WHY?
THERE IS A DEMAND

Organisations 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
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

tlog: http://scribery.github.io/tlog

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

- What to record: input / output / window resizes
- “You are being recorded” notice
- Low latency vs. low overhead
- Rate limiting
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)`
SCHEMA AND JOURNAL FORMAT

```
{
  "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 / analyse zoo?

- Open Source
- Scalable
- Active community
ELASTICSEARCH + KIBANA

ViaQ: https://github.com/ViaQ

- Normalize logs
- Put them into Elasticsearch
- Dashboards and analytics
- Part of OpenShift, coming to OpenStack & others
DELIVER TO ELASTICSEARCH

Any popular logging service:

- Fluentd
- RSYSLOG*
- Logstash

Or our coming solution:

ViaQ

* Distributed by Red Hat now
3. LOGGING AUDIT EVENTS
AUSHAPE

We made a tool for that too -

**aushape**


- Listens for audit events
- Convert to JSON or XML
- Log to syslog

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

```
<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

- Session **playback**
- See input, output & **audit**
- **Search** for input, output, commands and files
- Reuse and integrate
- **PoC**: [Cockpit](https://cockpit-project.org) plugin, journal storage
COCKPIT SCENARIO

Diagram showing connections between Auditd, Aushape, Tlog, Journal, and WebUI.
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
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

[Diagram showing the integration of FreeIPA, SSSD, Tlog, Rsyslogd, Aushape, Audispd, Auditd, and Ansible Playbooks.]

- FreeIPA
  - HBAC rule
  - conf
- SSSD
  - HBAC rule
  - conf
- Tlog
- Rsyslogd
- Aushape
  - Audispd
  - Auditd
- Ansible Playbooks
CHALLENGES

- Don't record passwords
- Seek, rewind, resize
- Audit log correlation
- Security (circumvention, privesc, ...)

WHAT TLOG ISN'T

- Command whitelisting / blacklisting
- Graphical session recording
- A one-stop shop
TRY IT

- [https://github.com/Scribery/tlog](https://github.com/Scribery/tlog)
- [https://github.com/Scribery/aushape](https://github.com/Scribery/aushape)
- [https://github.com/Scribery/cockpit/tree/SCRIBERY](https://github.com/Scribery/cockpit/tree/SCRIBERY)
- [https://github.com/ViaQ](https://github.com/ViaQ)
- Issues, suggestions, pull requests welcome
THANK YOU

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