redhat.

USER SESSION RECORDING

An Open Source solution

Fraser Tweedale
@hackuador
2017-10-22

ABOUT ME

- Working at Red Hat, Platform Engineering (Security)
- FreeIPA and Dogtag Certificate System

WHY?

THERE IS A DEMAND

Customers need to...

- comply with government or industry regulations
- track what contractors do on our systems
- know who broke our server, and how

AND A DREAM

What companies and governments want:

- Record everything users do
- Store that somewhere safe
- Let us find who did *that thing*
- Show us how they did it

THERE IS A SUPPLY

A number of commercial offerings:

- From application-level proxies on dedicated hardware
- To user-space processes on the target system
- Recording keystrokes, display, commands, apps, URLs, etc.
- Integrated with identity management, and access control
- With central storage, searching, and playback

BUT NOT GOOD ENOUGH

Customers are not satisfied:

- Expensive
- Can’t fix it yourself
- Can’t improve it yourself

WHAT CAN BE BETTER?

The customers want:

- Lower costs
- Open Source, so they can fix, or at least understand it better
- Commercial support

WAIT, WE HAVE IT ALREADY!

Nope, not really:

- `script(1)` plus duct tape
  - popular, but not security-oriented; lots of DIY
- `sudo(8)` I/O logging
  - security-oriented, has searching, but not centralised
- TTY audit with `auditd(8)`
  - security-oriented, can be centralized, only records input

SO, WHAT DO WE NEED?

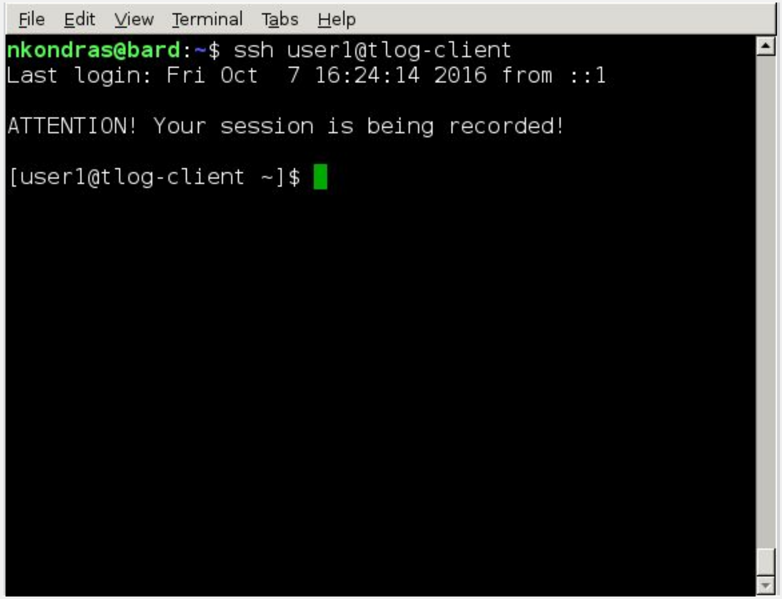
1. Record terminal I/O from userspace
2. Prompt, secure, centralised logging
3. Log kernel audit events, too
4. Search & playback of recorded sessions; correlation
5. Centralised control

1. RECORDING TERMINAL I/O

1. RECORD SESSION I/O

We made a tool for that - **`tlog`**
http://scribery.github.io/tlog

- A shim between the terminal and the shell, started at login
- Converts terminal activity to JSON
- Log to **file**, **syslog** or **journal**
- Playback to terminal



```
nkondras@bard:~$ ssh user1@tlog-client
Last login: Fri Oct  7 16:24:14 2016 from ::1

ATTENTION! Your session is being recorded!

[user1@tlog-client ~]$
```

EXTRA TLOG FEATURES

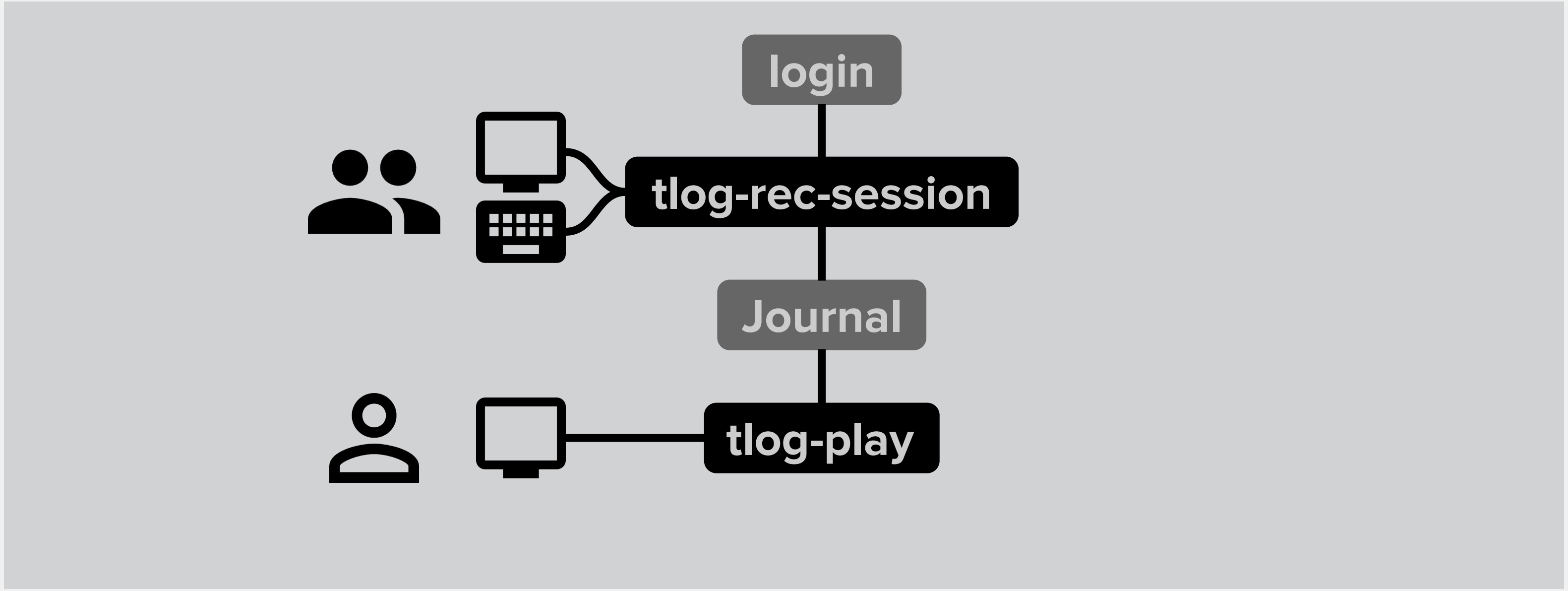
Also control:

- What to record: input/output/window resizes
- “*You are being recorded*” notice
- Low latency vs. low overhead

Basic playback on the terminal:

- From elasticsearch, journal or file

TLOG MINIMAL SETUP

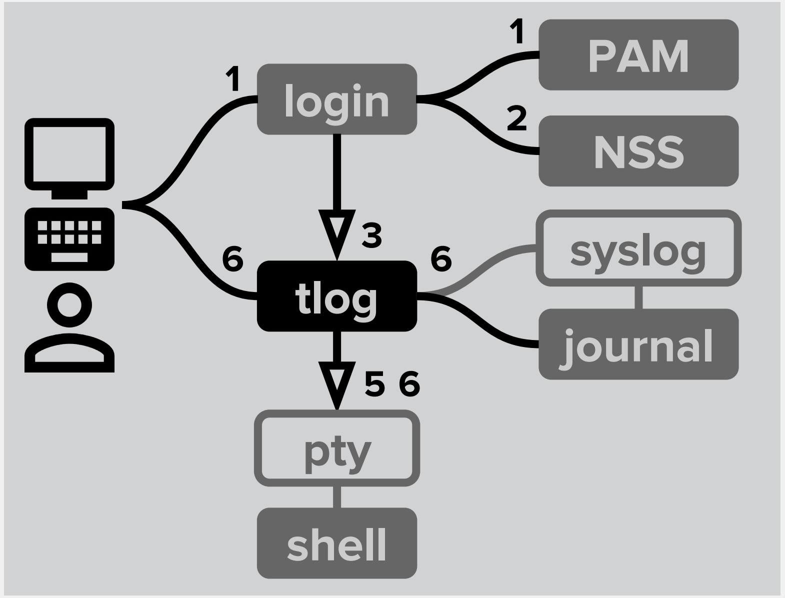


HOW TLOG WORKS?

Console login example

Starting a console session:

1. User authenticates to **`login`** via **PAM**
2. **NSS** tells **`login`**: **`tlog`** is the shell
3. **`login`** starts **`tlog`**
4. env/config tell **`tlog`** the actual shell
5. **`tlog`** starts the actual shell in a **pty**
6. **`tlog`** logs everything passing between its **terminal** and the **pty**, via **`syslog(3)`** or **`sd-journal(3)`**



TLOG SCHEMA

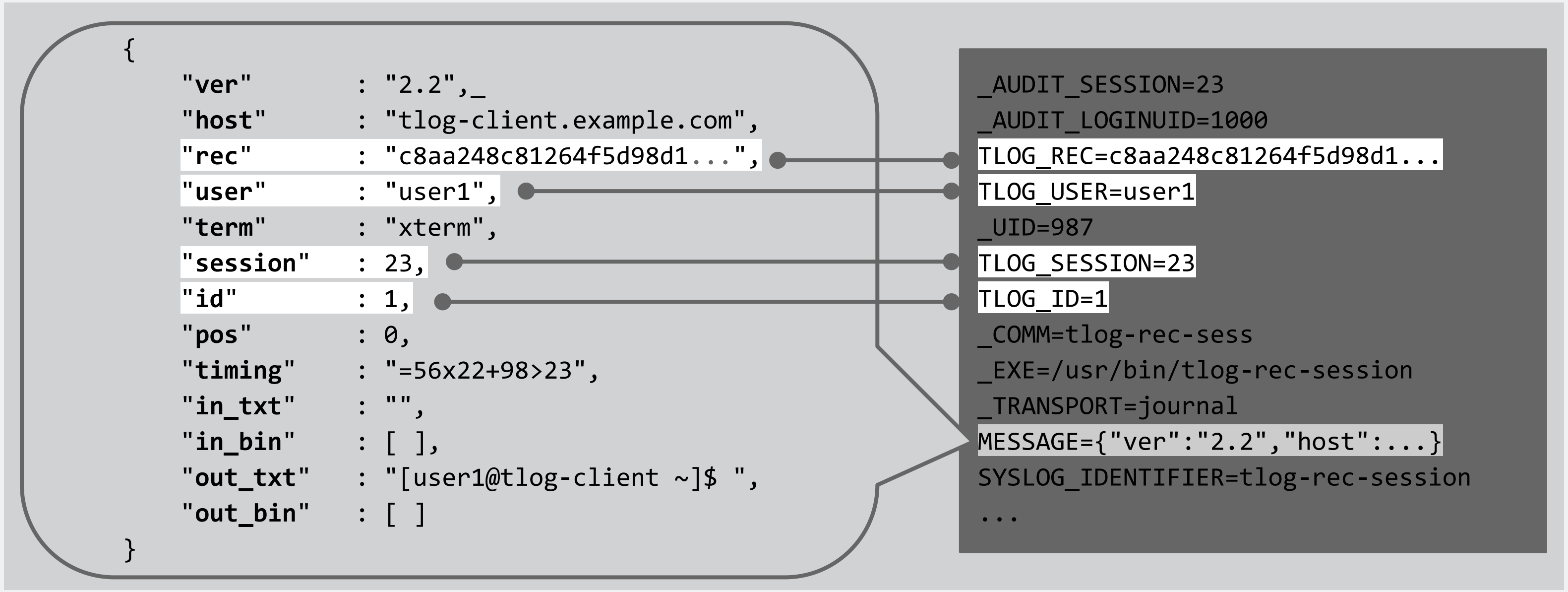
Optimized for streaming and searching:

- Chopped into messages for streaming, which can be merged
- Input and output stored separately
- All I/O preserved
- Invalid UTF-8 stored separately
- Timing separate, ms precision
- Window resizes preserved

```
{
    "ver"       : "2.2",
    "host"      : "tlog-client.example.com",
    "rec"       : "c8aa248c81264f5d98d1...",
    "user"      : "user1",
    "term"      : "xterm",
    "session"   : 23,
    "id"        : 1,
    "pos"       : 0,
    "timing"    : "=56x22+98>23",
    "in_txt"    : "",
    "in_bin"    : [ ],
    "out_txt"   : "[user1@tlog-client ~]$ ",
    "out_bin"   : [ ]
}
```

JOURNAL FORMAT

Exposes key fields



```
{
    "ver"         : "2.2",_
    "host"        : "tlog-client.example.com",
    "rec"         : "c8aa248c81264f5d98d1...",
    "user"        : "user1",
    "term"        : "xterm",
    "session"     : 23,
    "id"          : 1,
    "pos"         : 0,
    "timing"      : "=56x22+98>23",
    "in_txt"      : "",
    "in_bin"      : [ ],
    "out_txt"     : "[user1@tlog-client ~]$ ",
    "out_bin"     : [ ]
}
```

```
_AUDIT_SESSION=23
_AUDIT_LOGINUID=1000
TLOG_REC=c8aa248c81264f5d98d1...
TLOG_USER=user1
_UID=987
TLOG_SESSION=23
TLOG_ID=1
_COMM=tlog-rec-sess
_EXE=/usr/bin/tlog-rec-session
_TRANSPORT=journal
MESSAGE={"ver":"2.2","host":...}
SYSLOG_IDENTIFIER=tlog-rec-session
...
```

2. LOGGING INFRASTRUCTURE

2. LOGGING INFRASTRUCTURE

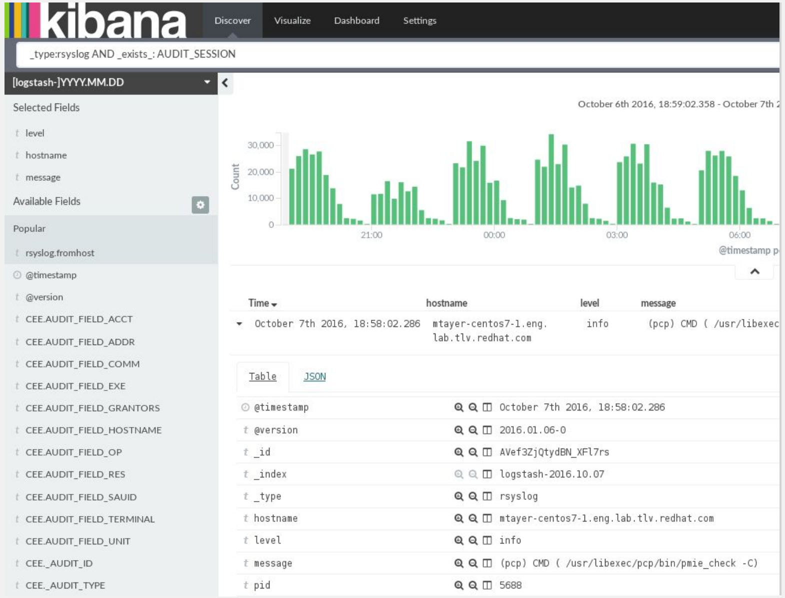
What to take out of the store/search/analyze zoo?

- Open Source
- Scalable
- Active community

YES, ELASTICSEARCH AND KIBANA!

The **ViaQ** project is bringing them to Red Hat product portfolio: https://github.com/ViaQ

- Normalize logs
- Put them into Elasticsearch
- Dashboards and analytics
- Part of OpenShift, coming to OpenStack and other Red Hat products!



_type:rsyslog AND _exists_: AUDIT_SESSION

Time	hostname	level	message
October 7th 2016, 18:58:02.286	mtayer-centos7-1.eng.lab.tlv.redhat.com	info	(pcp) CMD (/usr/libexec

@timestamp	October 7th 2016, 18:58:02.286
@version	2016.01.06-0
_id	AVef3ZjQtydBN_XFl7rs
_index	logstash-2016.10.07
_type	rsyslog
hostname	mtayer-centos7-1.eng.lab.tlv.redhat.com
level	info
message	(pcp) CMD (/usr/libexec/pcp/bin/pmie_check -C)
pid	5688

DELIVER TO ELASTICSEARCH

Any popular logging service:

fluentd

RSYSLOG*

Logstash

Or our coming solution:

ViaQ

* Distributed by Red Hat now

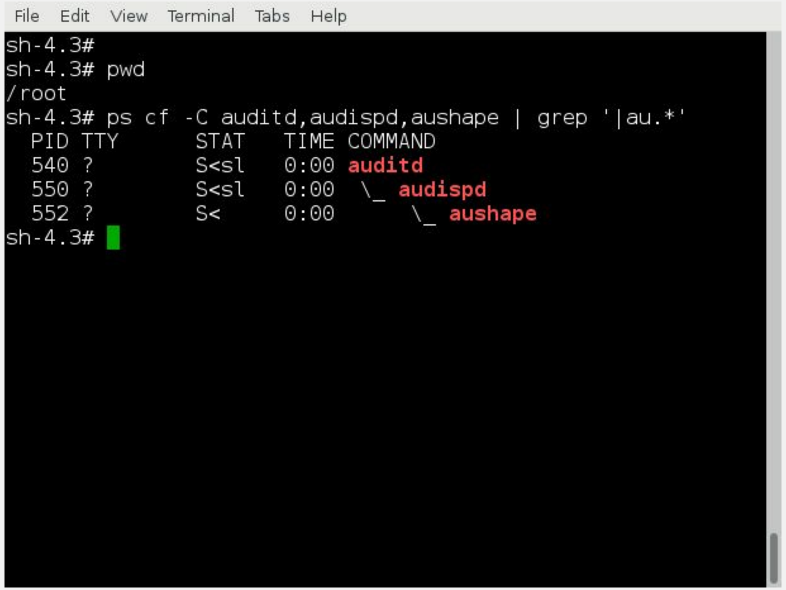
redhat.

3. LOGGING AUDIT EVENTS

AUSHAPE

We made a tool for that too - **aushape**
http://scribery.github.io/aushape/

- Listens for audit events
- Converts them to JSON or XML
- Both have official schemas
- Logs to syslog

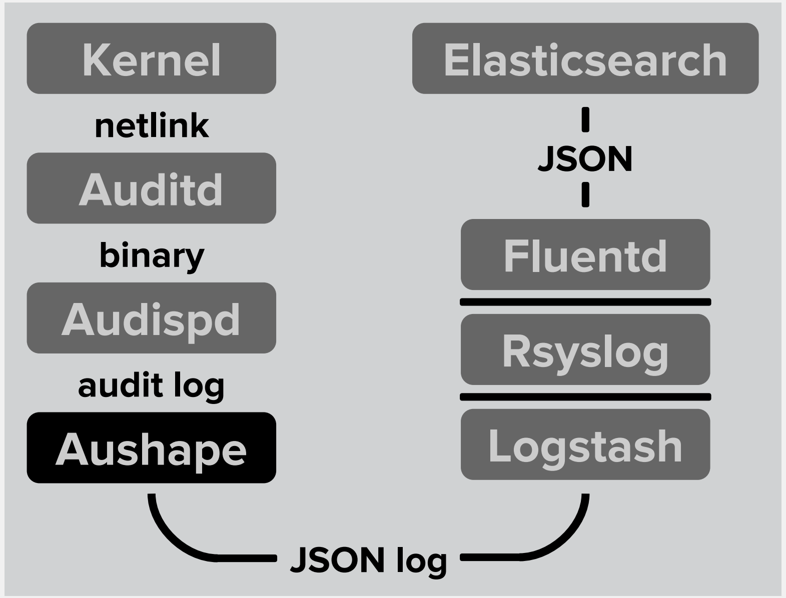


```
sh-4.3#
sh-4.3# pwd
/root
sh-4.3# ps cf -C auditd,audispd,aushape | grep '|au.*'
  PID TTY          STAT   TIME COMMAND
  540 ?            S<sl   0:00 auditd
  550 ?            S<sl   0:00  \_ audispd
  552 ?            S<     0:00      \_ aushape
sh-4.3#
```

HOW AUSHAPE WORKS

From the kernel to Elasticsearch:

- **Kernel** sends messages to **`auditd`**
- **`auditd`** passes messages to **`audispd`**
- **`audispd`** distributes them to plugins, including **`aushape`**
- **`aushape`** formats JSON
- **`aushape`** logs it through **`syslog(3)`**
- **Fluentd/rsyslog/Logstash** deliver it to **Elasticsearch**



AUSHAPE EXAMPLE

A *heavily-trimmed* event

```xml
<event serial="880"
       time="2016-09-28T19:34:44.771+03:00">
    <data>
        <syscall>
            <syscall i="execve" r="59"/>
            <success i="yes"/>
        </syscall>
        <cwd>
            <cwd i="/home/user"/>
        </cwd>
        <execve>
            <a i="ps"/>
        </execve>
    </data>
</event>
```

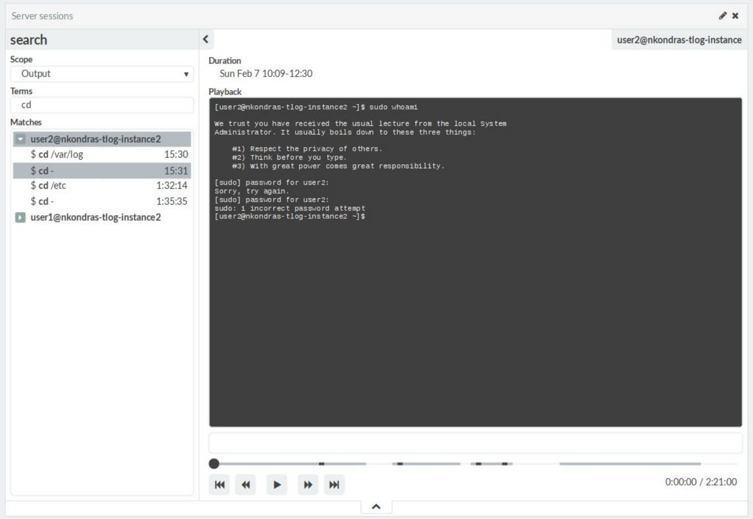
```json
{
    "serial":880,
    "time":"2016-09-28T19:34:44.771+03:00",
    "data":{
        "syscall":{
            "syscall":["execve","59"],
            "success":["yes"]
        },
        "cwd":{
            "cwd":["/home/user"]
        },
        "execve":[
            "ps"
        ]
    }
}
```

4. SESSION PLAYBACK / ANALYSIS

COCKPIT WEB UI

We're building a Web UI

- Playback data from Elasticsearch
- See input, output, commands executed and files accessed
- Search for input, output, commands and files
- Reuse and integrate
- PoC: [Cockpit]() plugin, journal storage



Sun Feb 7 10:09-12:30

user2@nkondras-tlog-instance2	
$ cd /var/log	15:30
$ cd -	15:31
$ cd /etc	1:32:14
$ cd -	1:35:35
user1@nkondras-tlog-instance2	

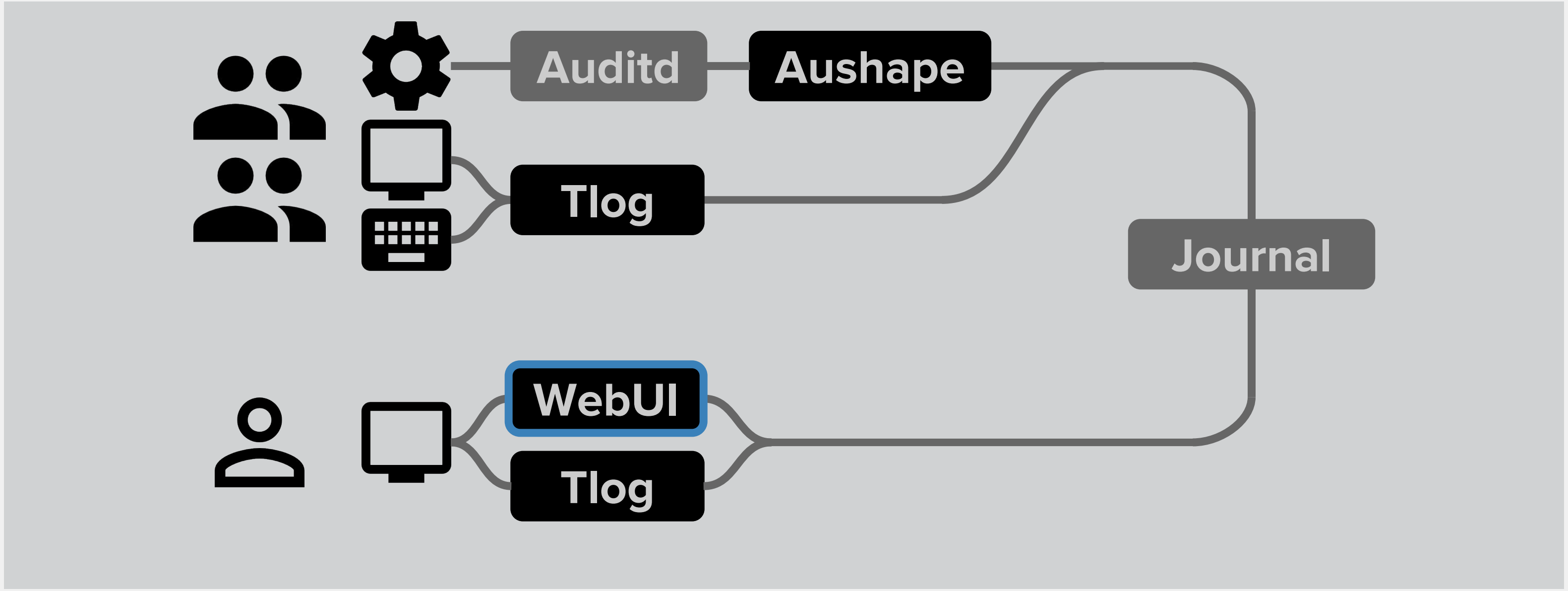
```
[user2@nkondras-tlog-instance2 ~]$ sudo whoami

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for user2:
Sorry, try again.
[sudo] password for user2:
sudo: 1 incorrect password attempt
[user2@nkondras-tlog-instance2 ~]$
```

COCKPIT SCENARIO

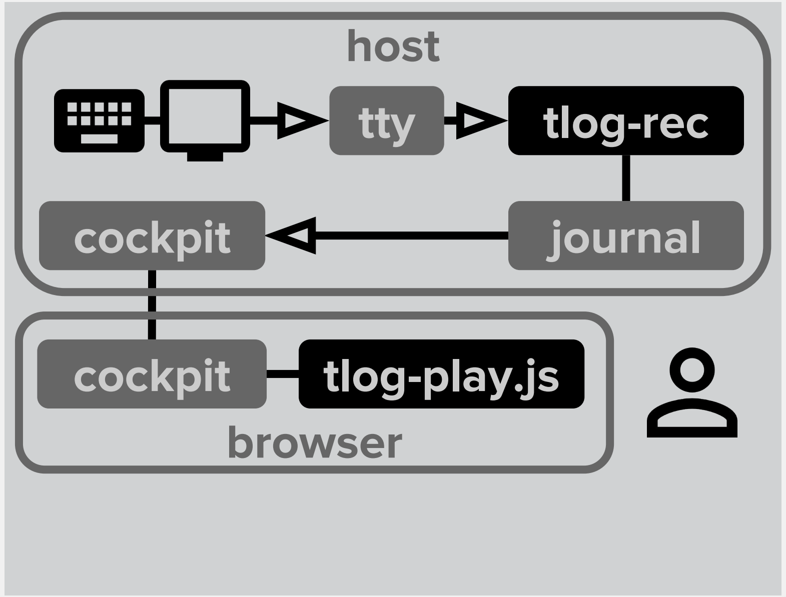


HOW COCKPIT UI WORKS?

Code in development

Setup for recordings in Cockpit:

- **`tlog`** logs to **Journal**, adding a **recording ID** field
- To list recordings, **Cockpit** looks for **`tlog`** messages in Journal, groups by **recording ID**
- **Cockpit JavaScript-based player** reads and plays back Journal messages with **recording ID**.

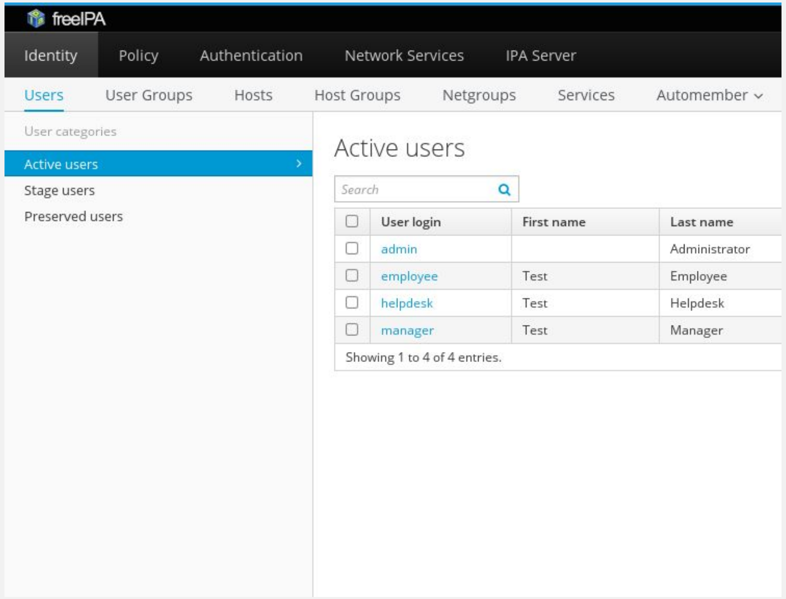


5. CENTRALISED CONTROL

5. CENTRALISED CONTROL

Naturally, **FreeIPA** and **SSSD**!

- Manage domains, hosts, groups, users, and more
- Cache credentials and authenticate offline
- Session Recording control linked to **HBAC** rules



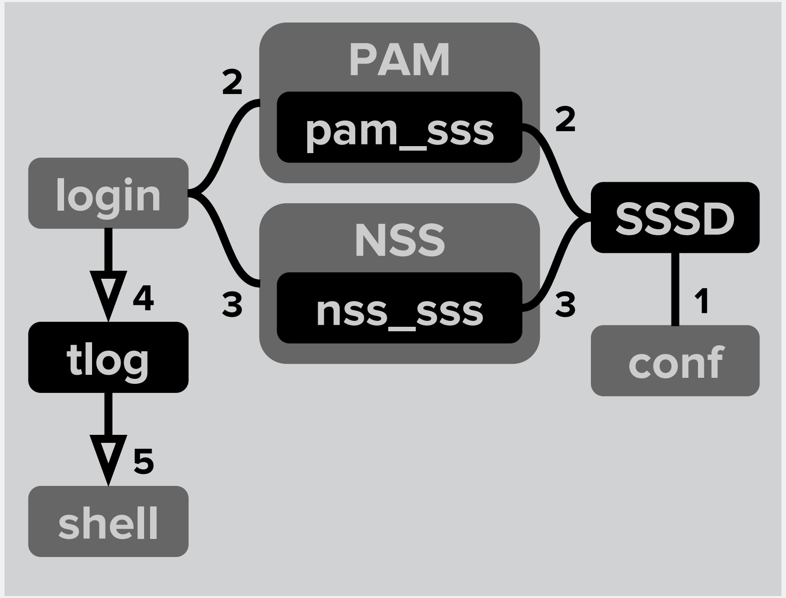
	User login	First name	Last name
☐	admin		Administrator
☐	employee	Test	Employee
☐	helpdesk	Test	Helpdesk
☐	manager	Test	Manager

Showing 1 to 4 of 4 entries.

CONTROL TLOG WITH SSSD

When a recorded user logs in:

1. **SSSD** finds a match for the user in its **configuration**
2. **`pam_sss`** stores the actual user **shell** in the PAM **environment**
3. **`nss_sss`** tells **`login`** "shell is **`tlog`**"
4. **`login`** starts **`tlog`** with PAM env
5. **`tlog`** starts the actual user **shell** retrieved from **environment**



CONTROL TLOG WITH FREEIPA

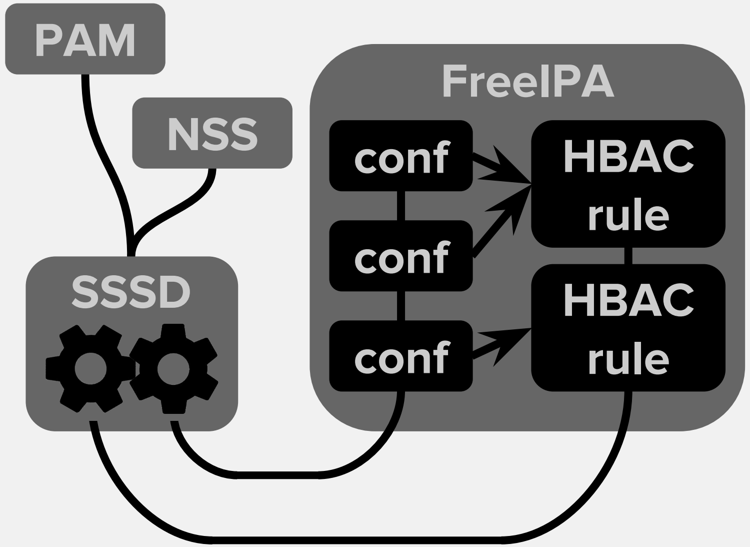
The plan:

Which users to record on which hosts:

- Recording **configurations** linked to **HBAC** rules

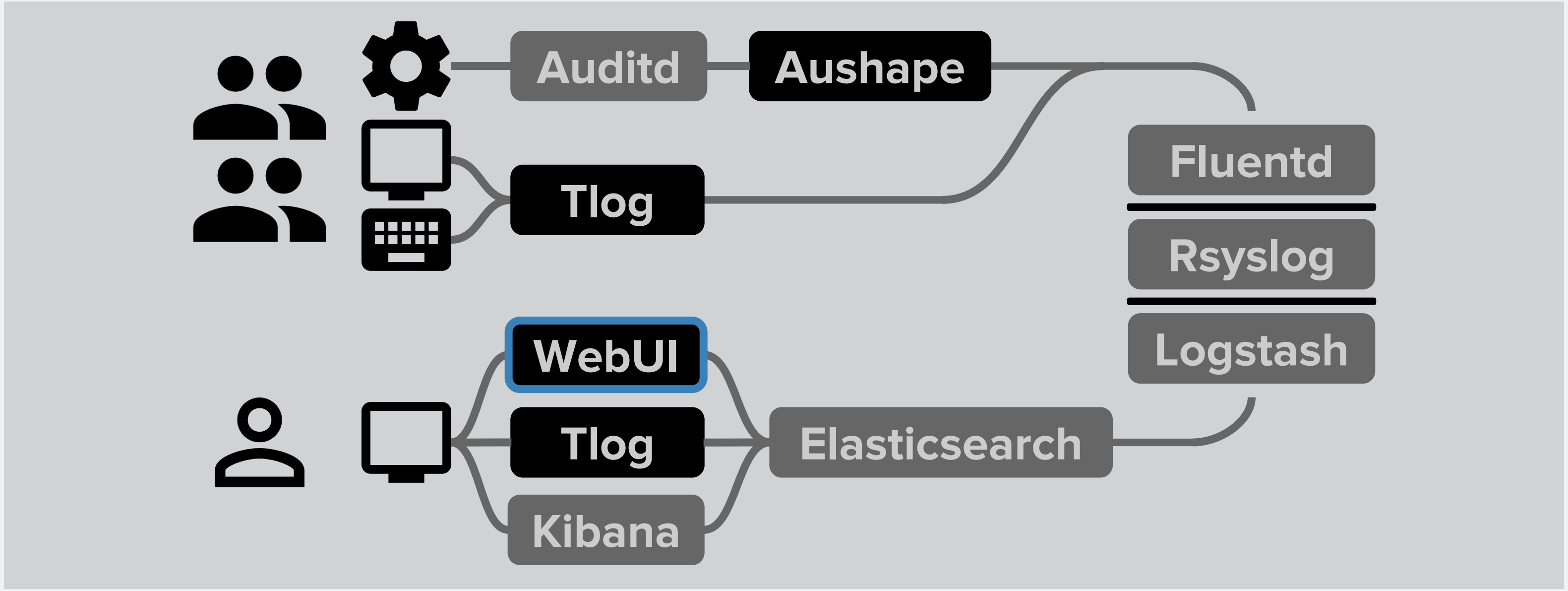
When users login:

- **SSSD** fetches applicable rules
- **SSSD** decides if recording is enabled
- Proceed as on previous slide



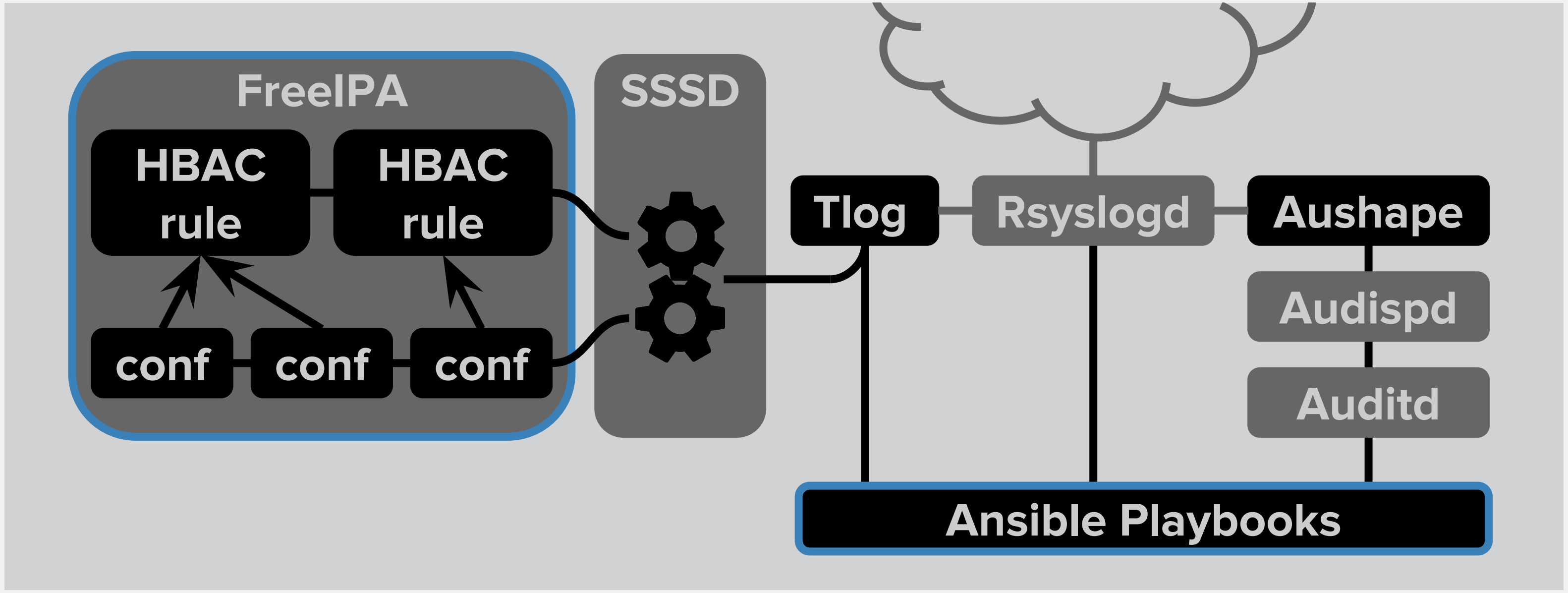
OUR APPROACH

Recording, delivery and storage



OUR APPROACH

Control



CHALLENGES

TLOG CHALLENGES

- How not to record passwords
- Detect graphical sessions and don’t record under them
- Charset conversion
  - Use iconv, and keep original text
- Session playback challenges
  - Seek, rewind, resize

AUSHAPE CHALLENGES

- Audit log is a mess
    - Can’t fix; track all the cases, use what `auditd` knows
- Somehow generate coherent schemas
    - Keep schema simple, use `auditd` record/field dictionaries

WEB UI CHALLENGES

On the road to first release for Cockpit:

- Journal has limited search capability
- Correlation with audit logs

SECURITY CHALLENGES

- Bypass session recording
- Privilege escalation vectors
- `tlog` alone cannot tell the whole story

TRY IT

TRY TLOG

https://github.com/Scribery/tlog

- Download and install a release RPM, or
- Build from source, dependencies:
  - `json-c-devel` / `libjson-c-dev`
  - `libcurl-devel` / `libcurl4-*-dev`
  - `systemd-devel/libsystemd-journal-dev`
- Log to and playback from file
  - Easiest, good for testing
- Log to and playback from Elasticsearch
- Instructions in `README.md`
- Submit issues, suggestions and pull requests!

TRY AUSHAPE

https://github.com/Scribery/aushape

- Download and install a release RPM, or
- Build from source
    - Only `audit-libs-devel` / `libauparse-dev` is required
- Convert your own `/var/log/audit/audit.log` single-shot
    - Try both JSON and XML
- Set up live forwarding to Elasticsearch
- Instructions in `README.md`
- Submit issues, suggestions and pull requests!

TRY COCKPIT UI

https://github.com/Scribery/cockpit/tree/scribery

- Checkout our scribery branch
- Build and run from source
  - Read `HACKING.md`
- Install `tlog`
- Set writer to “`journal`” in `/etc/tlog/tlog-rec-session.conf`
- Create a user with shell set to `/usr/bin/tlog-rec-session`
- Login as that user and do some stuff
- Checkout “Session Recording” page at http://localhost:9090

redhat.

THANK YOU

User Session Recording Project
http://scribery.github.io/