# The Linux Audit System (auditd) – its little quirks and how to handle them

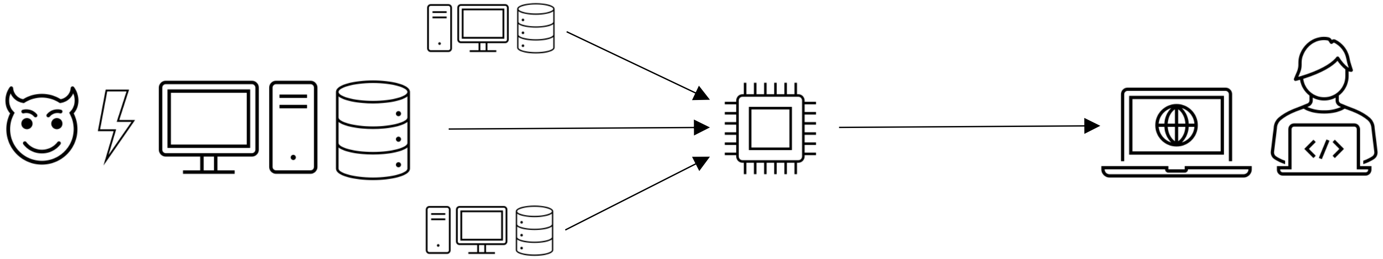
Felix Kosterhon

25.08.2021

# About myself

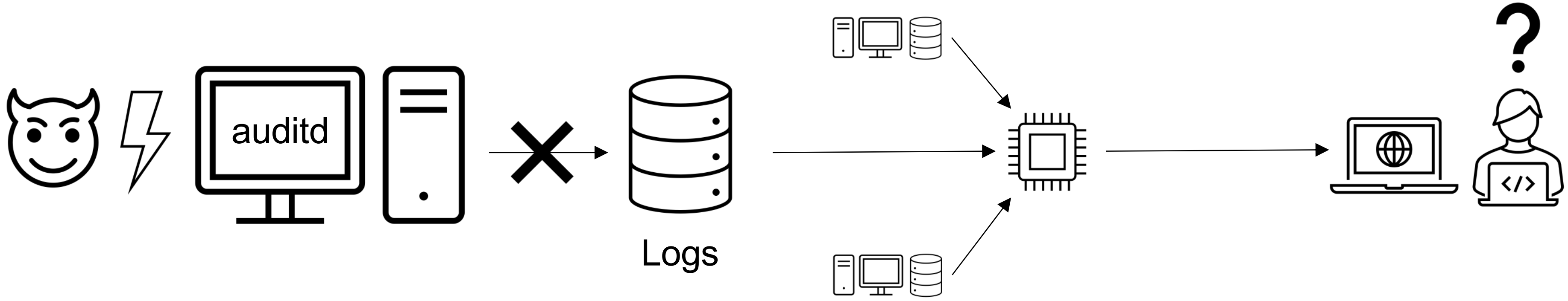
- Felix Kosterhon
- Graduated at TU Darmstadt in IT Security
- Cyber Defense Consultant at SECUINFRA GmbH
  - Protect companies from attacks using SIEM systems
  - SIEM = **S**ecurity **I**nformation and **E**vent **M**anagement
  - Centralize, analyze and correlate logs to identify attacks
- Company-internal responsibility for auditd
  - CVE 2020-35501 identified in Nov 2020

SECUINFRA
Cyber Defense. Made in Germany.



# What is auditd & why do we care?

- The Linux Audit Framework (*auditd*) enables us to monitor user-defined events
- Windows: Sysmon → Linux: Auditd
- Widely used by companies in their Security Operation Center (SOC)
- Reliable logs build the foundation for monitoring systems such as SIEM systems



# Outline

- auditd 101
- File watch implementation
- Monitoring of files and directories
- Logging actions always and never
- Conclusion

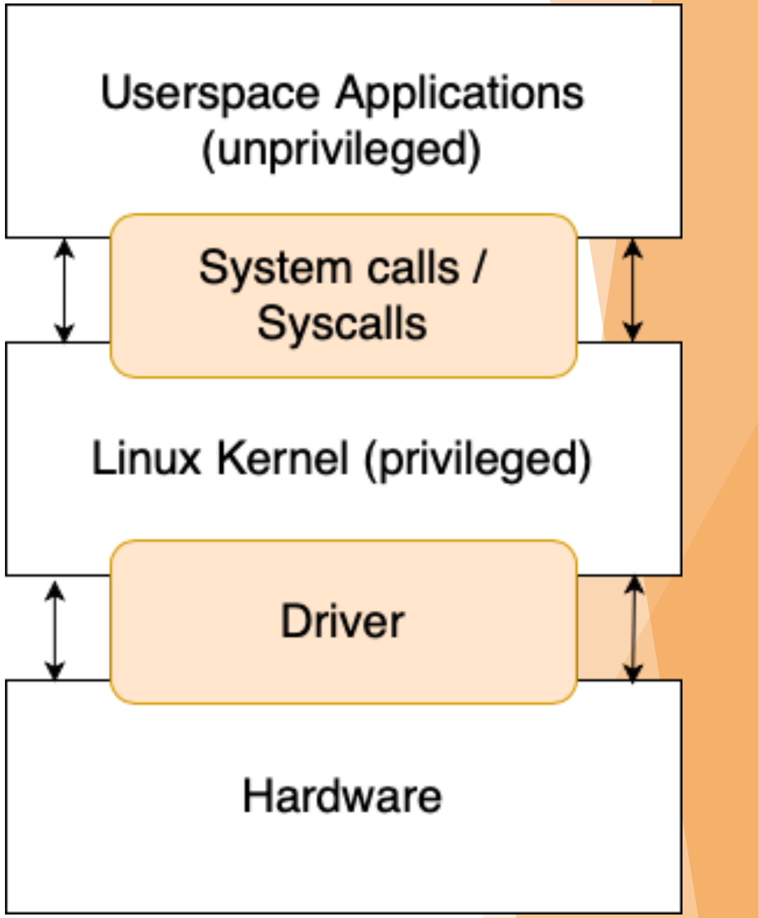
# Outline

- **auditd 101**
- File watch implementation
- Monitoring of files and directories
- Logging actions always and never
- Conclusion

# auditd 101

## Linux – System calls

- Open source, many distributions
- Common in server environments
- Operating system is split into user- and kernelspace
- Syscalls used as interface between userspace and the kernel
  - File handling: Read, write, open, close, stat, …
  - Process handling: fork, execve, kill, …
  - Network: socket, connect, listen, …
  - …


Userspace Applications (unprivileged)
System calls / Syscalls
Linux Kernel (privileged)
Driver
Hardware

# auditd 101

## Why do we need additional logging?

- Syslog used as native logging system
- Only offers information to pre-defined events
- Logs often provide only limited context

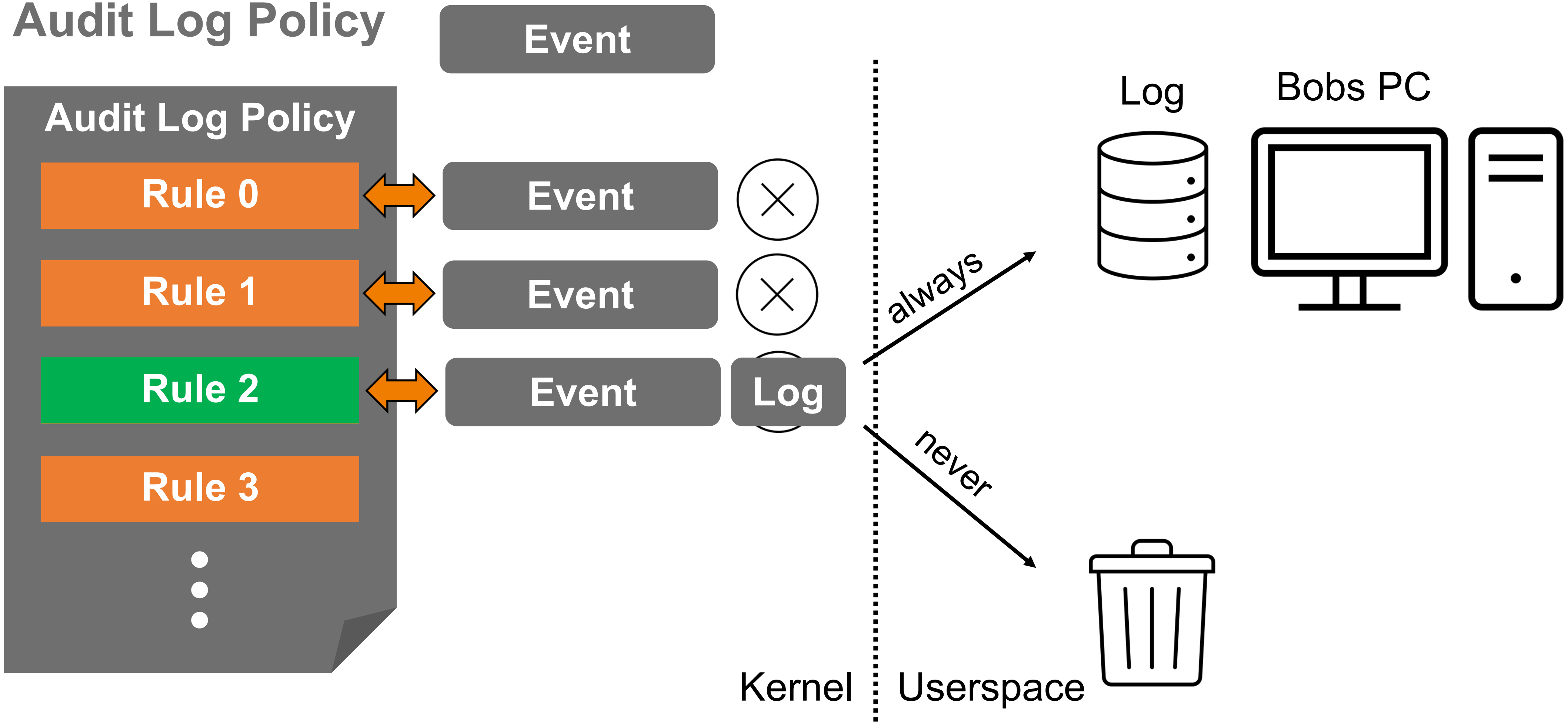
# auditd 101

## Introduction to auditd

- Developed by *RedHat, Inc.*,
- Shipped with most distributions
- Many events included by default
- Allows user-defined monitoring of system & user activities based on rules
- Rule matching is performed directly in the kernel
- Import of single rules or multiple rules in so-called *audit log policies*
- The rule order is First Match only!

# auditd 101

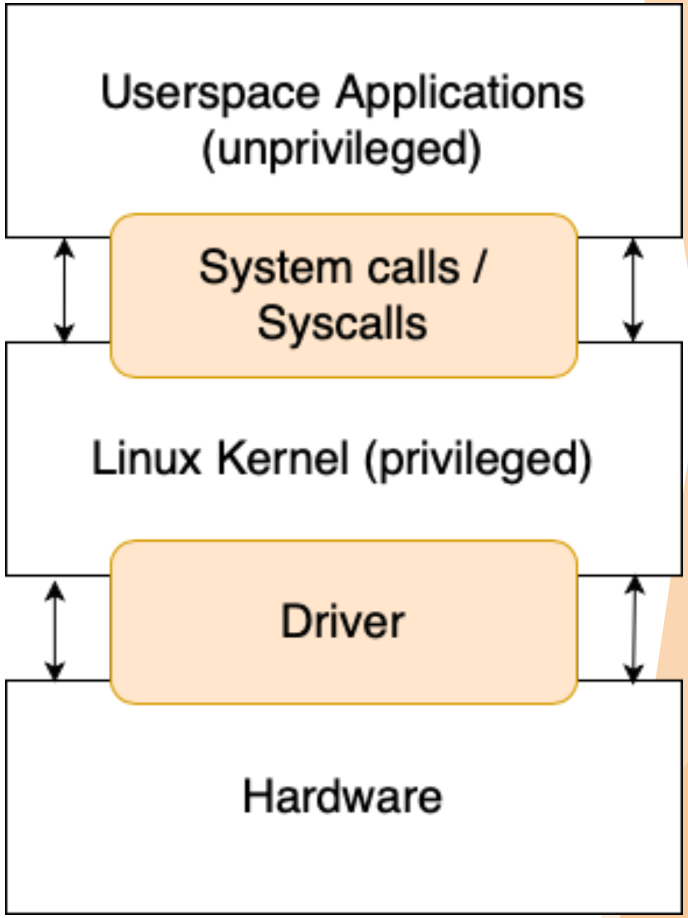
## Audit Log Policy



# auditd 101

## Rules

- Three types of auditd rules
  - **Control**: Configuration
  - **(File) Watches**: File monitoring
  - **Syscall**: Monitoring of specific system calls



Userspace Applications (unprivileged)

System calls / Syscalls

Linux Kernel (privileged)

Driver

Hardware

# auditd 101

## (File) Watches

- Monitoring of files (non-recursive) and directories (recursive)
- Example: *-w /etc/shadow -p wa -k "ShadowFileModified“*
  - -w adds a file watch
  - Permissions (p):
    - Write access (w), read access (r), execution (x), modifications of file attributes (a)
  - Rule identification using a key (-k)

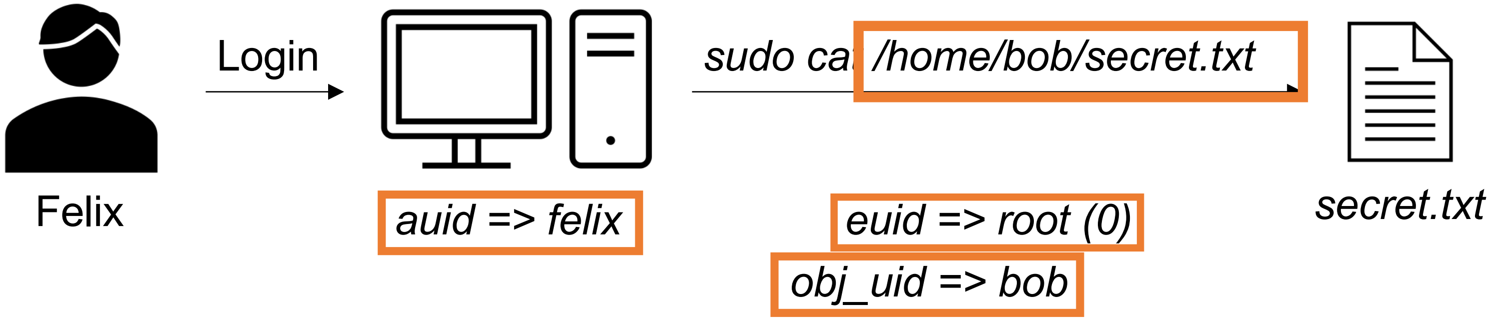
# auditd 101

## Syscall rules

Syscall examples:

- *-a exit,always -F dir=/home -F euid=0 -C auid!=obj_uid -k sudoAbuse*
  - File accessed in the /home directory
  - File accessed in the context of user ID 0 (root)
  - The object id (owner of the file) is not the same as the logged-in user

    => User accessed the file of another user using sudo



# From an attacker's perspective

- Auditd is widely used by companies to detect attackers
- How can we bypass the monitoring?
  - Syscall monitoring is done in the linux kernel ⊗
  - File watches seem to be more promising ✓
- How to start our research?
  - Documentation ⊗
  - Source code ⊗
  - Experimental approach ✓

# Outline

- **auditd 101**
- File watch implementation
- Monitoring of files and directories
- Logging actions always and never
- Conclusion

# Outline

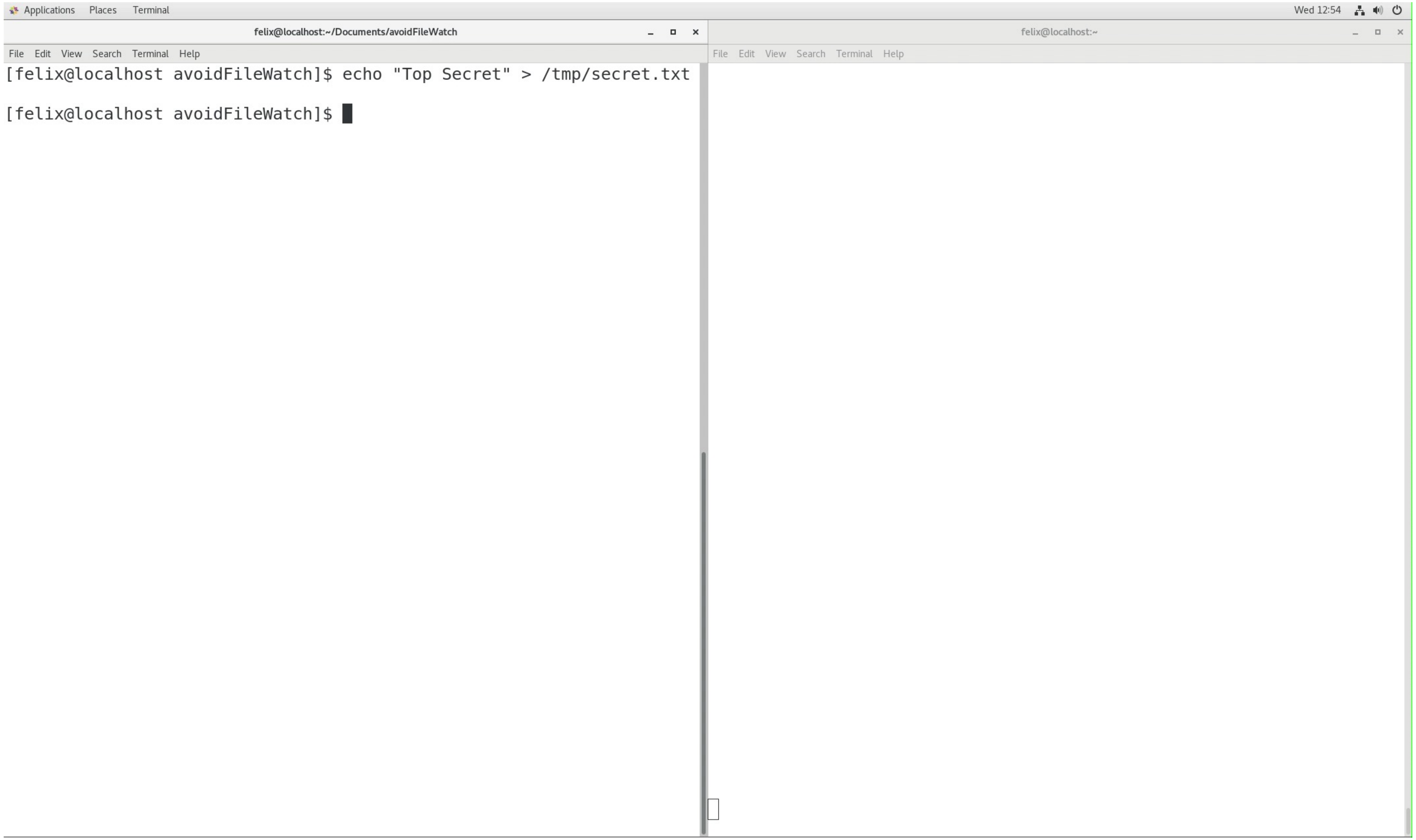
- auditd 101
- **File watch implementation**
- Monitoring of files and directories
- Logging actions always and never
- Conclusion

# File watch implementation

## Experimental approach

- Man-Pages:
  - “[…] read & write syscall are omittet […] would overwhelm the logs.”
  - “[…] open flags are looked at to see what permission was requested.”
- Maybe not all ways to open a file are monitored?
  - Multiple open syscalls available: *open, openat, open_by_handle_at*
- We can bypass file watch monitoring using *open_by_handle_at*
  - *CVE-2020-35501*
- Limitation
  - The user needs elevated privileges to execute the syscall (*CAP_DAC_READ_SEARCH*)

# CVE-2020-35501

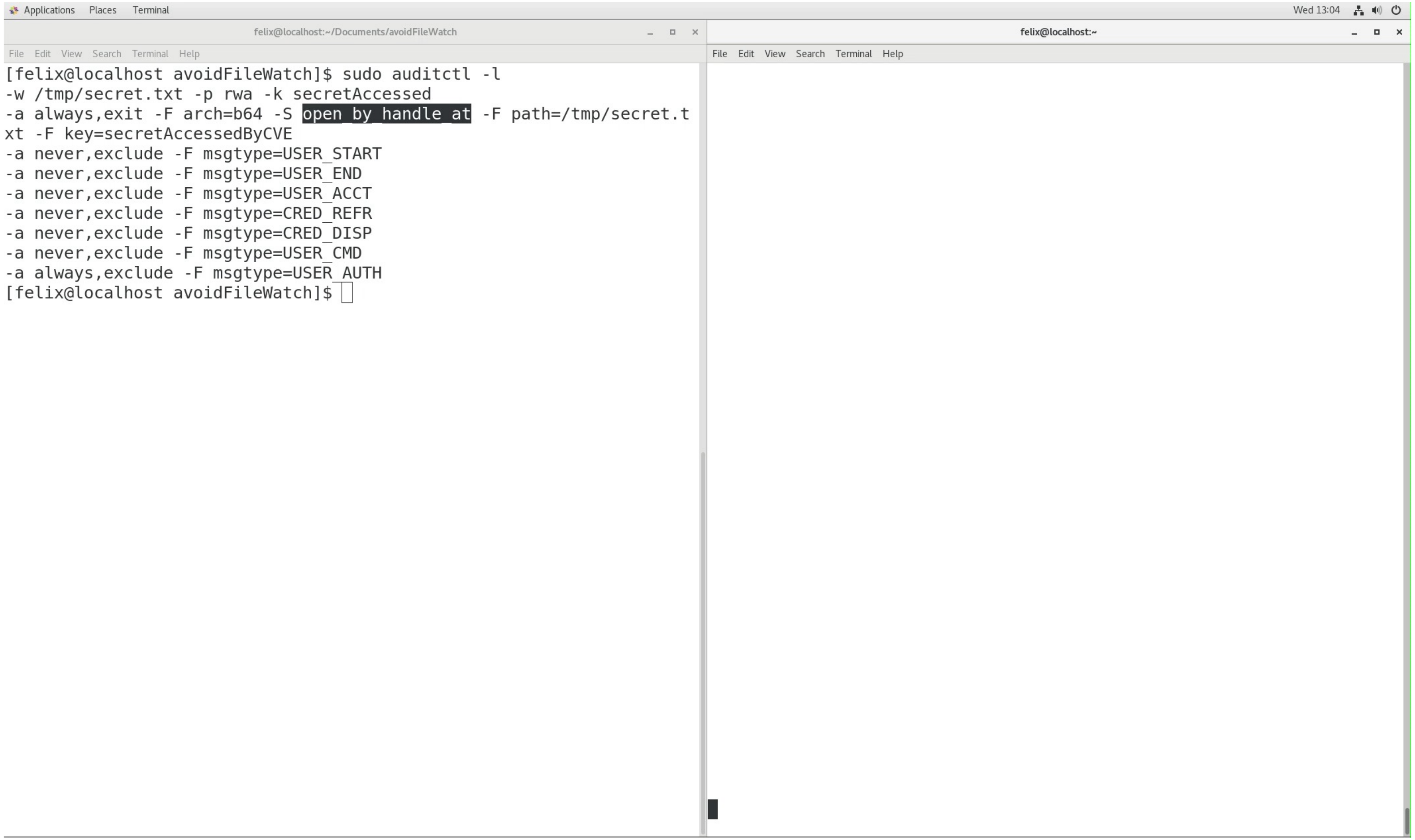


Applications
Places
Terminal
Wed 12:54
felix@localhost:~/Documents/avoidFileWatch
File Edit View Search Terminal Help
felix@localhost:~
File Edit View Search Terminal Help

```
[felix@localhost avoidFileWatch]$ echo "Top Secret" > /tmp/secret.txt

[felix@localhost avoidFileWatch]$ 
```

# CVE-2020-35501

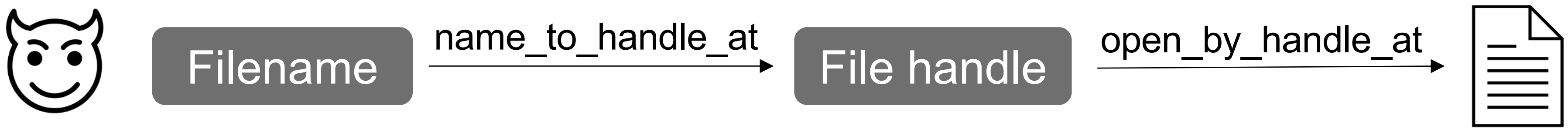

Applications Places Terminal
Wed 13:04
felix@localhost:~/Documents/avoidFileWatch
File Edit View Search Terminal Help
felix@localhost:~
File Edit View Search Terminal Help

```
[felix@localhost avoidFileWatch]$ sudo auditctl -l
-w /tmp/secret.txt -p rwa -k secretAccessed
-a always,exit -F arch=b64 -S open_by_handle_at -F path=/tmp/secret.t
xt -F key=secretAccessedByCVE
-a never,exclude -F msgtype=USER_START
-a never,exclude -F msgtype=USER_END
-a never,exclude -F msgtype=USER_ACCT
-a never,exclude -F msgtype=CRED_REFR
-a never,exclude -F msgtype=CRED_DISP
-a never,exclude -F msgtype=USER_CMD
-a always,exclude -F msgtype=USER_AUTH
[felix@localhost avoidFileWatch]$ 
```

# CVE-2020-35501

## Mitigation

- **Monitor the usage of open_by_handle_at syscalls**
  *-a exit,always –F arch=b64 –S open_by_handle_at -F dir=/etc/ -k […]*
- **Challenge**
  - The filename is not directly passed to the *open_by_handle_at* syscall
  - Instead, a handle is passed as argument
  - The log does not contain the file name, only the inode
- **Monitor additionally the usage of name_to_handle_at syscalls**:
  *-a exit,always –F arch=b64 –S name_to_handle_at –F dir=/etc/ -k […]*



Filename

name_to_handle_at

File handle

open_by_handle_at

# From an attacker's perspective

- We can bypass file watches entirely if we have elevated privileges
- What could we achieve with user privileges?
- All interactions with auditd require elevated privileges
- What about the monitored files?

# Outline

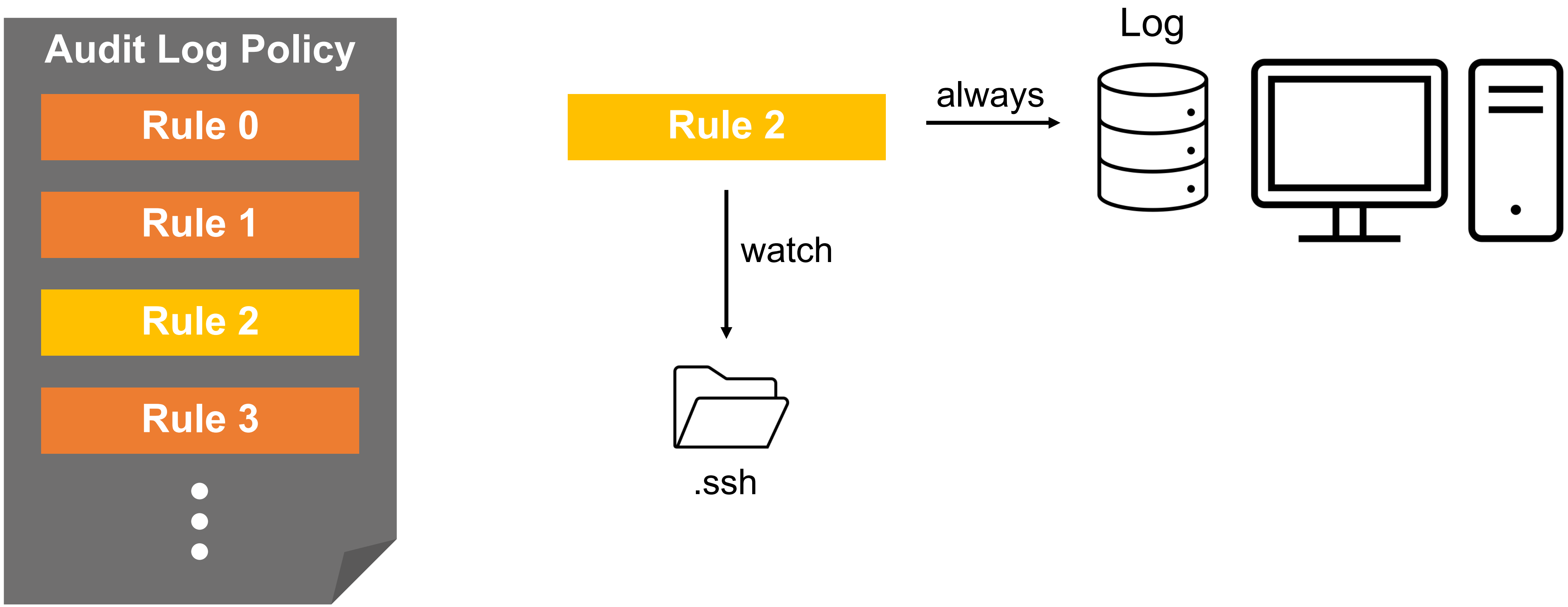
- auditd 101
- **File watch implementation**
- Monitoring of files and directories
- Logging actions always and never
- Conclusion

# Outline

- auditd 101
- File watch implementation
- **Monitoring of files and directories**
- Logging actions always and never
- Conclusion

# Monitoring of files and directories

## Directory monitoring



# Monitoring of fil es

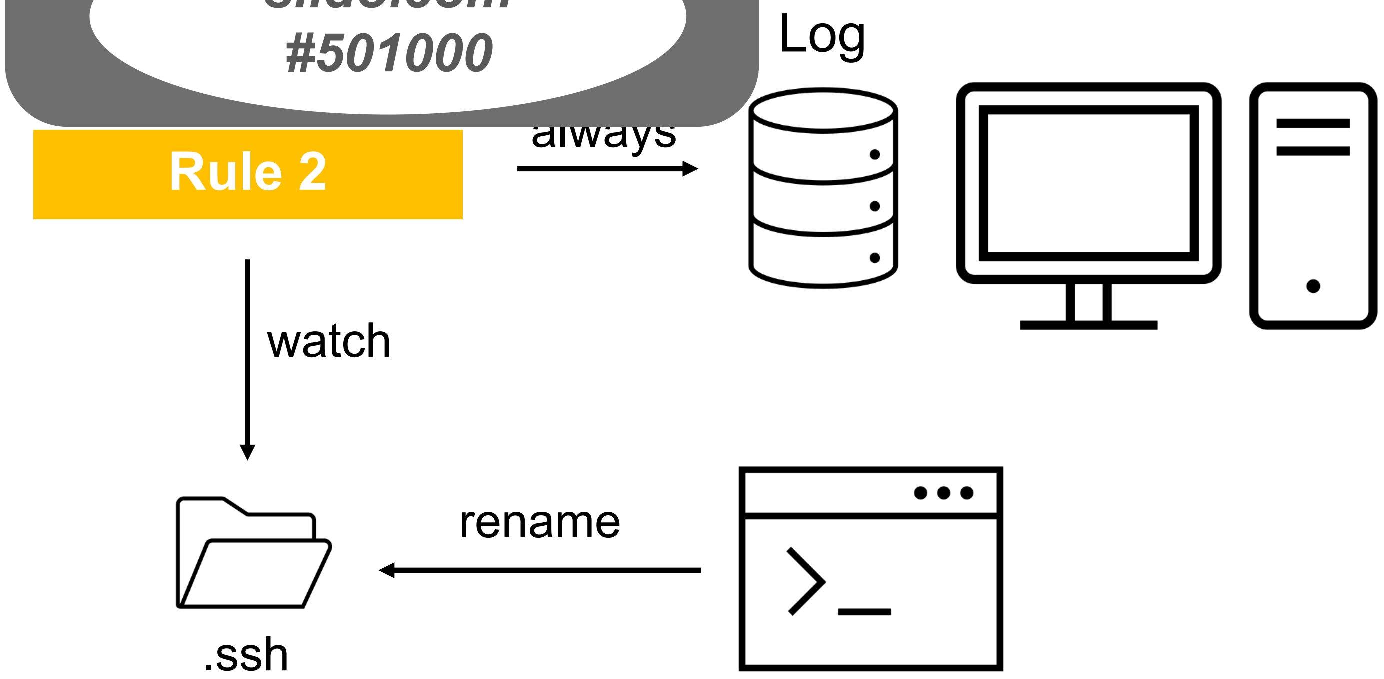
Is the directory still being monitored?

*slido.com*
*#501000*

## Directory monitoring

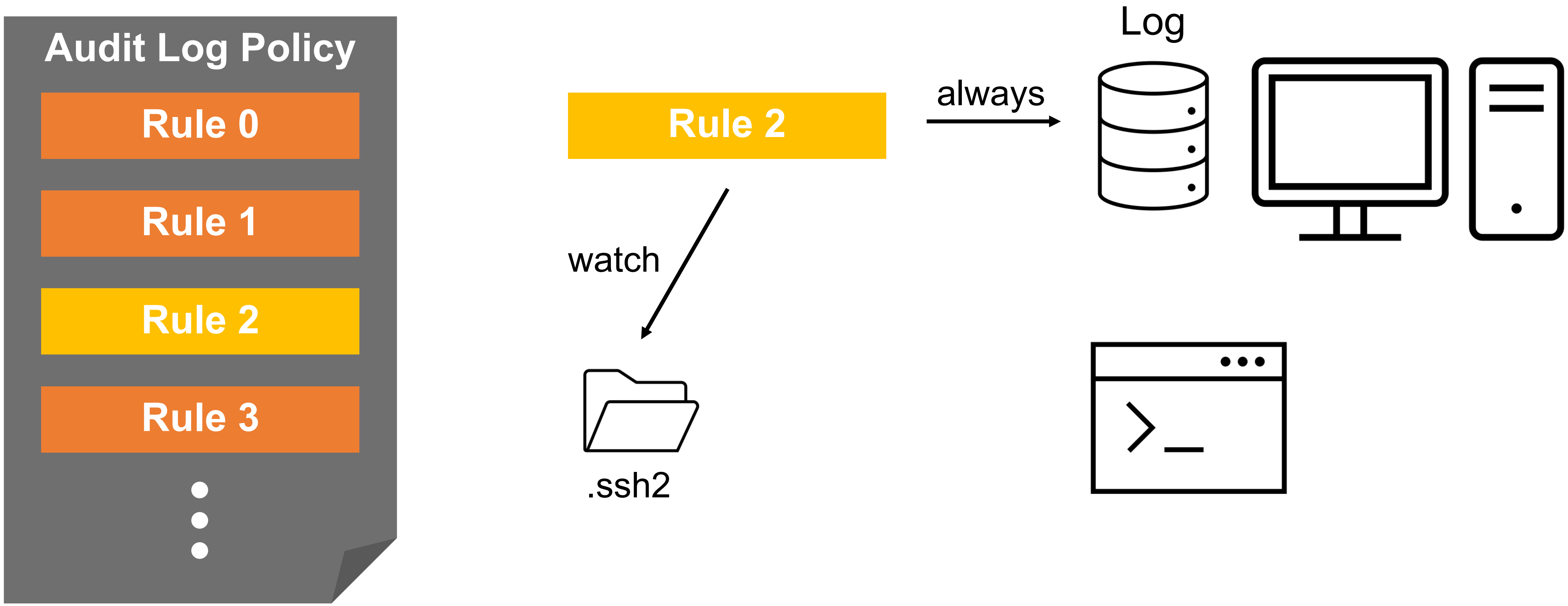
**Audit Log Policy**

- Rule 0
- Rule 1
- Rule 2
- Rule 3
- ⋮



# Monitoring of files and directories

## Directory monitoring



Audit Log Policy

Rule 0

Rule 1

Rule 2

Rule 3

Rule 2

always

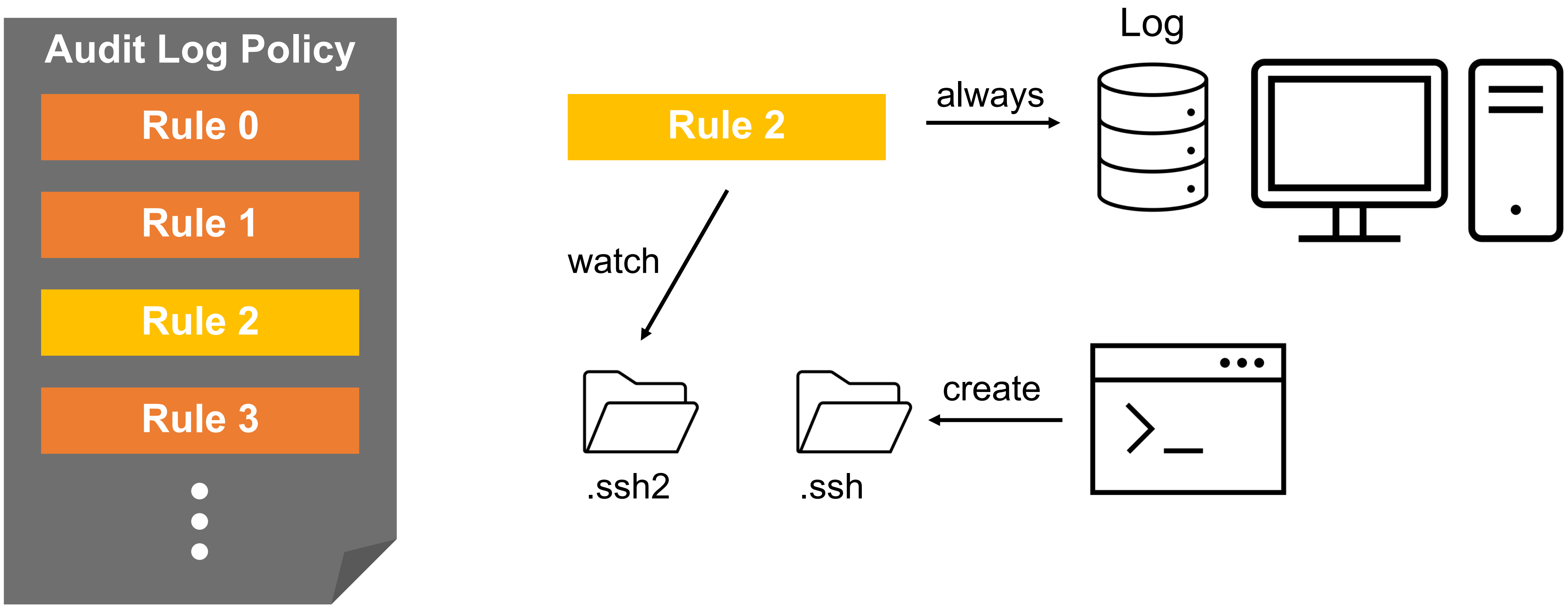
Log

watch

.ssh2

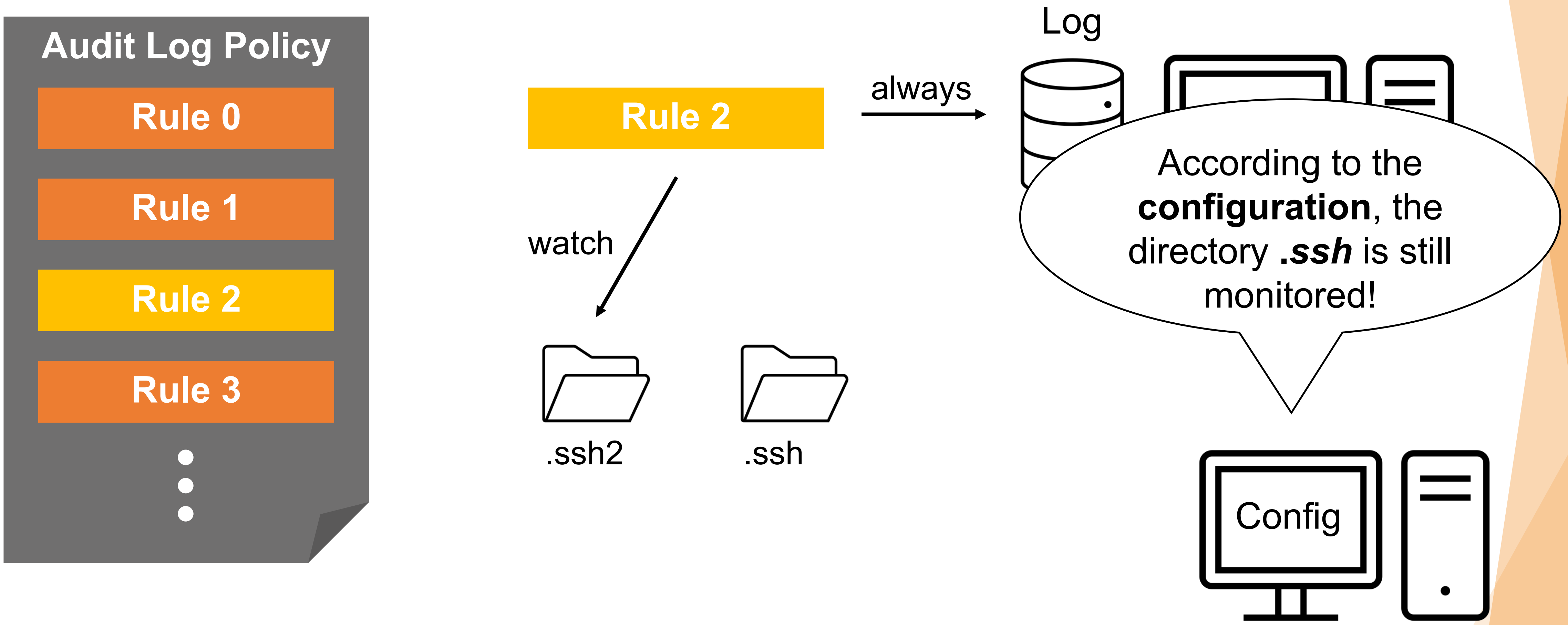
# Monitoring of files and directories

## Directory monitoring



# Monitoring of files and directories

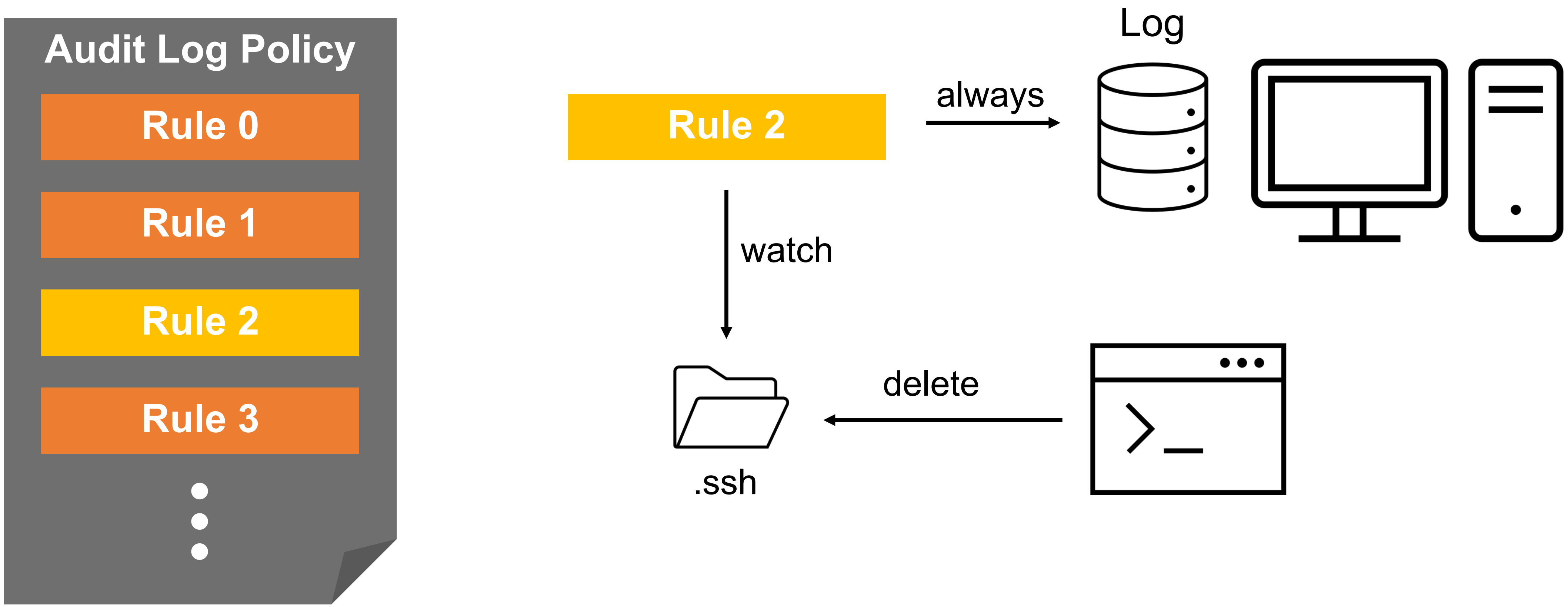
## Directory monitoring



According to the **configuration**, the directory ***.ssh*** is still monitored!

# Monitoring of files and directories

## Directory monitoring



Audit Log Policy

Rule 0

Rule 1

Rule 2

Rule 3

Rule 2

always

Log

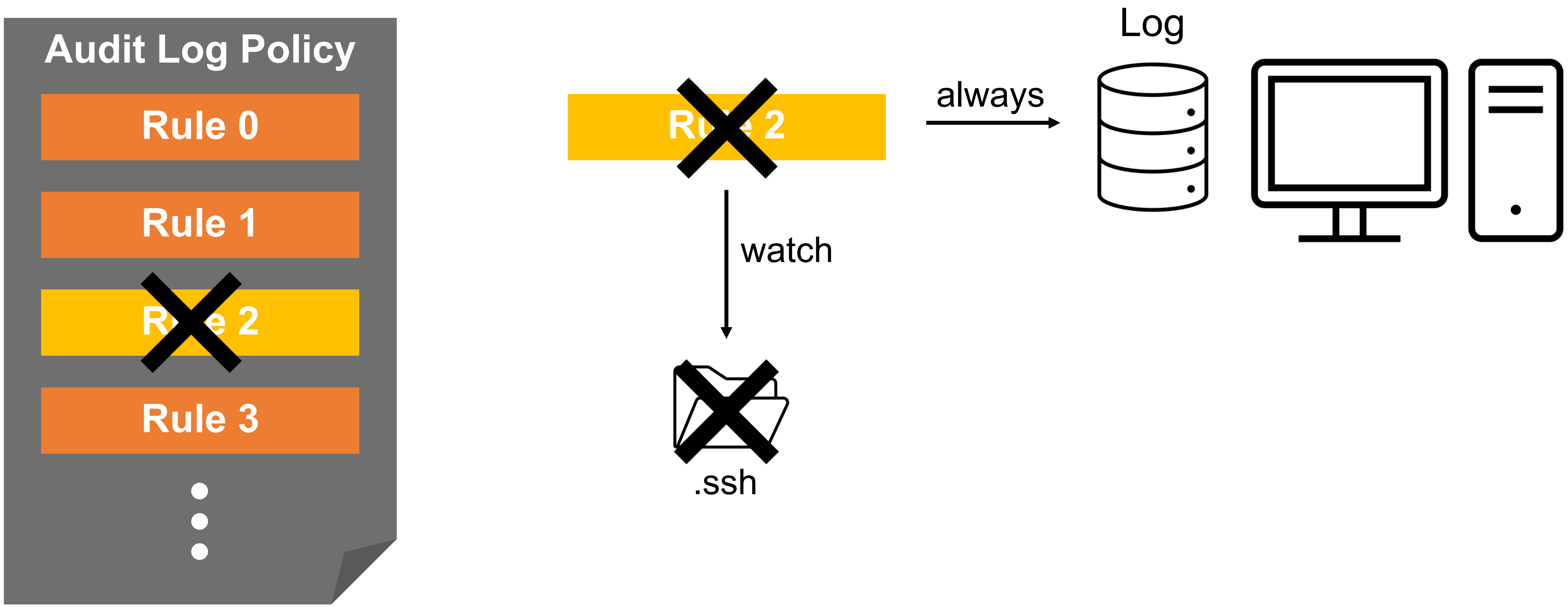
watch

delete

.ssh

# Monitoring of files and directories

## Directory monitoring



# Monitoring of files and directories

## Directory monitoring

Audit Log Policy

Rule 0

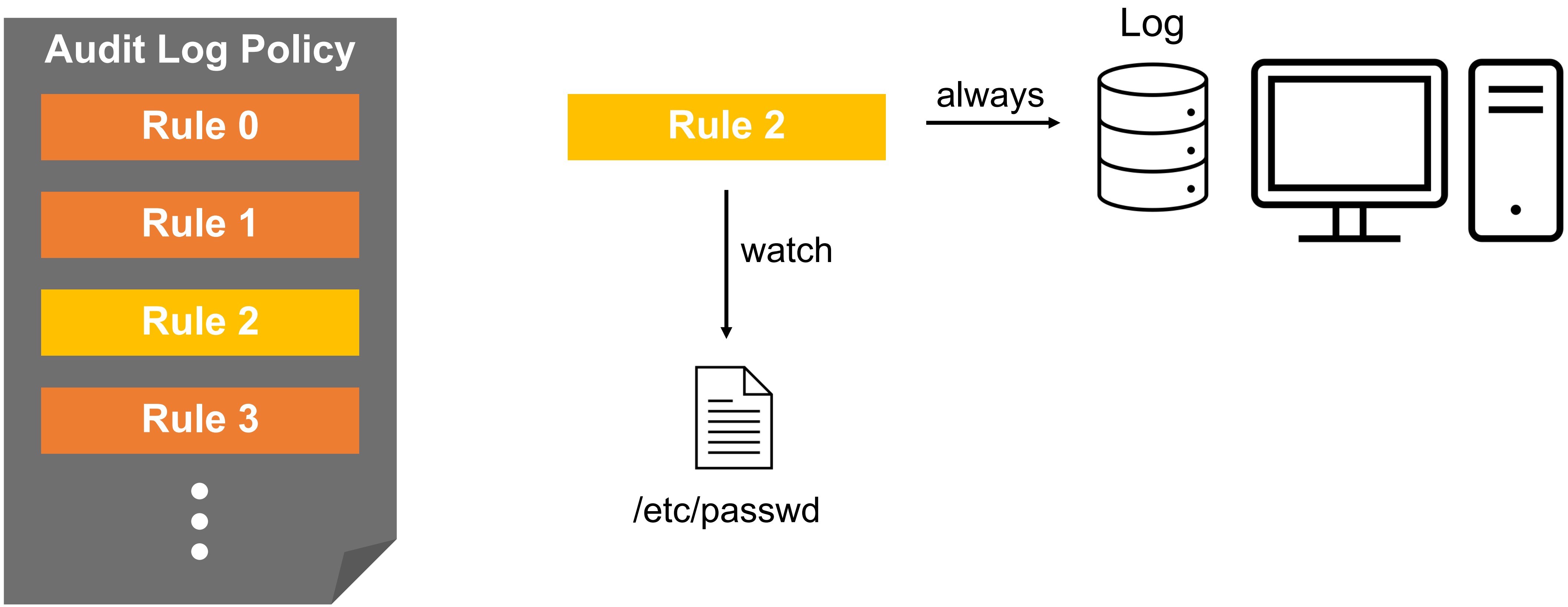
Rule 1

Rule 3

Log

# Monitoring of files and directories

## File monitoring



Audit Log Policy

Rule 0

Rule 1

Rule 2

Rule 3

Rule 2

always

Log

watch

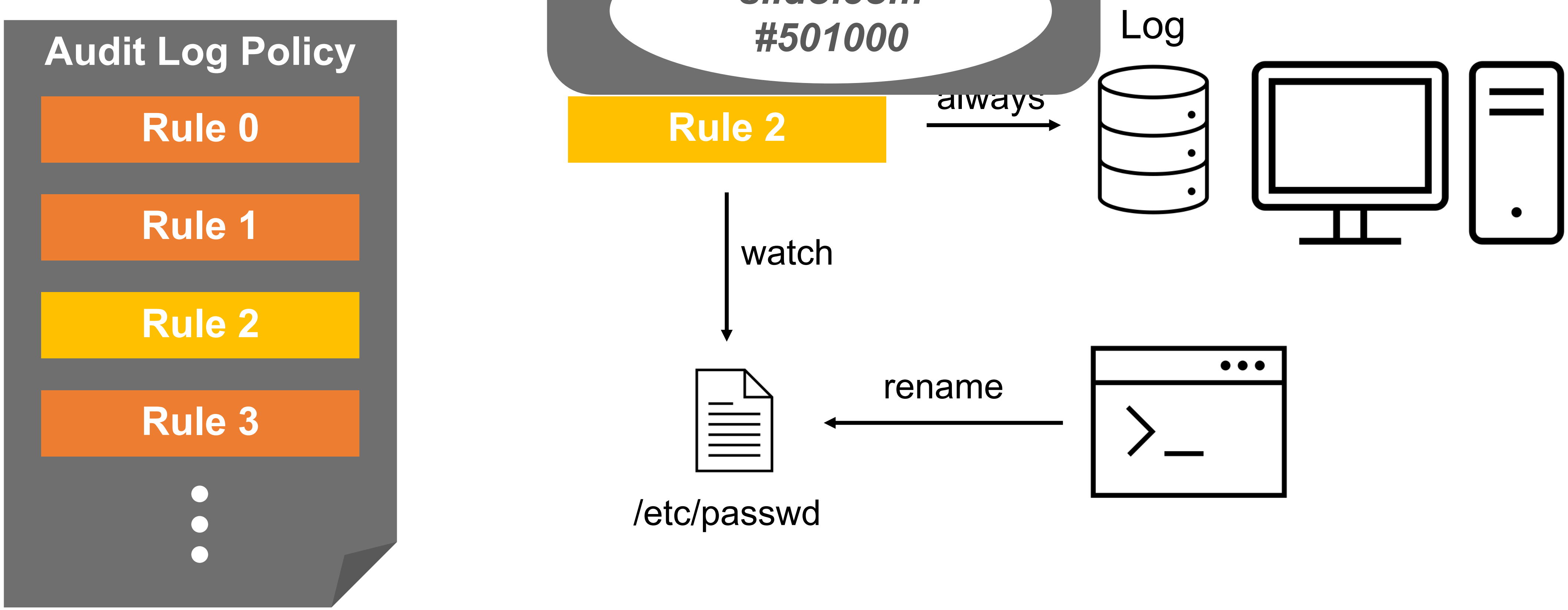
/etc/passwd

# Monitoring of fil es

## File monitoring

Is the file still being monitored?

*slido.com*
***#501000***



Audit Log Policy

Rule 0

Rule 1

Rule 2

Rule 3

⋮

Rule 2

always

Log

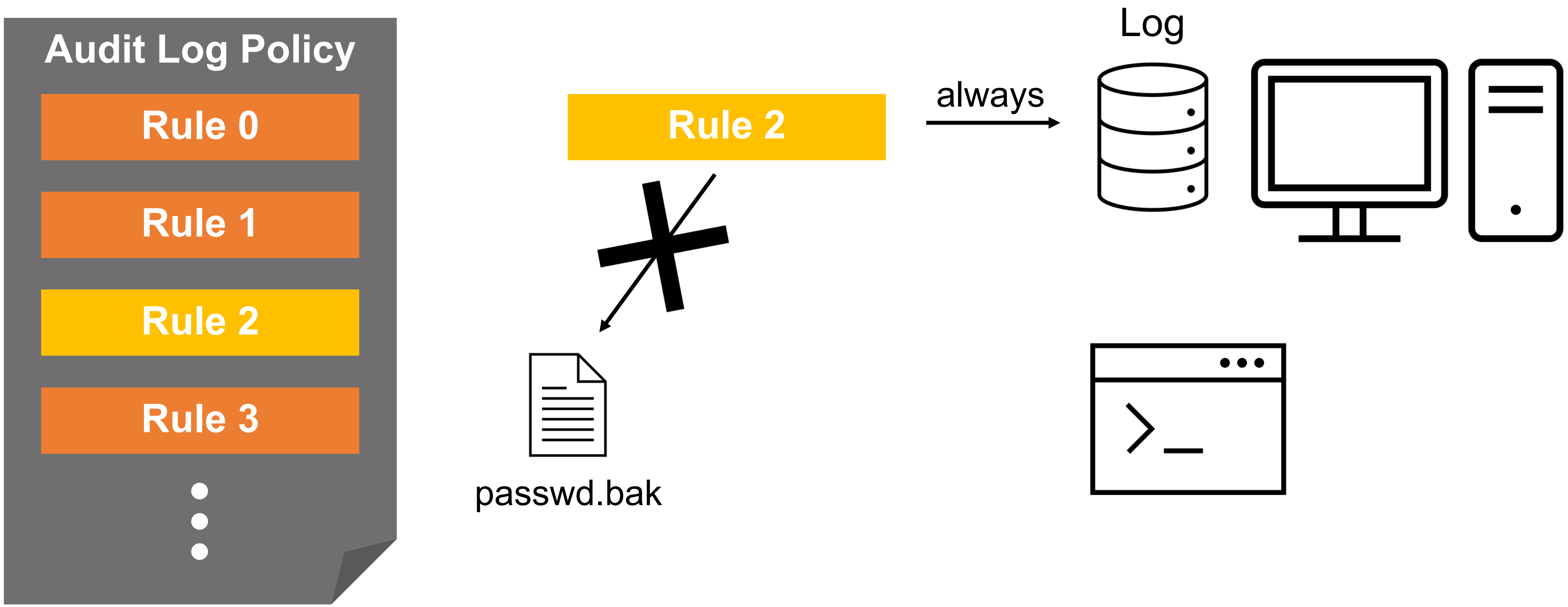
watch

rename

/etc/passwd

# Monitoring of files and directories

## File monitoring

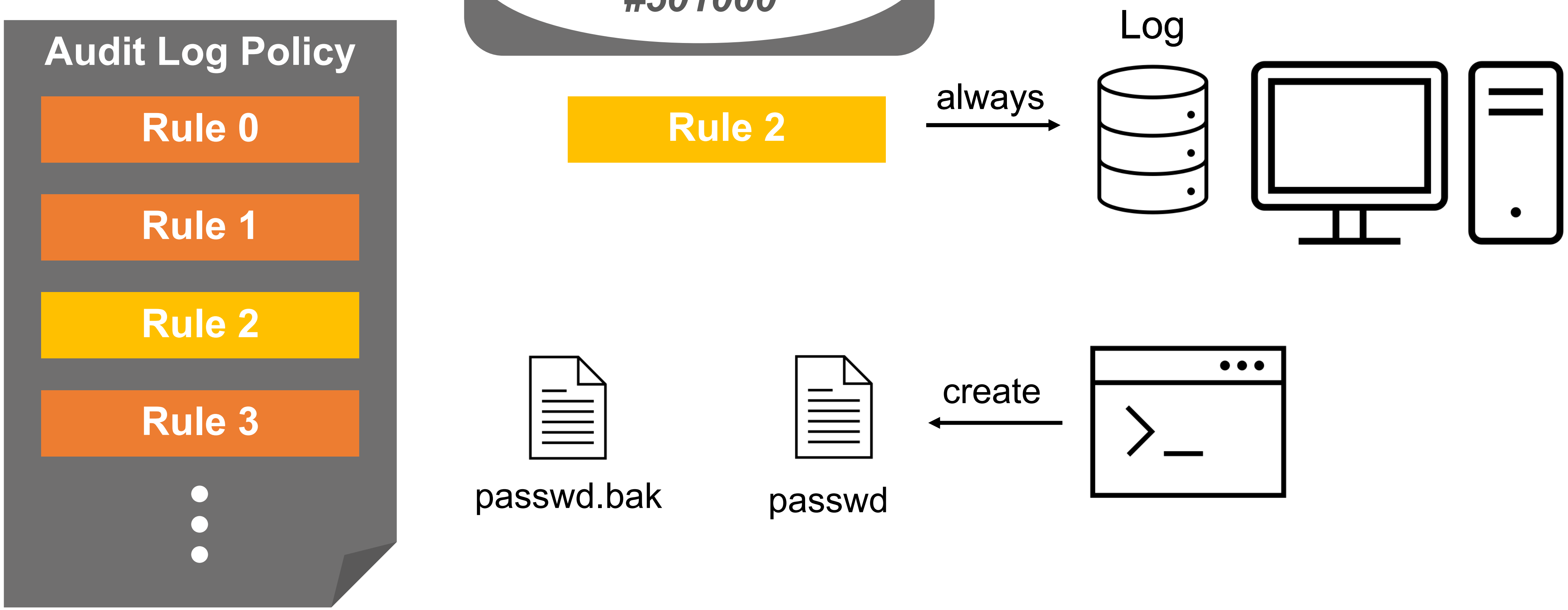


# Monitoring of ctories

**Is the file monitored?**

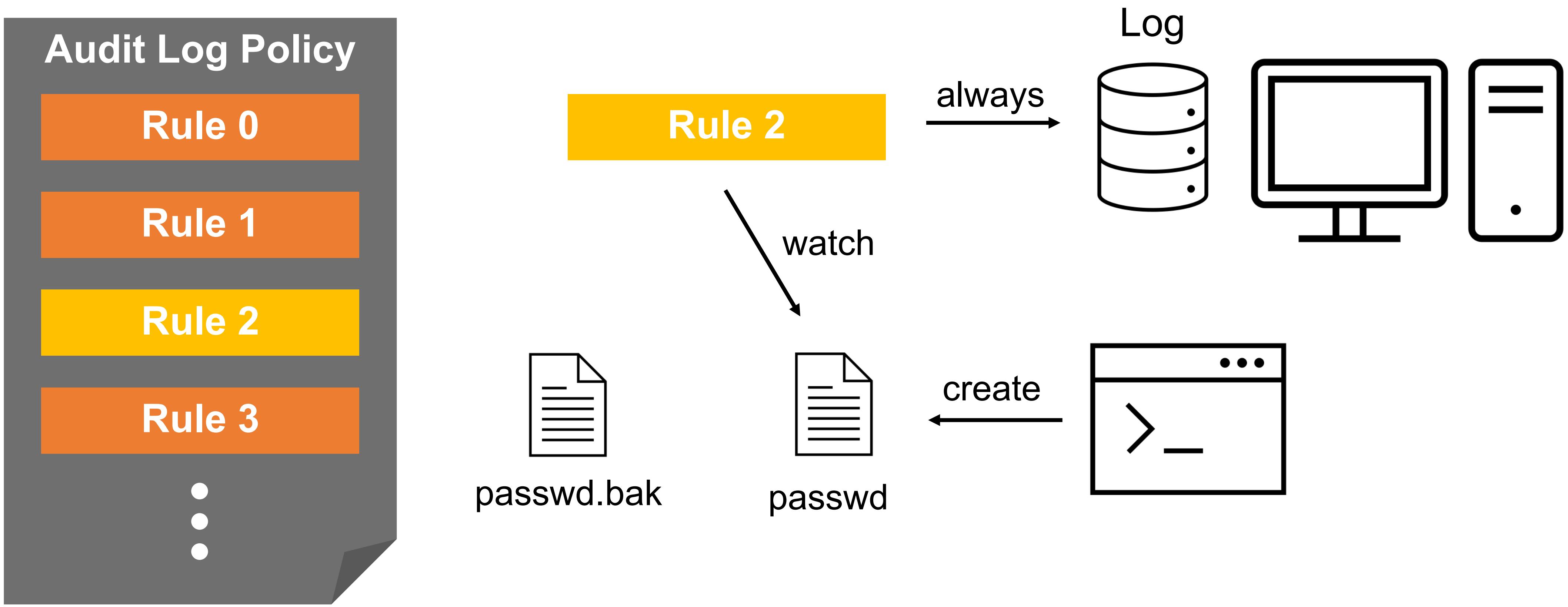
*slido.com*
*#501000*

## File monitoring



# Monitoring of files and directories

## File monitoring



# Monitoring of files and directories

## Everything is a file?

- Why is the handling of files and directories different?
  - Guess: Recursive vs. non-recursive monitoring ✓
- How can we test it?
  - File Watches: Auditd chooses automatically
  - Syscall Rules: Distinction is explicitly possible using the attributes “*dir*” and “*path*“
- What happens if we watch a directory in a non-recursive manner?
  - Same behavior as observed for files (and not recursive!)
- What happens if we monitor a file recursively?
  - Same behavior as observed for directories

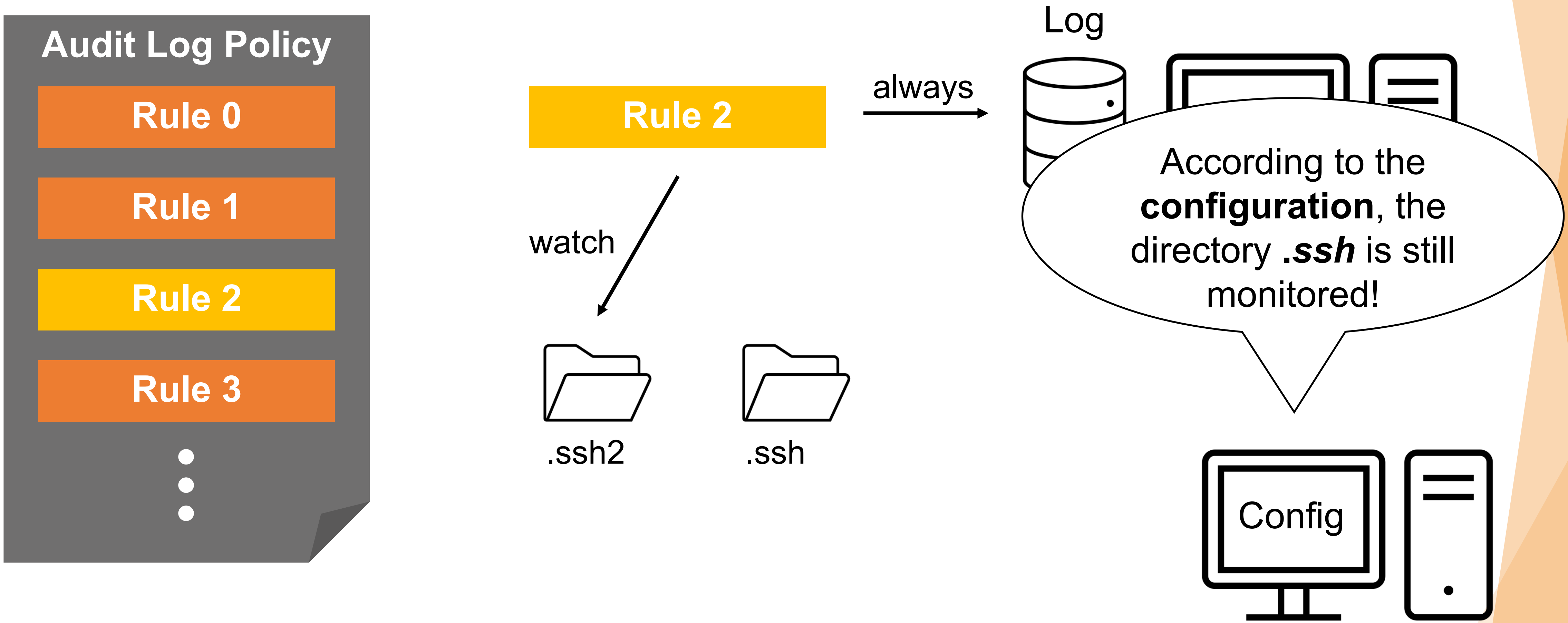
# Monitoring of files and directories

## Recursive (directory) vs. non-recursive (file) monitoring

| Action | Non-recursive monitoring | Non-recursive policy rule | Recursive monitoring | Recursive policy rule |
|---|---|---|---|---|
| **Rename** | - | unchanged | monitored | unchanged |
| **Delete** | - | unchanged | - | rule gets removed |
| **Recreated with same name** | monitored | unchanged | - | unchanged |

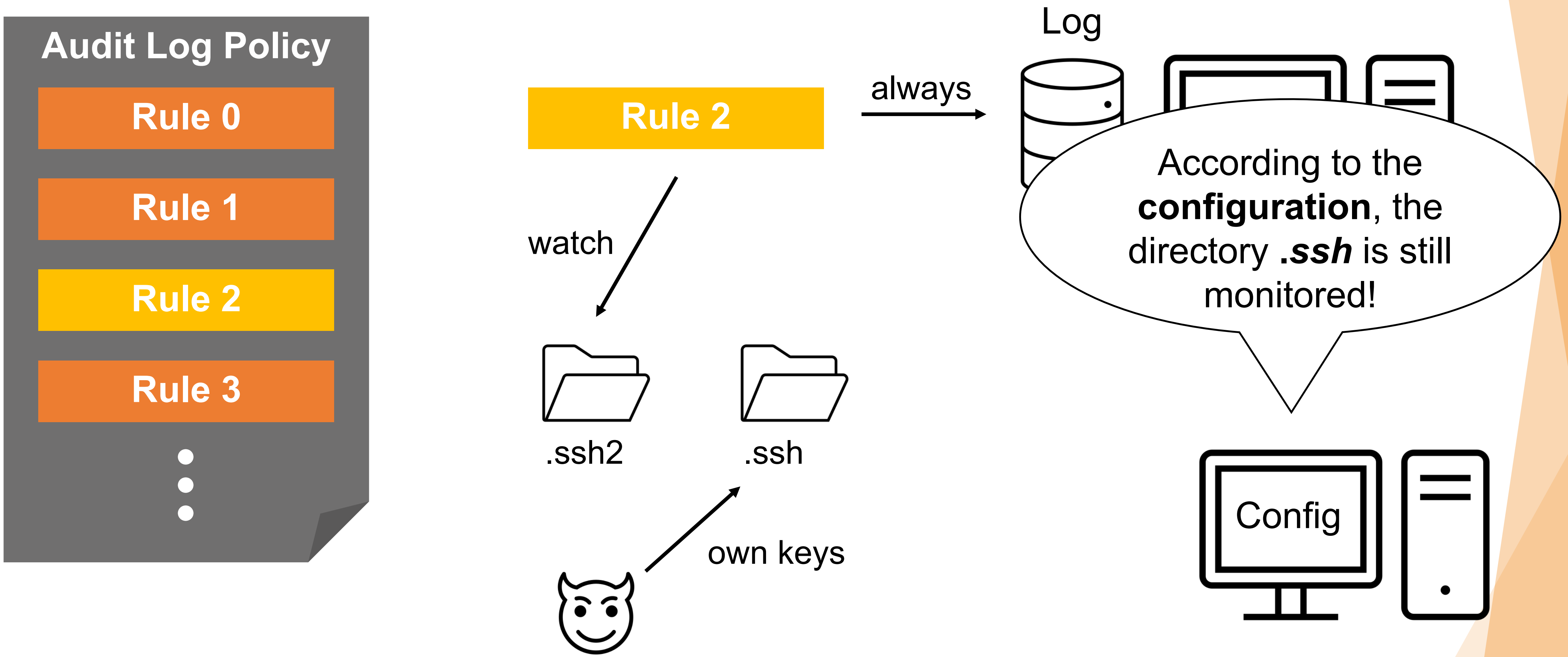
# Monitoring of files and directories

## From an attacker’s perspective


Audit Log Policy
Rule 0
Rule 1
Rule 2
Rule 3
Rule 2
always
Log
watch
.ssh2
.ssh
According to the **configuration**, the directory ***.ssh*** is still monitored!
Config

# Monitoring of files and directories

## From an attacker's perspective

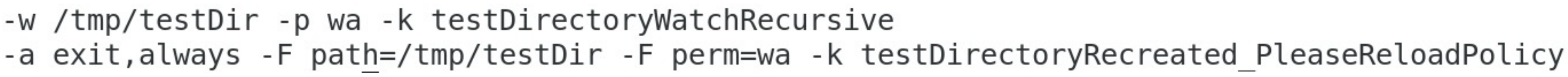


According to the **configuration**, the directory ***.ssh*** is still monitored!

# Monitoring of files and directories

## How to deal with it

- Monitoring behavior of files / directories depends on whether the monitoring is recursive or not
- **Problem** with the **renaming of directories**:
  - How does the analyst know if the current audit configuration reflects the true monitoring behavior?
- **Possible solution**: Use a **non-recursive backup rule** to detect folder changes (renaming or deletion)



```
-w /tmp/testDir -p wa -k testDirectoryWatchRecursive
-a exit,always -F path=/tmp/testDir -F perm=wa -k testDirectoryRecreated_PleaseReloadPolicy
```

# Outline

- auditd 101
- File watch implementation
- **Monitoring of files and directories**
- Logging actions always and never
- Conclusion

# Outline

- auditd 101
- File watch implementation
- Monitoring of files and directories
- **Logging actions always and never**
- Conclusion

# Logging actions always and never

## Insights

- Two identical rules with different actions do not trigger under the same conditions
  - Read access to the file */tmp/test* was monitored

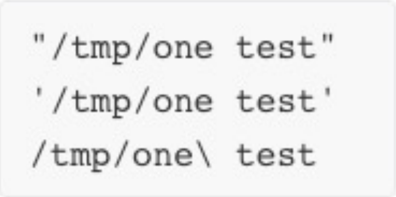
    *-a exit,**never** –F path=/tmp/test –F perm=r*

    *-a exit,**always** –F path=/tmp/test –F perm=r –k ShouldNotTrigger*
  - After renaming the file to *test2* and back to *test*, the exclusion didnot work anymore
- Observed only for non-recursive monitoring (*path=)*
- Reasons remain unclear
- *RedHat, Inc.* was contacted

# Logging actions always and never

## Auditbeat as an alternative?

- Using auditbeat instead of auditd
- Advantages of auditbeat
  - Reliable filtering of events (with reservations)
  - Usage of auditd log policies
  - Extensive options for log processing and enrichment
- Drawbacks of auditbeat
  - Difficult to deploy one policy on a variety of systems
  - Files / folders with spaces cannot be used in rules
    - Monitoring only possible in an “indirect manner”



```
"/tmp/one test"
'/tmp/one test'
/tmp/one\ test
```

# Outline

- auditd 101
- File watch implementation
- Monitoring of files and directories
- **Logging actions always and never**
- Conclusion

# Outline

- auditd 101
- File watch implementation
- Monitoring of files and directories
- Logging actions always and never
- **Conclusion**

# Conclusion

## Are they no more than little quirks?

Identified quirks:

- File watches only consider *open* & *openat* syscalls (CVE)
- Recursive monitoring lacks consistent behavior
- Non-recursive exclusion rules seem to be ignored sometimes

# Conclusion

## Are they no more than little quirks?

Advantages of auditd:

- Rule-Matching in the kernel increases tamper resistance
- Syscall rules offer fully transparent monitoring
- Many useful configuration options (e.g. ignore policy errors)
- Variety of tools to facilitate usage
- Many built-in logs provided with additional context

# Conclusion

## Are they no more than little quirks?

My personal conclusion:

- Advantages outweigh the disadvantages
- If reliable filtering is essential, auditbeat seems to be a good alternative (with reservations)
- Since auditbeat brings its own little quirks, a compromise is inevitable

# Thank you for your attention!

# Are there any questions?

felix.kosterhon@secuinfra.com

[1]

# References

- [1]: https://www.pngwing.com/en/free-png-nbyly