redhat.

USER SESSION RECORDING IN COCKPIT

Nikolai Kondrashov
Software Engineer
03.02.2018

INTRODUCTION

Nikolai “spbnick” Kondrashov, a software engineer

- Working at Red Hat Common Logging team
- Focusing on the User Session Recording project
- Maintaining FreeRADIUS packages
- Founder and maintainer of the DIGImend project
- Flirting with embedded as a hobby

WHAT ARE WE TRYING TO DO?

User Session Recording Project:

- Record what users see on and type into a terminal
- Record the commands they execute and files they access
- Control centrally what, where and who is recorded
- Store recordings centrally and securely
- Allow searching, correlation, and playback of recordings

WHY ORGANIZATIONS NEED IT?

Government, medical, financial, and others:

- Required by law
- Want to find out who broke the servers and how
- Need to know who stole their data
- Want to trace user problems

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 NO OPEN SOURCE

All we have is:

- 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

OUR APPROACH

- Use logging infrastructure for delivery
  - Centralization solved
  - Easily correlate with other logs
  - Save on resources and maintenance
- Record terminal I/O from userspace with tlog
  - Fast to iterate
  - Easy for users to try
- Use audit logs for the rest
  - Commands executed, files accessed, everything already there

OUR TARGETS

Long- and short-term

- Enterprise-ready long-term
  - Storage in Elasticsearch
  - Central control with FreeIPA and SSSD
  - Playback via a Web UI component
  - Embedded in OpenShift, CloudForms, etc.
- Cockpit short-term
  - Storage in Journal
  - Control via SSSD or manual
  - Configuration and playback in Cockpit Web UI

WHAT IS COCKPIT?

A server management WebUI with a new twist:

- “A Linux session in a browser”
  - Each login creates an actual user session
- Not taking over the system
  - Jump between the WebUI and the command line any time
- Can manage multiple hosts in one session
- Releases every other week
- Extensively tested

DEMO

DEMO

In this demo:

- A recorded user logs in and works on a terminal
- User's terminal I/O is recorded to Journal
- Live recording appears and plays back in Cockpit

HOW IT WORKS

RECORDING SETUP

- Recording process starts as the user’s login shell
- Executes the actual shell under a PTY
- Captures everything passing between TTY and PTY
- Cuts it into pieces on time and size limits
- Encodes to JSON and logs

JSON SCHEMA

For every message

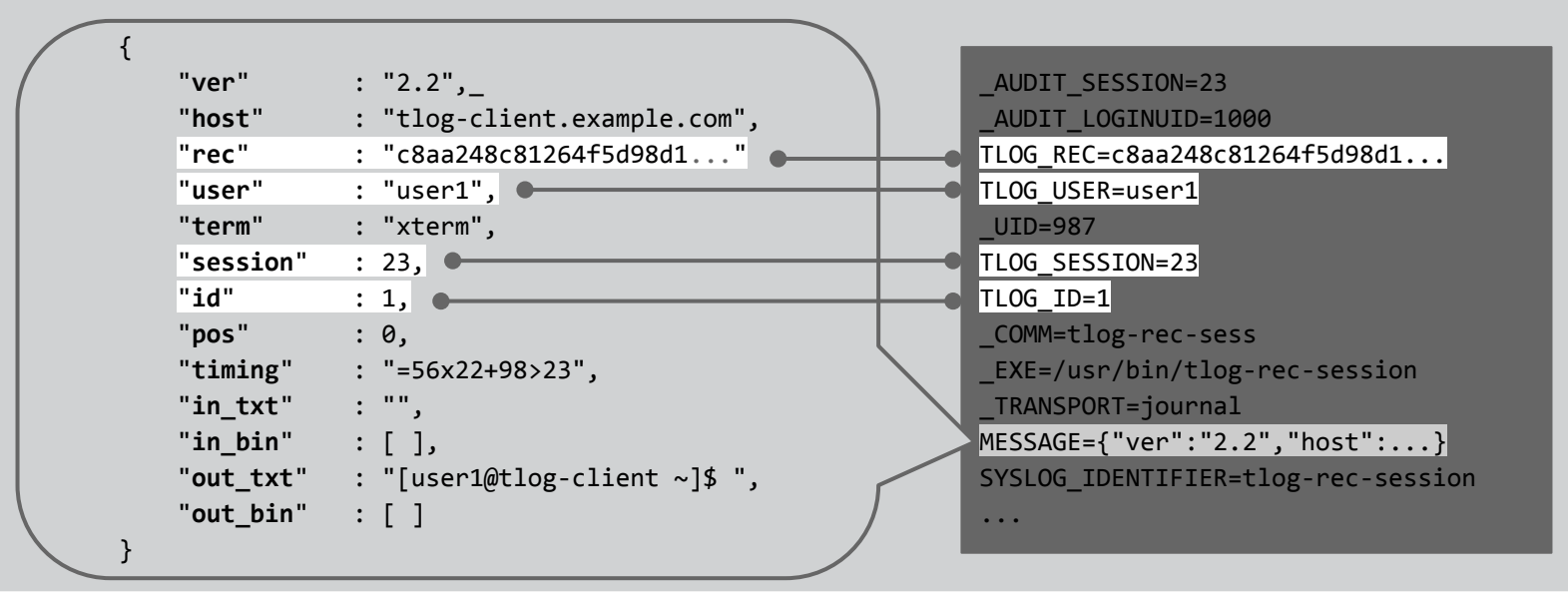
Optimized for streaming and searching:

- Stream cut into time- and size-limited pieces, but can be merged back
- Input and output stored separately
- Timing separate, ms precision
- Window resizes preserved
- All I/O preserved
- Invalid UTF-8 stored separately

```
{
    "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



COCKPIT JOURNAL INTERFACE

Simple but effective

- Host side runs `journalctl --output=json`
- Browser side supplies options and arguments and gets JSON
- Not very efficient, but simple and reliable

LISTING RECORDINGS

- Add a match on the UID of SUID recording process
  - E.g. `_UID=987`
- Add a match on recorded username, if filtering
  - E.g. `TLOG_USER=user1`
- Add `--since` and `--until`, if limiting by time
- Run `journalctl --lines=all --follow`
- Read all returned entries
- Aggregate IDs of unique recordings and their info

PLAYING RECORDINGS

- Add a match on the UID of the SUID recording process
  - E.g. `_UID=987`
- Add a match on recording ID
  - E.g. `TLOG_REC=c8aa248c81264f5d98d1...`
- Run `journalctl --lines=all --follow`
- Read and decode all returned entries in background
- Playback as necessary

CHALLENGES

GETTING AUDIT LOGS

Herding cats

- We need audit log to get more data about the session
  - Session boundaries
  - Commands executed
  - Files accessed
- Journald logs audit events, but it is
  - Unreliable under load (says auditd team)
  - Raw, messy data
- We made a tool to cook audit logs, called `aushape`
  - Parses, augments, normalizes
  - Logs in JSON or XML

ADDING AUDIT LOGS

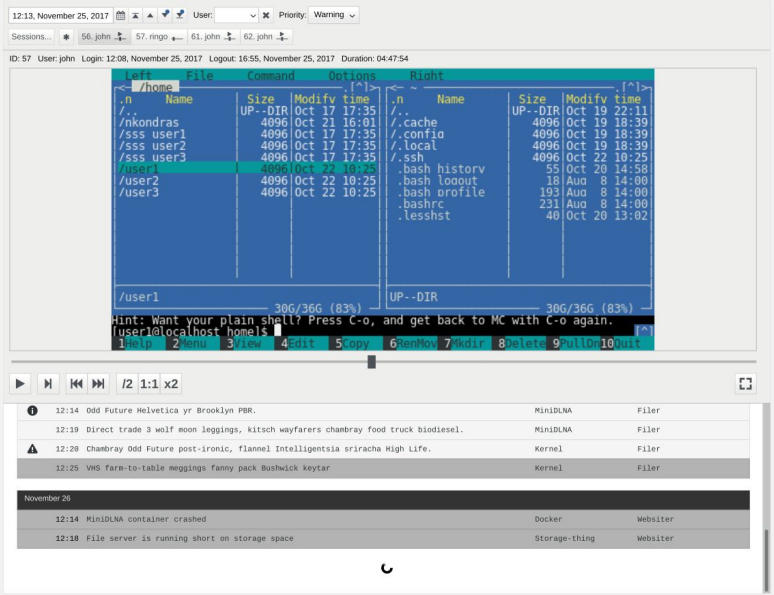
Befriending cats

- Make aushape log data as Journal fields
- or...
- Just get on with Journald audit logs
- still...
- Journal doesn't support partial field matches
  - Searching commands/files is inconvenient
  - Searching I/O is impossible

INTEGRATION WITH LOGS PAGE

New design

- Show sessions active at each point in time, recorded and otherwise
- Show a list of all sessions
- Sync log scrolling with playback
- Support full-screen playback

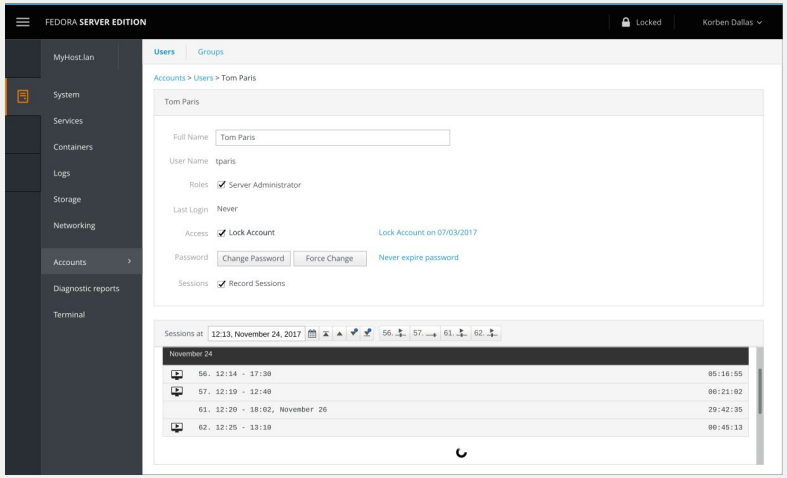


redhat.

INTEGRATION WITH ACCOUNTS PAGE

New design

- Enable/disable recording users by changing their shells
- Enable/disable recording particular users/groups via SSSD
  - Only available for accounts managed by SSSD
- See the list of sessions, recorded and otherwise, for each user/group



TERMINAL TYPES

- Not many types today, but quirks still possible
- Same terminal needed to playback on command line
- Hard to cleanup after playback on command line
- Only a subset is supported by Web UI playback
- Embed a terminal emulator library into recording, long-term
  - Ensures single terminal type to deal with
  - Libvterm seems a good fit

CHARACTER ENCODINGS

- We need UTF-8 to store and search consistently
- Not everyone uses UTF-8
- Converting charset of I/O might lose data
- We'll need to keep both original and converted I/O
  - Original I/O compressed?
  - Converted I/O sanitized?

PLAYBACK SEEKING

- Terminal state accumulates, depends on everything before
- Seeking requires a known state to build upon
- At the moment it’s the start state only
  - Slow for big recordings
- Web UI player has access to terminal emulator internals
  - Build and use terminal state snapshots — “key frames”
- If we embed terminal emulator library into recording
  - Take and log “key frames” on the fly

TRY IT!

TRY IT!

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

- Checkout our scribery branch
- Build and run from source
  - Read HACKING.md
- Install tlog
- 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/

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos