redhat.

USER SESSION RECORDING FOR THE ENTERPRISE

An Open-Source Effort by Red Hat

Nikolai Kondrashov
Software Engineer
FOSDEM 2017

INTRODUCTION

Nikolai “spbnick” Kondrashov, a software engineer

- Working at Red Hat Identity Management Group
- Sometimes helping SSSD
- Maintaining FreeRADIUS packages
- Focusing on the User Session Recording project
- Founder and (still) maintainer of the DIGImend project
- Flirting with embedded as a hobby

WHY?

THERE IS A DEMAND

Customers have been telling us for a long time:

- We need to comply with government regulations
- We need to track what contractors do on our systems
- We need to know who broke our server and how

AND A DREAM

What people 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 great 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

Still people are not satisfied:

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

WHAT CAN BE BETTER?

The customers want:

- Free (as in beer), who doesn't?
- Open-Source, so they can fix, or at least understand it better
- Yet still has support (I know a company!)

WAIT, WE HAVE IT ALREADY!

Nope, not really:

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

WHAT?

SO, WHAT DO WE NEED?

Hottest features requested:

- Record what the user enters, sees on the screen, executes, accesses
- Get it off the machine ASAP, and store centrally and securely
- Search, analyze, and correlate with other events
- Playback in real time, or later
- Control centrally

SOUNDS FAMILIAR!

Let’s do it with logs!

- Audit system records processes executed, files accessed
- Logging servers know how to deliver
- There’s a whole zoo of storing/searching/analyzing solutions

FIRST...

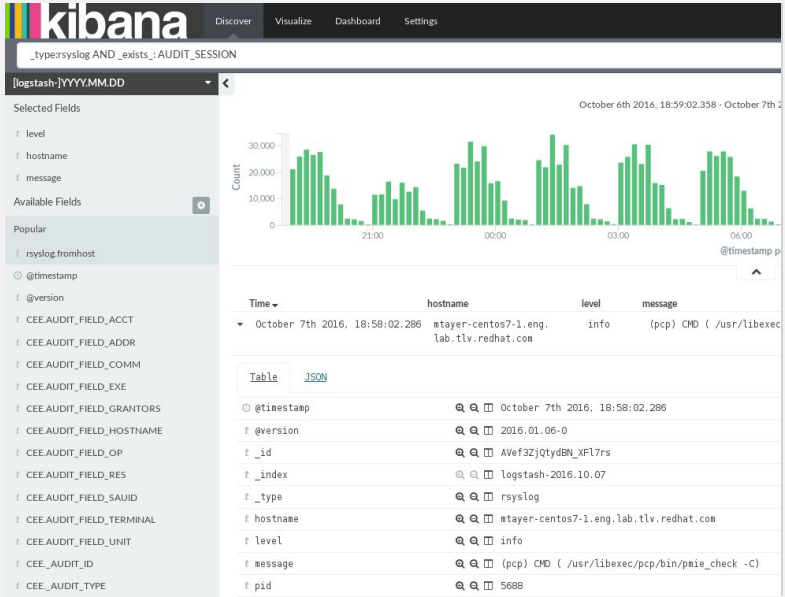
What to take out of the store/search/analyze zoo?

- Open-Source
- Scalable
- Hip

YES, ELASTICSEARCH AND KIBANA!

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

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



THEN...

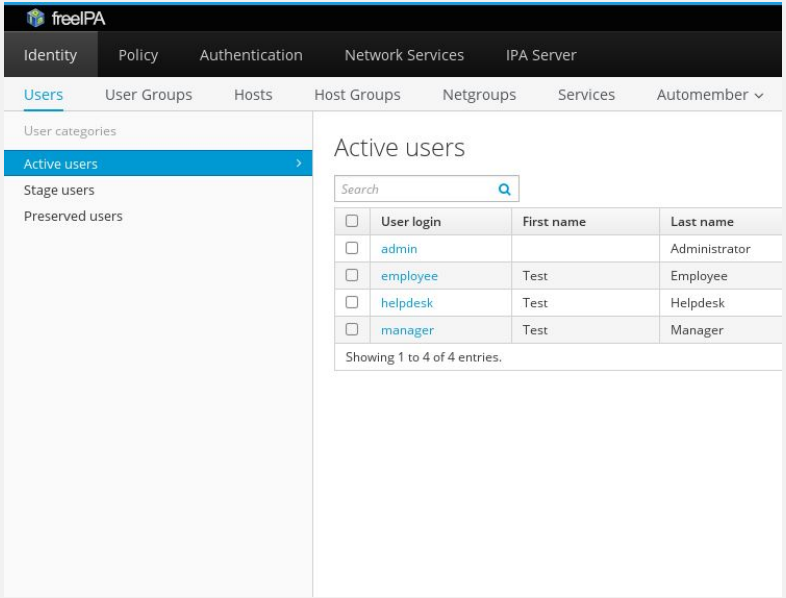
How can we:

- Control centrally what, where and whom to record?
- Log what user enters and sees?
- Make sense of audit logs?
- Deliver to Elasticsearch?
- Play everything back?

CENTRAL CONTROL?

Naturally, FreeIPA and SSSD!

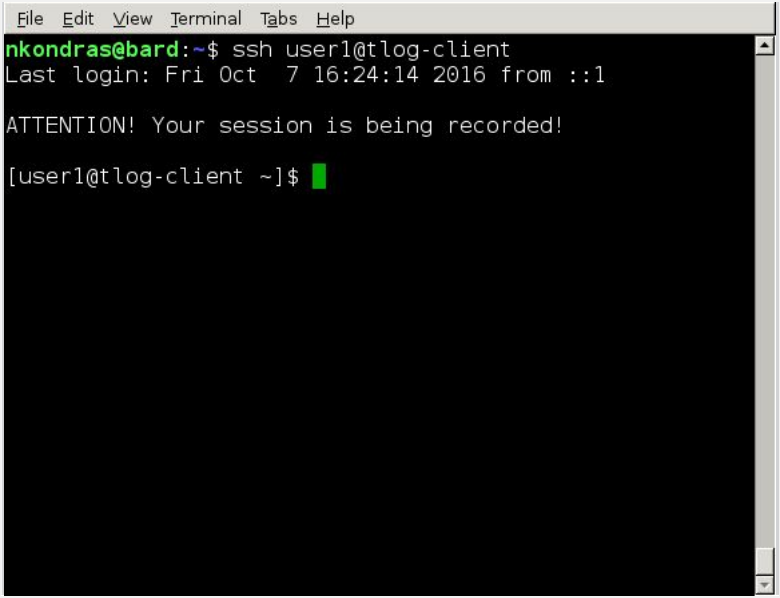
- Manage domains, hosts, groups, users, and more
- Cache credentials and authenticate offline
- Session Recording control in development



LOG INPUT AND OUTPUT?

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

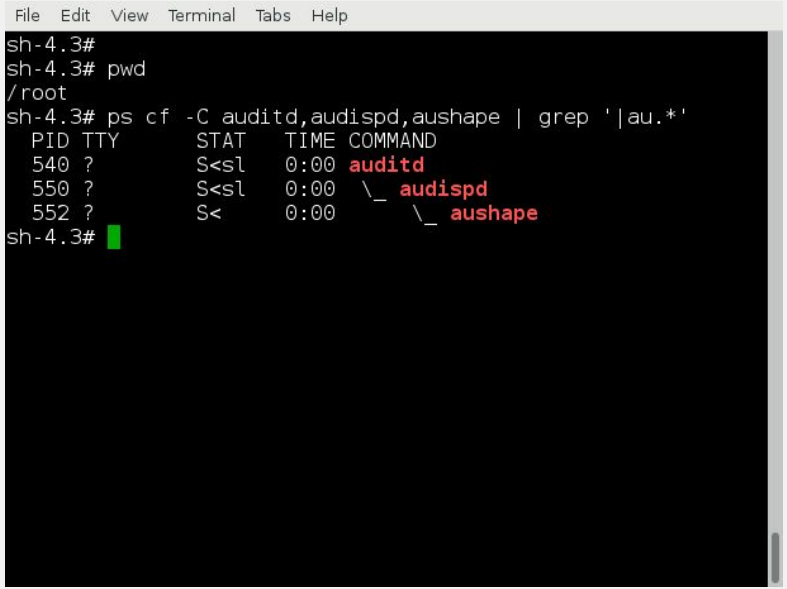
- A shim between the terminal and the shell, started at login
- Converts what passes in between to searchable JSON
- Logs to syslog
- Available on GitHub



MAKE SENSE OF AUDIT LOGS?

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
- Developed with the help from auditd
- Available on GitHub



DELIVER TO ELASTICSEARCH?

Any popular logging service:

RSYSLOG*

Or our coming solution:

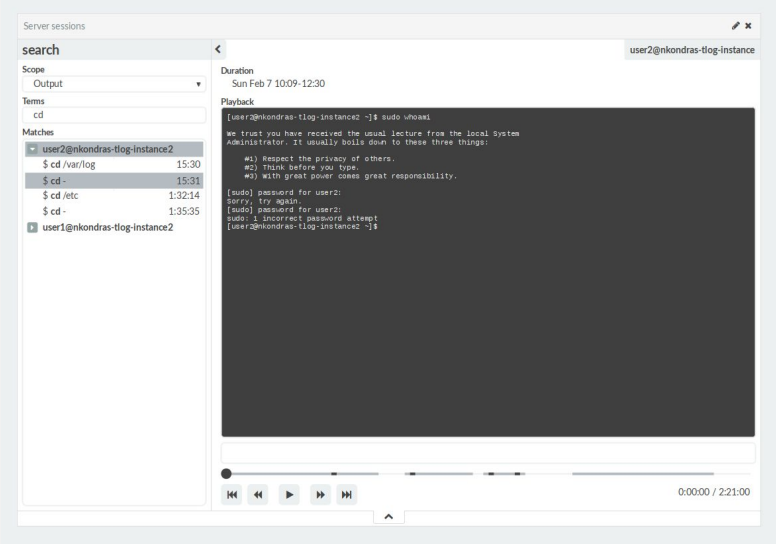
ViaQ

* Distributed by Red Hat now

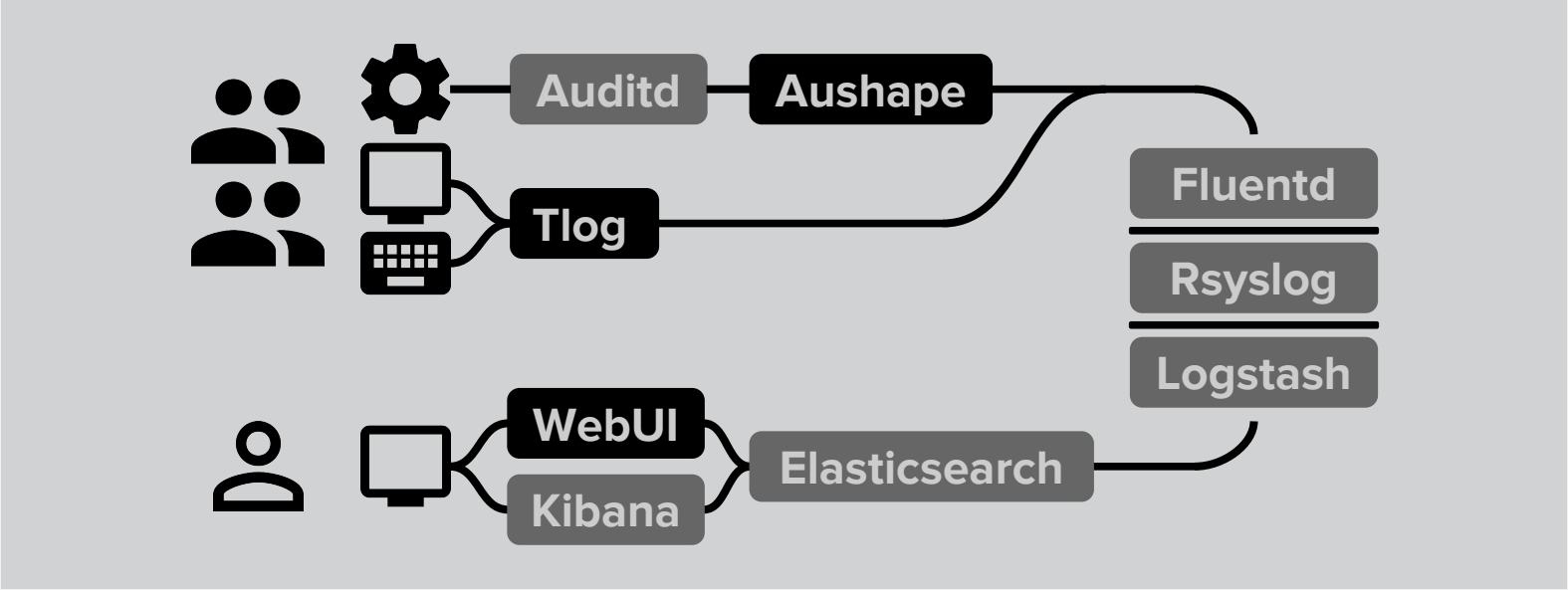
PLAY EVERYTHING BACK?

We’re designing a Web UI

- Playback data from Elasticsearch
- See input, commands executed and files accessed
- Search for input, output, commands and files
- Reuse and integrate



ALL TOGETHER NOW!



LEAN AND MEAN

Why it’s better:

- Reuses log plumbing
  - No separate infrastructure needed
  - Saves resources
  - Reduces maintenance load
- Allows easy correlation with all the other logs
  - Not just an isolated “video of the terminal”
  - Lets you see what was behind the scenes

DEMO!

IN THIS DEMO...

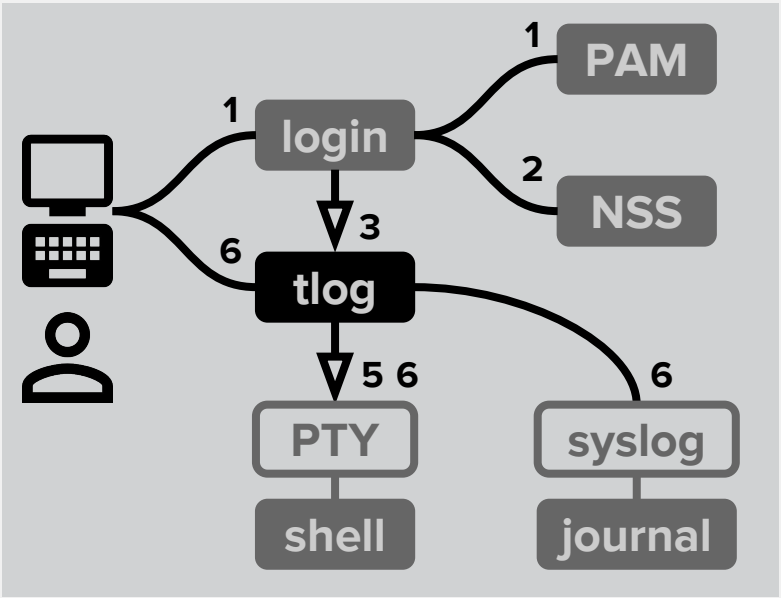
- A recorded user logs in
- Playback of the session is started at the same time
- Some work is done on the terminal
- Terminal I/O and converted audit logs are seen in journal
- Logs in ElasticSearch are displayed by Kibana

HOW?

HOW TLOG WORKS?

Starting a console session:

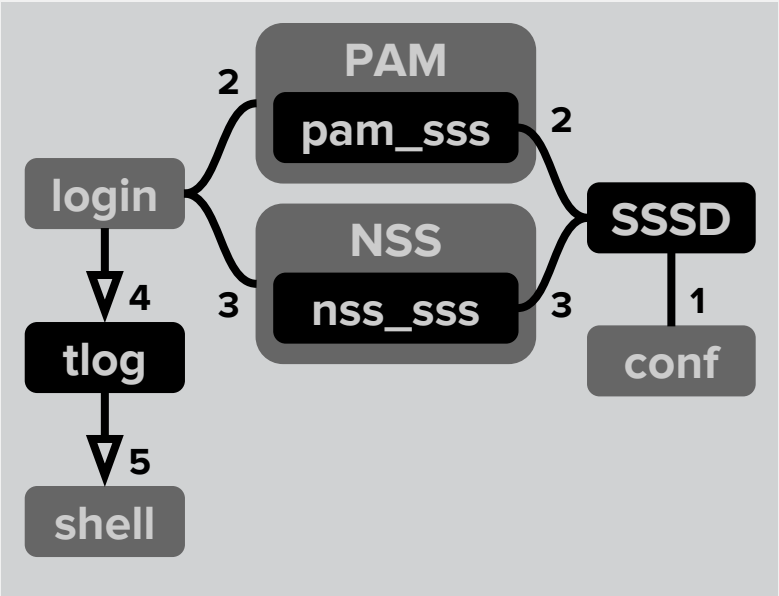
1. User authenticates to **login** via **PAM**
2. **NSS** tells **login**: **tlog** is the shell
3. **login** starts **tlog**
4. Env/config tell **tlog** the actual shell
5. **tlog** starts the actual shell in a **PTY**
6. **tlog** logs everything passing between its **terminal** and the **PTY**, via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



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**: **tlog** is the shell
4. **login** starts **tlog** with **PAM** environment
5. **tlog** starts the actual user **shell** retrieved from environment



CONTROL TLOG WITH FREEIPA

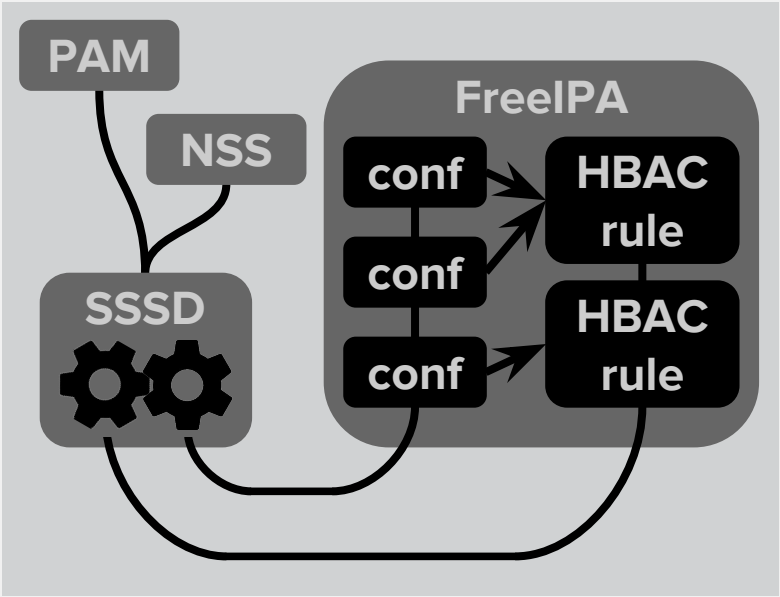
At least that's the plan!

Which users to record on which hosts:

- Recording **configurations** are linked to **HBAC** rules, like SELinux maps

When users login:

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



EXTRA TLOG FEATURES

Also control:

- What to record: input/output/window resizes
- “You are being recorded” notice
- Where to write: syslog(3) or file
- Low latency vs. low overhead

Basic playback on the terminal:

- From Elasticsearch
- From file

TLOG SCHEMA

Optimized for streaming and searching:

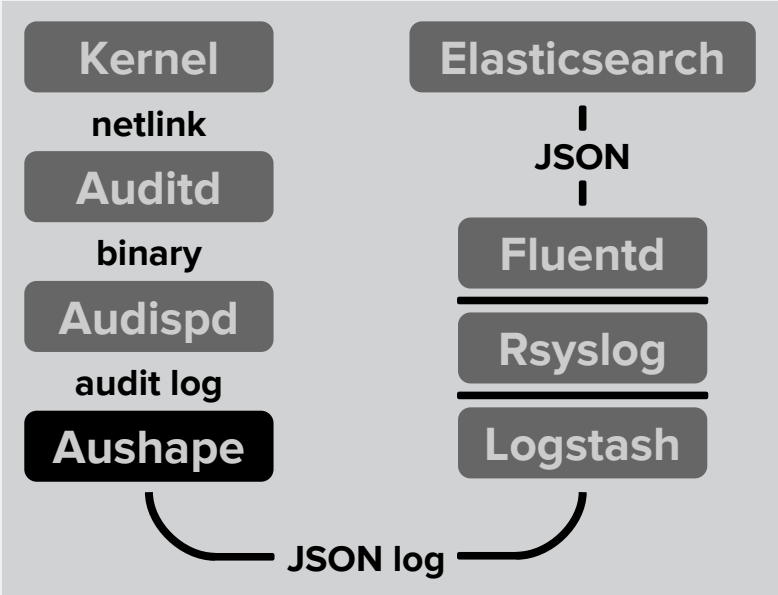
- Chopped into messages for streaming, but can be merged
- Input and output stored separately
- All I/O preserved
- Invalid UTF-8 stored separately
- Timing separate, ms precision
- Window resizes preserved

```
{
    "timestamp" : "...",
    "ver"       : 1,
    "host"      : "tlog-client.example.com",
    "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"   : [ ]
}
```

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 SCHEMAS

Mimicking the audit log, XML and JSON are similar, raw log can be kept

```xml
<log>
  <event serial="number"
         time="timestamp">
    <text>
      <line>log message</line> ...
    </text>
    <data>
      <record>
        <field i="value" r="value"/> ...
      </record> ...
    </data>
  </event> ...
</log>
```

```json
[
  {
    "serial": number,
    "time": "timestamp",
    "text": [
      "log message", ...
    ],
    "data": {
      "record": {
        "field": ["value", "value"], ...
      }, ...
    }
  }, ...
]
```

AUSHAPE EXAMPLES

A heavily-trimmed event

```
<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>
```

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

CHALLENGES!

TLOG CHALLENGES

Lots of fun problems:

- How not to record passwords
  - Detect “echo off” mode, or cooperate with TTY audit
- Detect graphical sessions and don’t record under them
  - Perhaps look at environment variables
- Support charset conversion
  - Use iconv, and keep original text

AUSHAPE CHALLENGES

Some more fun (and not so fun) problems:

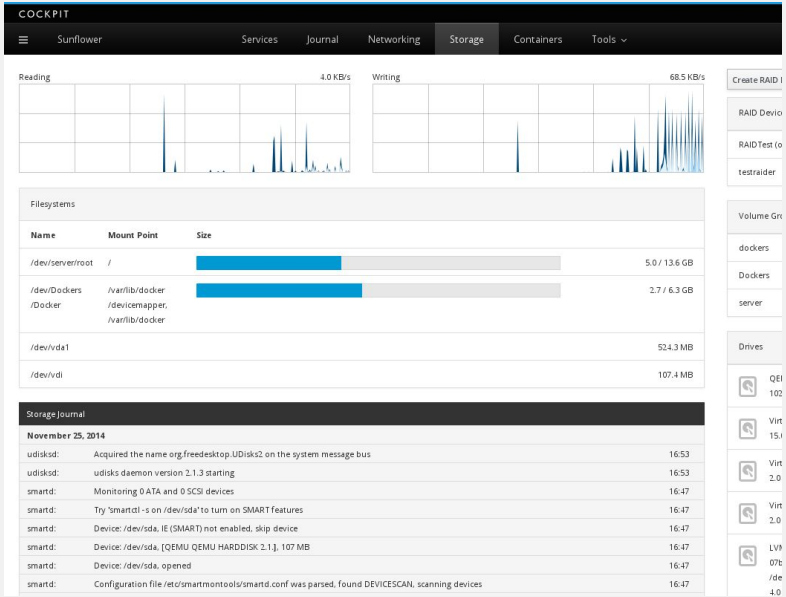
- Audit log is a mess
  - Can’t fix. Track all the cases, use what auditd knows
- Somehow generate coherent schemas
  - Keep schema simple, use auditd record/field dictionaries
- Convert character encodings
  - Iconv, and keep invalid text in base64 or discard

WEB UI CHALLENGES

First step

Integrate into Cockpit Project - a server management Web UI:

- Log locally
- Also forward somewhere else
- Playback from local storage
- Display in a JavaScript terminal



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`
- 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!

QUESTIONS?

redhat.

THANK YOU

User Session Recording Project

http://scribery.github.io/

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos