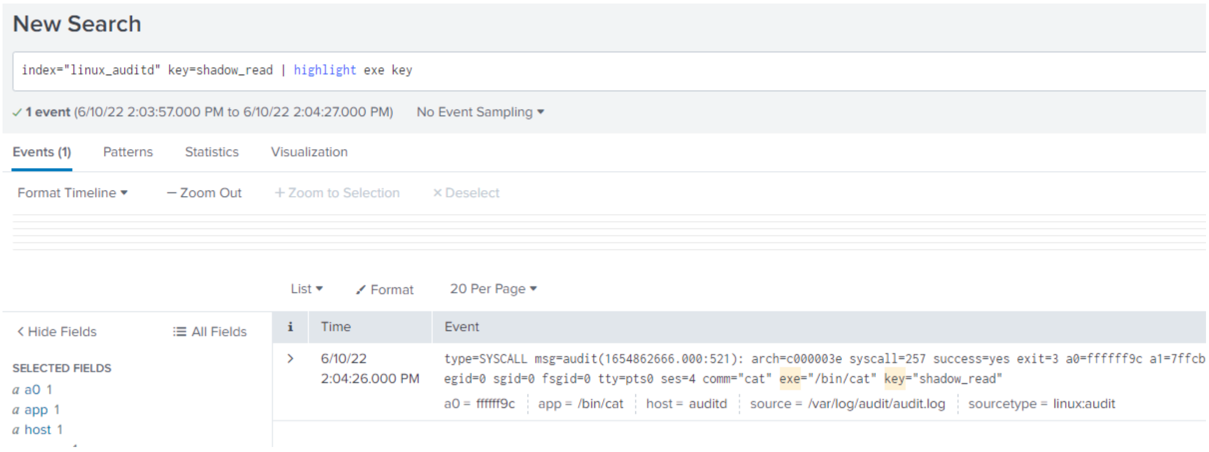
```
# Test 1
sudo cat /etc/shadow > sys_pwds

# Test 2
cat /etc/passwd > sys_accs

# Test 3: bash script
function testcat(){ echo "$(< $1)"; }
testcat /etc/passwd > credential_dump
testcat /etc/shadow >> credential_dump
# execute with
sudo bash credential_access.sh
```

### Test 1

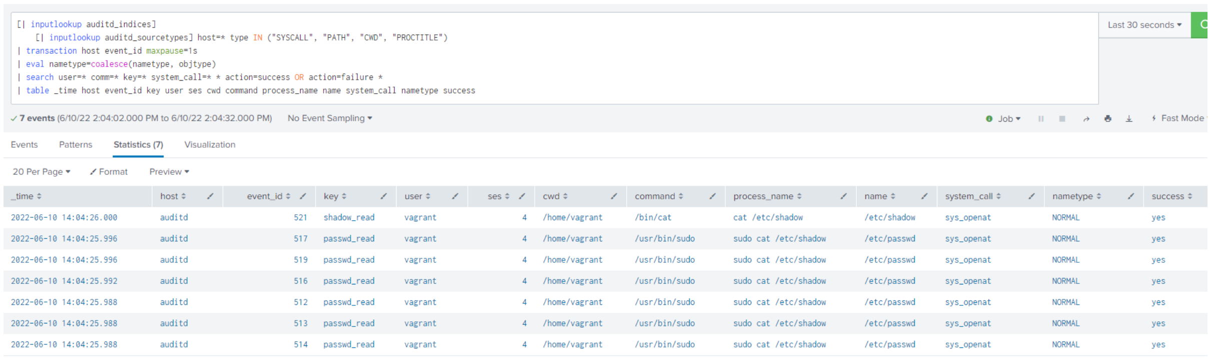
Searching for the key, we see the logged credential dump:



```
index="linux_auditd" key=shadow_read | highlight exe key
```

```
6/10/22 2:04:26.000 PM
type=SYSCALL msg=audit(1654862666.000:521): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7ffcb
egid=0 sgid=0 fsgid=0 tty=pts0 ses=4 comm="cat" exe="/bin/cat" key="shadow_read"
a0 = ffffff9c | app = /bin/cat | host = auditd | source = /var/log/audit/audit.log | sourcetype = linux:audit
```

In the App we get a view of the transaction:



```
[| inputlookup auditd_indices]
    [| inputlookup auditd_sourcetypes] host=* type IN ("SYSCALL", "PATH", "CWD", "PROCTITLE")
| transaction host event_id maxpause=1s
| eval nametype=coalesce(nametype, objtype)
| search user=* comm=* key=* system_call=* * action=success OR action=failure *
| table _time host event_id key user ses cwd command process_name name system_call nametype success
```

Last 30 seconds

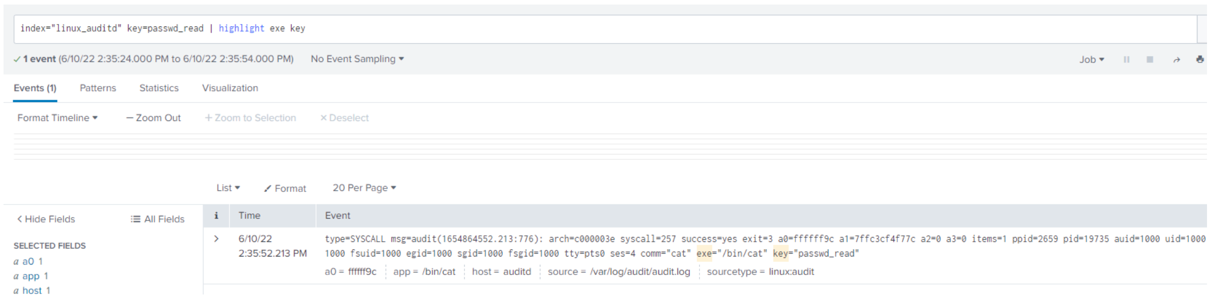
✓ **7 events** (6/10/22 2:04:02.000 PM to 6/10/22 2:04:32.000 PM) No Event Sampling

Events | Patterns | Statistics (7) | Visualization

| _time | host | event_id | key | user | ses | cwd | command | process_name | name | system_call | nametype | success |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2022-06-10 14:04:26.000 | auditd | 521 | shadow_read | vagrant | 4 | /home/vagrant | /bin/cat | cat /etc/shadow | /etc/shadow | sys_openat | NORMAL | yes |
| 2022-06-10 14:04:25.996 | auditd | 517 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo cat /etc/shadow | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:04:25.996 | auditd | 519 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo cat /etc/shadow | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:04:25.992 | auditd | 516 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo cat /etc/shadow | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:04:25.988 | auditd | 512 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo cat /etc/shadow | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:04:25.988 | auditd | 513 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo cat /etc/shadow | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:04:25.988 | auditd | 514 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo cat /etc/shadow | /etc/passwd | sys_openat | NORMAL | yes |

## **Test 2**:

Monitored events:



```
index="linux_auditd" key=passwd_read | highlight exe key
```

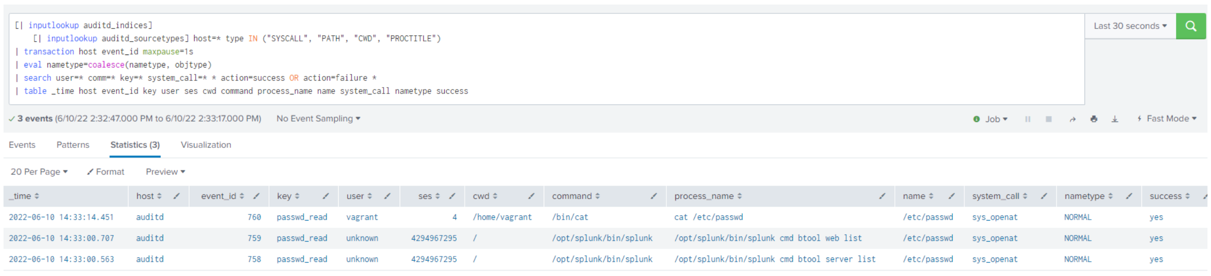
✓ **1 event** (6/10/22 2:35:24.000 PM to 6/10/22 2:35:54.000 PM) No Event Sampling

Events (1) | Patterns | Statistics | Visualization

SELECTED FIELDS
a a0 1
a app 1
a host 1

| Time | Event |
|---|---|
| 6/10/22 2:35:52.213 PM | type=SYSCALL msg=audit(1654864552.213:776): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7ffc3cf4f77c a2=0 a3=0 items=1 ppid=2659 pid=19735 auid=1000 uid=1000 1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=4 comm="cat" exe="/bin/cat" key="passwd_read" <br> a0 = ffffff9c \| app = /bin/cat \| host = auditd \| source = /var/log/audit/audit.log \| sourcetype = linux:audit |

False positives with running Splunk process (not sure what this is…):



```
[| inputlookup auditd_indices]
    [| inputlookup auditd_sourcetypes] host=* type IN ("SYSCALL", "PATH", "CWD", "PROCTITLE")
| transaction host event_id maxpause=1s
| eval nametype=coalesce(nametype, objtype)
| search user=* comm=* key=* system_call=* * action=success OR action=failure *
| table _time host event_id key user ses cwd command process_name name system_call nametype success
```

Last 30 seconds

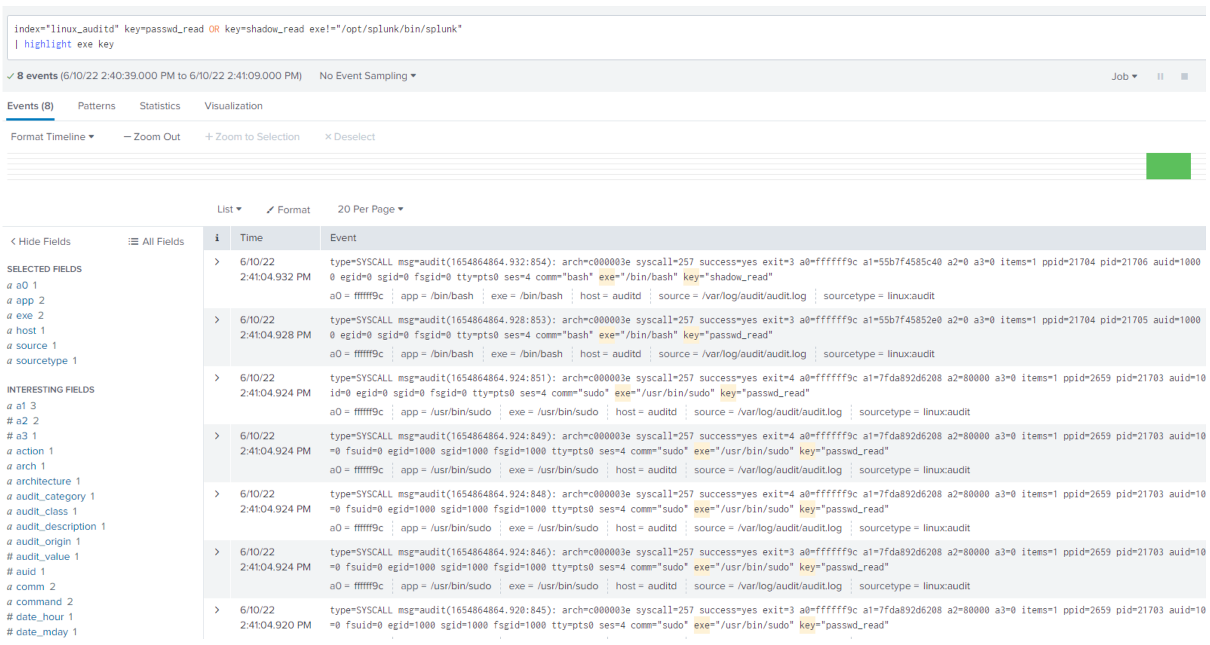
✓ **3 events** (6/10/22 2:32:47.000 PM to 6/10/22 2:33:17.000 PM) No Event Sampling

Events | Patterns | Statistics (3) | Visualization

| _time | host | event_id | key | user | ses | cwd | command | process_name | name | system_call | nametype | success |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2022-06-10 14:33:14.451 | auditd | 760 | passwd_read | vagrant | 4 | /home/vagrant | /bin/cat | cat /etc/passwd | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:33:00.707 | auditd | 759 | passwd_read | unknown | 4294967295 | / | /opt/splunk/bin/splunk | /opt/splunk/bin/splunk cmd btool web list | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:33:00.563 | auditd | 758 | passwd_read | unknown | 4294967295 | / | /opt/splunk/bin/splunk | /opt/splunk/bin/splunk cmd btool server list | /etc/passwd | sys_openat | NORMAL | yes |

## **Test 3**:

After running the bash script, we get the following events:



```
index="linux_auditd" key=passwd_read OR key=shadow_read exe!="/opt/splunk/bin/splunk"
| highlight exe key
```

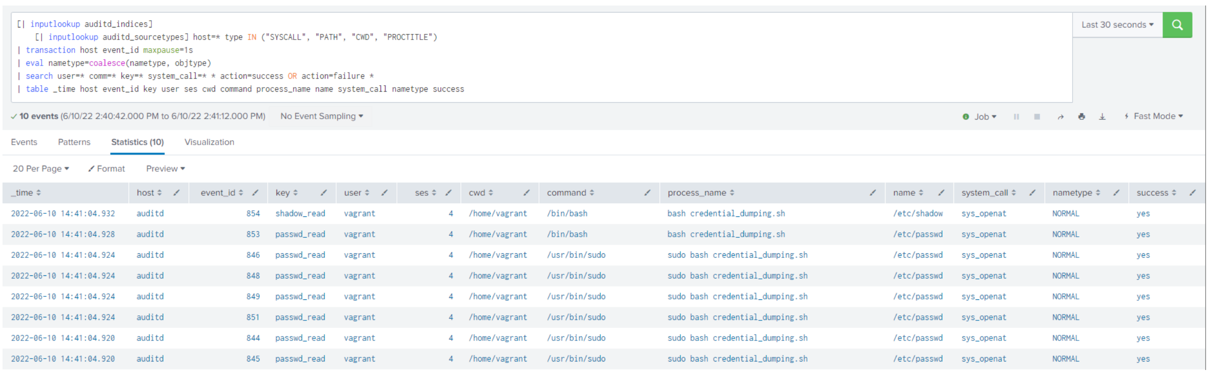
✓ **8 events** (6/10/22 2:40:39.000 PM to 6/10/22 2:41:09.000 PM)   No Event Sampling ▾

SELECTED FIELDS
a a0 1
a app 2
a exe 2
a host 1
a source 1
a sourcetype 1

INTERESTING FIELDS
a a1 3
# a2 2
# a3 1
a action 1
a arch 1
a architecture 1
a audit_category 1
a audit_class 1
a audit_description 1
a audit_origin 1
# audit_value 1
# auid 1
a comm 2
a command 2
# date_hour 1
# date_mday 1

```
i  Time                    Event
>  6/10/22                 type=SYSCALL msg=audit(1654864864.932:854): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=55b7f4585c40 a2=0 a3=0 items=1 ppid=21704 pid=21706 auid=1000
   2:41:04.932 PM          0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=4 comm="bash" exe="/bin/bash" key="shadow_read"
                           a0 = ffffff9c | app = /bin/bash | exe = /bin/bash | host = auditd | source = /var/log/audit/audit.log | sourcetype = linux:audit
>  6/10/22                 type=SYSCALL msg=audit(1654864864.928:853): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=55b7f45852e0 a2=0 a3=0 items=1 ppid=21704 pid=21705 auid=1000
   2:41:04.928 PM          0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=4 comm="bash" exe="/bin/bash" key="passwd_read"
                           a0 = ffffff9c | app = /bin/bash | exe = /bin/bash | host = auditd | source = /var/log/audit/audit.log | sourcetype = linux:audit
>  6/10/22                 type=SYSCALL msg=audit(1654864864.924:851): arch=c000003e syscall=257 success=yes exit=4 a0=ffffff9c a1=7fda892d6208 a2=80000 a3=0 items=1 ppid=2659 pid=21703 auid=10
   2:41:04.924 PM          id=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=4 comm="sudo" exe="/usr/bin/sudo" key="passwd_read"
                           a0 = ffffff9c | app = /usr/bin/sudo | exe = /usr/bin/sudo | host = auditd | source = /var/log/audit/audit.log | sourcetype = linux:audit
>  6/10/22                 type=SYSCALL msg=audit(1654864864.924:849): arch=c000003e syscall=257 success=yes exit=4 a0=ffffff9c a1=7fda892d6208 a2=80000 a3=0 items=1 ppid=2659 pid=21703 auid=10
   2:41:04.924 PM          =0 fsuid=0 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=4 comm="sudo" exe="/usr/bin/sudo" key="passwd_read"
                           a0 = ffffff9c | app = /usr/bin/sudo | exe = /usr/bin/sudo | host = auditd | source = /var/log/audit/audit.log | sourcetype = linux:audit
>  6/10/22                 type=SYSCALL msg=audit(1654864864.924:848): arch=c000003e syscall=257 success=yes exit=4 a0=ffffff9c a1=7fda892d6208 a2=80000 a3=0 items=1 ppid=2659 pid=21703 auid=10
   2:41:04.924 PM          =0 fsuid=0 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=4 comm="sudo" exe="/usr/bin/sudo" key="passwd_read"
                           a0 = ffffff9c | app = /usr/bin/sudo | exe = /usr/bin/sudo | host = auditd | source = /var/log/audit/audit.log | sourcetype = linux:audit
>  6/10/22                 type=SYSCALL msg=audit(1654864864.924:846): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7fda892d6208 a2=80000 a3=0 items=1 ppid=2659 pid=21703 auid=10
   2:41:04.924 PM          =0 fsuid=0 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=4 comm="sudo" exe="/usr/bin/sudo" key="passwd_read"
                           a0 = ffffff9c | app = /usr/bin/sudo | exe = /usr/bin/sudo | host = auditd | source = /var/log/audit/audit.log | sourcetype = linux:audit
>  6/10/22                 type=SYSCALL msg=audit(1654864864.920:845): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7fda892d6208 a2=80000 a3=0 items=1 ppid=2659 pid=21703 auid=10
   2:41:04.920 PM          =0 fsuid=0 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=4 comm="sudo" exe="/usr/bin/sudo" key="passwd_read"
```

Transaction:



```
[| inputlookup auditd_indices]
    [| inputlookup auditd_sourcetypes] host=* type IN ("SYSCALL", "PATH", "CWD", "PROCTITLE")
| transaction host event_id maxpause=1s
| eval nametype=coalesce(nametype, objtype)
| search user=* comm=* key=* system_call=* * action=success OR action=failure *
| table _time host event_id key user ses cwd command process_name name system_call nametype success
```

✓ **10 events** (6/10/22 2:40:42.000 PM to 6/10/22 2:41:12.000 PM)   No Event Sampling ▾

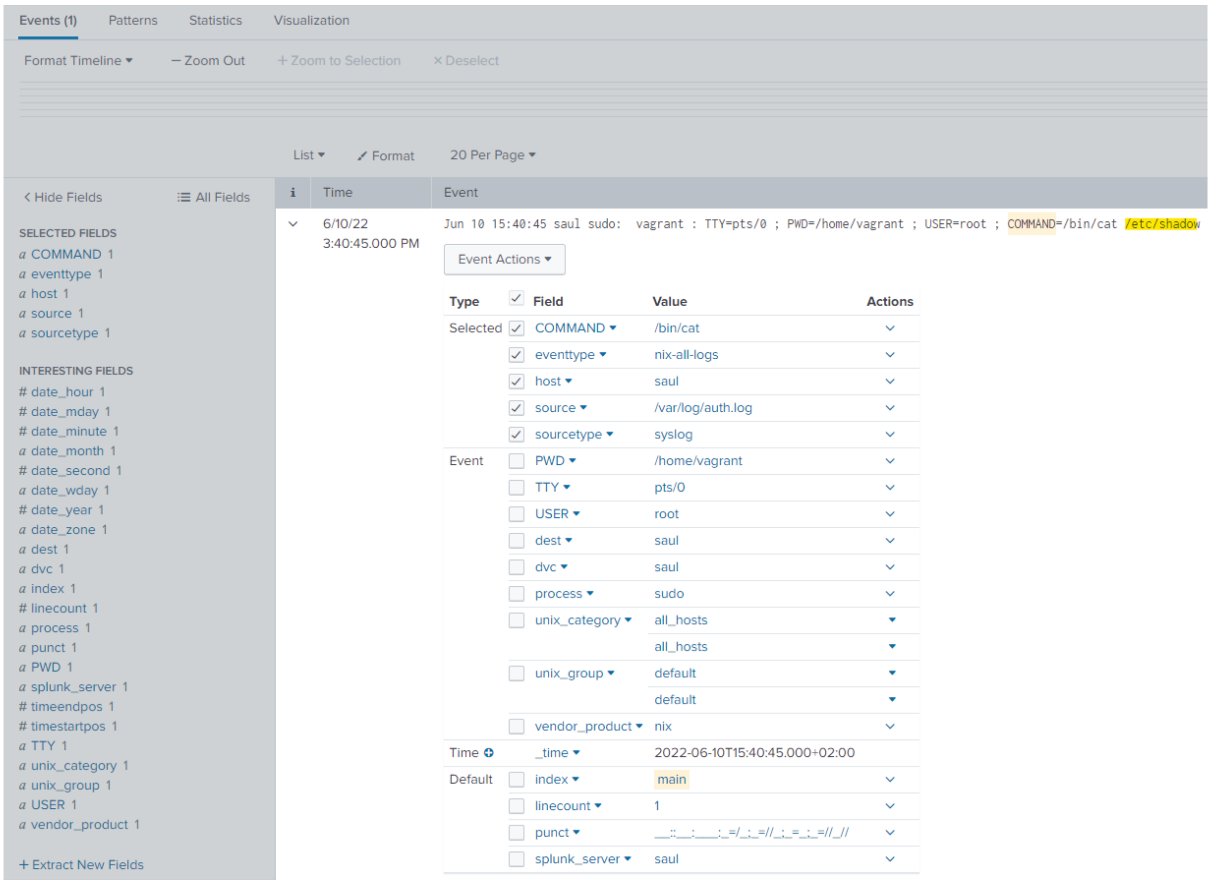
| _time | host | event_id | key | user | ses | cwd | command | process_name | name | system_call | nametype | success |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2022-06-10 14:41:04.932 | auditd | 854 | shadow_read | vagrant | 4 | /home/vagrant | /bin/bash | bash credential_dumping.sh | /etc/shadow | sys_openat | NORMAL | yes |
| 2022-06-10 14:41:04.928 | auditd | 853 | passwd_read | vagrant | 4 | /home/vagrant | /bin/bash | bash credential_dumping.sh | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:41:04.924 | auditd | 846 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo bash credential_dumping.sh | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:41:04.924 | auditd | 848 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo bash credential_dumping.sh | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:41:04.924 | auditd | 849 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo bash credential_dumping.sh | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:41:04.924 | auditd | 851 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo bash credential_dumping.sh | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:41:04.920 | auditd | 844 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo bash credential_dumping.sh | /etc/passwd | sys_openat | NORMAL | yes |
| 2022-06-10 14:41:04.920 | auditd | 845 | passwd_read | vagrant | 4 | /home/vagrant | /usr/bin/sudo | sudo bash credential_dumping.sh | /etc/passwd | sys_openat | NORMAL | yes |

Similar false positives as before and with `cron` jobs.

## SAUL Monitoring

**Test 1**:

Here we do see an event from credential dumping:



SELECTED FIELDS
- a COMMAND 1
- a eventtype 1
- a host 1
- a source 1
- a sourcetype 1

INTERESTING FIELDS
- # date_hour 1
- # date_mday 1
- # date_minute 1
- a date_month 1
- # date_second 1
- a date_wday 1
- # date_year 1
- a date_zone 1
- a dest 1
- a dvc 1
- a index 1
- # linecount 1
- a process 1
- a punct 1
- a PWD 1
- a splunk_server 1
- # timeendpos 1
- # timestartpos 1
- a TTY 1
- a unix_category 1
- a unix_group 1
- a USER 1
- a vendor_product 1

| i | Time | Event |
|---|---|---|
|  | 6/10/22 3:40:45.000 PM | `Jun 10 15:40:45 saul sudo:  vagrant : TTY=pts/0 ; PWD=/home/vagrant ; USER=root ; COMMAND=/bin/cat /etc/shadow` |

| Type | ✓ Field | Value | Actions |
|---|---|---|---|
| Selected | ✓ COMMAND | /bin/cat | ⌄ |
|  | ✓ eventtype | nix-all-logs | ⌄ |
|  | ✓ host | saul | ⌄ |
|  | ✓ source | /var/log/auth.log | ⌄ |
|  | ✓ sourcetype | syslog | ⌄ |
| Event | ☐ PWD | /home/vagrant | ⌄ |
|  | ☐ TTY | pts/0 | ⌄ |
|  | ☐ USER | root | ⌄ |
|  | ☐ dest | saul | ⌄ |
|  | ☐ dvc | saul | ⌄ |
|  | ☐ process | sudo | ⌄ |
|  | ☐ unix_category | all_hosts | ▾ |
|  |  | all_hosts | ▾ |
|  | ☐ unix_group | default | ▾ |
|  |  | default | ▾ |
|  | ☐ vendor_product | nix | ⌄ |
| Time | _time | 2022-06-10T15:40:45.000+02:00 |  |
| Default | ☐ index | main | ⌄ |
|  | ☐ linecount | 1 | ⌄ |
|  | ☐ punct | __::__:____:_=/_;_=//_;_=_;_=//_// | ⌄ |
|  | ☐ splunk_server | saul | ⌄ |

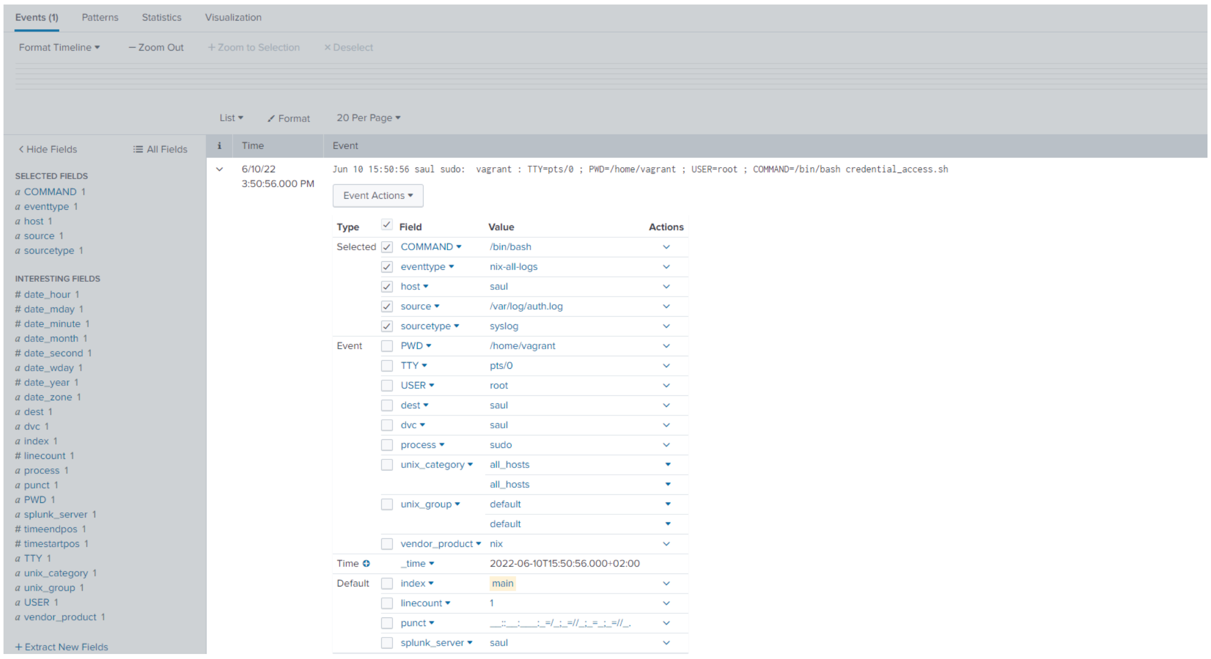
From the field COMMAND it is clear that someone had read the shadow file. The reason this event is logged is because `sudo` was necessary in order to read the hashed passwords. This writes an entry in the `auth.log`.

**Test 2**:

Nothing is logged and read from the monitored inputs.

**Test 3**:

Just as in Test 1, this event is logged because `sudo` was required to read `shadow`:



SELECTED FIELDS
- a COMMAND 1
- a eventtype 1
- a host 1
- a source 1
- a sourcetype 1

INTERESTING FIELDS
- # date_hour 1
- # date_mday 1
- # date_minute 1
- a date_month 1
- # date_second 1
- a date_wday 1
- # date_year 1
- a date_zone 1
- a dest 1
- a dvc 1
- a index 1
- # linecount 1
- a process 1
- a punct 1
- a PWD 1
- a splunk_server 1
- # timeendpos 1
- # timestartpos 1
- a TTY 1
- a unix_category 1
- a unix_group 1
- a USER 1
- a vendor_product 1

| i | Time | Event |
|---|---|---|
|  | 6/10/22 3:50:56.000 PM | `Jun 10 15:50:56 saul sudo:  vagrant : TTY=pts/0 ; PWD=/home/vagrant ; USER=root ; COMMAND=/bin/bash credential_access.sh` |

| Type | ✓ Field | Value | Actions |
|---|---|---|---|
| Selected | ✓ COMMAND | /bin/bash | ⌄ |
|  | ✓ eventtype | nix-all-logs | ⌄ |
|  | ✓ host | saul | ⌄ |
|  | ✓ source | /var/log/auth.log | ⌄ |
|  | ✓ sourcetype | syslog | ⌄ |
| Event | ☐ PWD | /home/vagrant | ⌄ |
|  | ☐ TTY | pts/0 | ⌄ |
|  | ☐ USER | root | ⌄ |
|  | ☐ dest | saul | ⌄ |
|  | ☐ dvc | saul | ⌄ |
|  | ☐ process | sudo | ⌄ |
|  | ☐ unix_category | all_hosts | ▾ |
|  |  | all_hosts | ▾ |
|  | ☐ unix_group | default | ▾ |
|  |  | default | ▾ |
|  | ☐ vendor_product | nix | ⌄ |
| Time | _time | 2022-06-10T15:50:56.000+02:00 |  |
| Default | ☐ index | main | ⌄ |
|  | ☐ linecount | 1 | ⌄ |
|  | ☐ punct | __::__:____:_=/_;_=//_;_=_;_=//_. | ⌄ |
|  | ☐ splunk_server | saul | ⌄ |

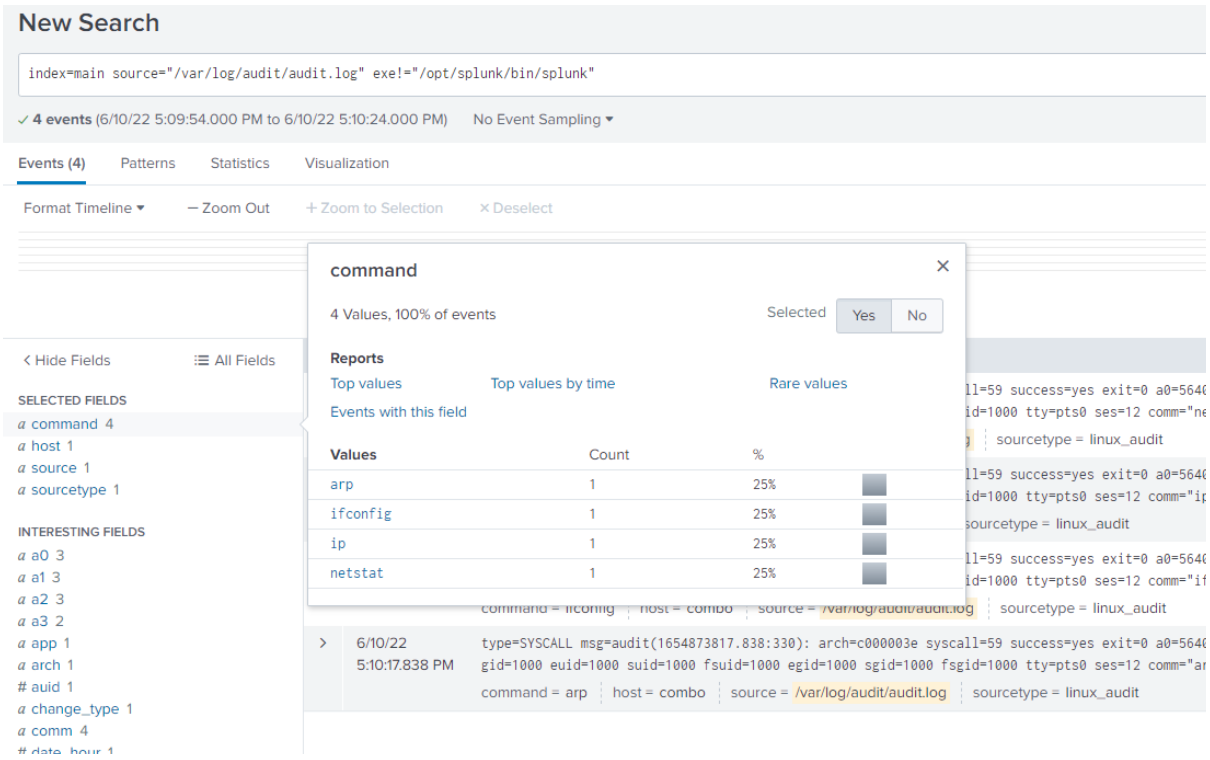
## SAUL + Auditd Consideration

Here we will use an additional instance to setup Splunk with Splunk Add-On for Unix and Linux (SAUL) just as before, but we will install and configure auditd as well. Since SAUL has a monitoring input on `/var/log`, the `audit.log` will be monitored. In fact, we need not install the Auditd TA. SAUL has the following stanza in the `props.conf` already for handling the log parsing:

```
[linux_audit]
REPORT-command = command_for_linux_audit
EVAL-status = if('res'=="failed","failure",'res')
FIELDALIAS-object = id as object
FIELDALIAS-dvc = hostname as dvc
FIELDALIAS-dest = hostname as dest
FIELDALIAS-object_id = id as object_id
EVAL-op = if(op=="PAM:authentication", res, op)
EVAL-vendor_product = if(isnull(vendor_product), "nix", vendor_product)
LOOKUP-action = nix_linux_audit_action_lookup op OUTPUT action,object_category
EVAL-object_attrs= case(type=="ADD_USER" OR type=="USER_MGMT" OR type=="DEL_USER",grp)
EVAL-app = "nix"
EVAL-change_type = "AAA"
EVAL-object = if((type="GRP_MGMT" OR type="DEL_GROUP" or type=="ADD_GROUP") AND isnotnull('grp'),'grp','object')
EVAL-user = case((type=="ADD_USER" OR type=="USER_MGMT" OR type=="DEL_USER" OR type=="USER_CMD") AND isnull('user'),'id',(type=="GRP_M
EVAL-user_name = case((type=="ADD_USER" OR type=="USER_MGMT" OR type=="DEL_USER" OR type=="USER_CMD") AND isnull('user'),'id',(type=="
EVAL-user_id = if(type="GRP_MGMT" OR type="DEL_GROUP" or type=="ADD_GROUP" ,'uid','id')
EVAL-src_user = case((type=="ADD_USER" OR type=="USER_MGMT" OR type=="DEL_USER" OR type=="USER_AUTH" ) AND uid=="0" ,"root",type=="ADD
EVAL-src_user_name = case((type=="ADD_USER" OR type=="USER_MGMT" OR type=="DEL_USER" OR type=="USER_AUTH" ) AND uid=="0" ,"root",type=
EVAL-src_user_id = if(type=="ADD_USER" OR type=="USER_MGMT" OR type=="DEL_USER" OR type=="USER_AUTH" ,'uid','src_user_id')
EVAL-reason = if(type="USER_AUTH" AND (res=="failed" OR res=="failure"),"other",'reason')
```

### Use Case 1: System Network Configuration Discovery

Execution of the network_discovery script has been successfully detected:



```
index=main source="/var/log/audit/audit.log" exe!="/opt/splunk/bin/splunk"
```

✓ 4 events (6/10/22 5:09:54.000 PM to 6/10/22 5:10:24.000 PM) No Event Sampling

SELECTED FIELDS
a command 4
a host 1
a source 1
a sourcetype 1

INTERESTING FIELDS
a a0 3
a a1 3
a a2 3
a a3 2
a app 1
a arch 1
# auid 1
a change_type 1
a comm 4
# date_hour 1

command

4 Values, 100% of events — Selected [✓ Yes] No

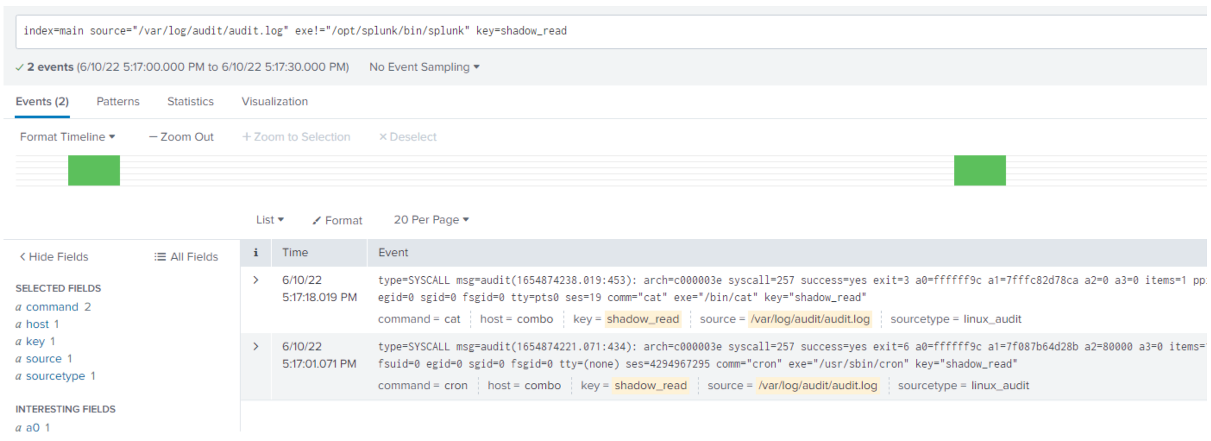
| Values | Count | % |
|---|---|---|
| arp | 1 | 25% |
| ifconfig | 1 | 25% |
| ip | 1 | 25% |
| netstat | 1 | 25% |

```
ll=59 success=yes exit=0 a0=564(
id=1000 tty=pts0 ses=12 comm="ne
sourcetype = linux_audit
ll=59 success=yes exit=0 a0=564(
id=1000 tty=pts0 ses=12 comm="ip
sourcetype = linux_audit
ll=59 success=yes exit=0 a0=564(
id=1000 tty=pts0 ses=12 comm="if
command = ifconfig   host = combo   source = /var/log/audit/audit.log   sourcetype = linux_audit
6/10/22          type=SYSCALL msg=audit(1654873817.838:330): arch=c000003e syscall=59 success=yes exit=0 a0=564(
5:10:17.838 PM   gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=12 comm="ar
                 command = arp   host = combo   source = /var/log/audit/audit.log   sourcetype = linux_audit
```

Note: similar false positives as before.

### Use Case 2: Credential Dumping

**Test 1**:

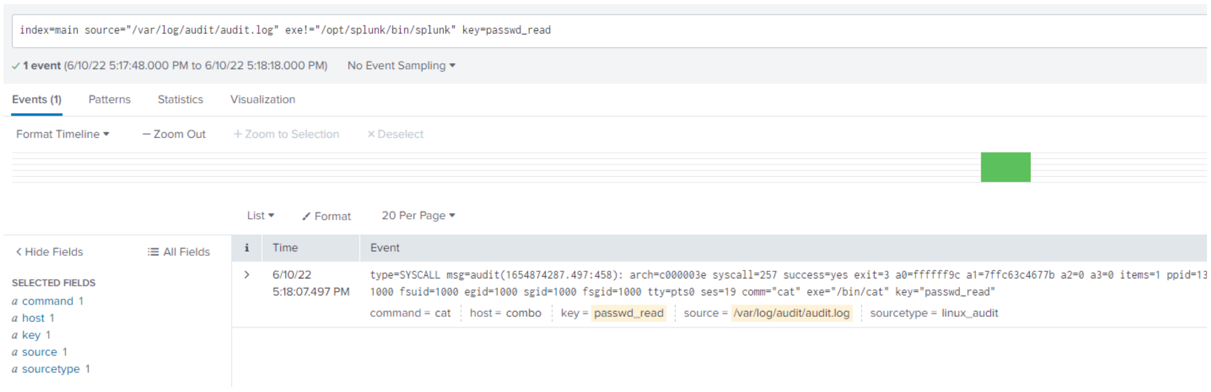


```
index=main source="/var/log/audit/audit.log" exe!="/opt/splunk/bin/splunk" key=shadow_read
```

✓ **2 events** (6/10/22 5:17:00.000 PM to 6/10/22 5:17:30.000 PM) No Event Sampling

| i | Time | Event |
|---|---|---|
| > | 6/10/22 5:17:18.019 PM | `type=SYSCALL msg=audit(1654874238.019:453): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7fffc82d78ca a2=0 a3=0 items=1 pp` `egid=0 sgid=0 fsgid=0 tty=pts0 ses=19 comm="cat" exe="/bin/cat" key="shadow_read"` command = cat host = combo key = shadow_read source = /var/log/audit/audit.log sourcetype = linux_audit |
| > | 6/10/22 5:17:01.071 PM | `type=SYSCALL msg=audit(1654874221.071:434): arch=c000003e syscall=257 success=yes exit=6 a0=ffffff9c a1=7f087b64d28b a2=80000 a3=0 items=` `fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="cron" exe="/usr/sbin/cron" key="shadow_read"` command = cron host = combo key = shadow_read source = /var/log/audit/audit.log sourcetype = linux_audit |

**Test 2**:

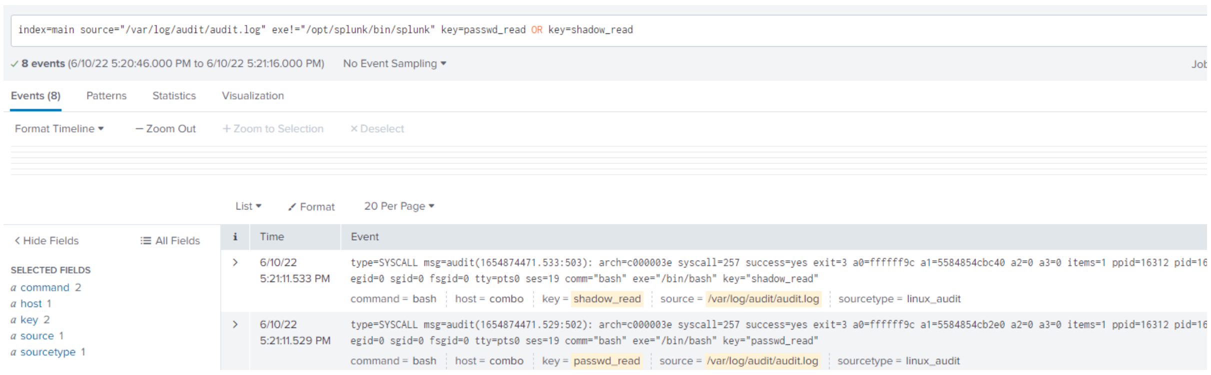


```
index=main source="/var/log/audit/audit.log" exe!="/opt/splunk/bin/splunk" key=passwd_read
```

✓ **1 event** (6/10/22 5:17:48.000 PM to 6/10/22 5:18:18.000 PM) No Event Sampling

| i | Time | Event |
|---|---|---|
| > | 6/10/22 5:18:07.497 PM | `type=SYSCALL msg=audit(1654874287.497:458): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7ffc63c4677b a2=0 a3=0 items=1 ppid=13` `1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=19 comm="cat" exe="/bin/cat" key="passwd_read"` command = cat host = combo key = passwd_read source = /var/log/audit/audit.log sourcetype = linux_audit |

**Test 3**:



```
index=main source="/var/log/audit/audit.log" exe!="/opt/splunk/bin/splunk" key=passwd_read OR key=shadow_read
```

✓ **8 events** (6/10/22 5:20:46.000 PM to 6/10/22 5:21:16.000 PM) No Event Sampling

| i | Time | Event |
|---|---|---|
| > | 6/10/22 5:21:11.533 PM | `type=SYSCALL msg=audit(1654874471.533:503): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=5584854cbc40 a2=0 a3=0 items=1 ppid=16312 pid=16` `egid=0 sgid=0 fsgid=0 tty=pts0 ses=19 comm="bash" exe="/bin/bash" key="shadow_read"` command = bash host = combo key = shadow_read source = /var/log/audit/audit.log sourcetype = linux_audit |
| > | 6/10/22 5:21:11.529 PM | `type=SYSCALL msg=audit(1654874471.529:502): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=5584854cb2e0 a2=0 a3=0 items=1 ppid=16312 pid=16` `egid=0 sgid=0 fsgid=0 tty=pts0 ses=19 comm="bash" exe="/bin/bash" key="passwd_read"` command = bash host = combo key = passwd_read source = /var/log/audit/audit.log sourcetype = linux_audit |

## Pros and Cons

### Auditd

**Pros**

- Easy to install
- Helpful App for SOC overview and building correlation searches
- Granular, “use case based” configuration
- Logs only what you need for specific use cases

**Cons**

- For more complex use cases, a deeper understanding of Linux needed and is more time-consuming to construct and test the rules
- No “out-of-the-box” logging: risk of missing something you weren’t looking for

**Pro/Cons?**

- No way (as far as I know) to get observability data out of it
- Not supported by Splunk

## SAUL

**Pros**

- Can ingest `audit.log` and parse the events without needing the Auditd TA
- Observability with elegant dashboards in App
- Can work with auditd with built in `linux_audit` sourcetype in `props.conf`

**Cons**

- No granular logging, i.e., not a “use case based” approach
- Default monitoring inputs potentially misses A LOT of things we need to detect