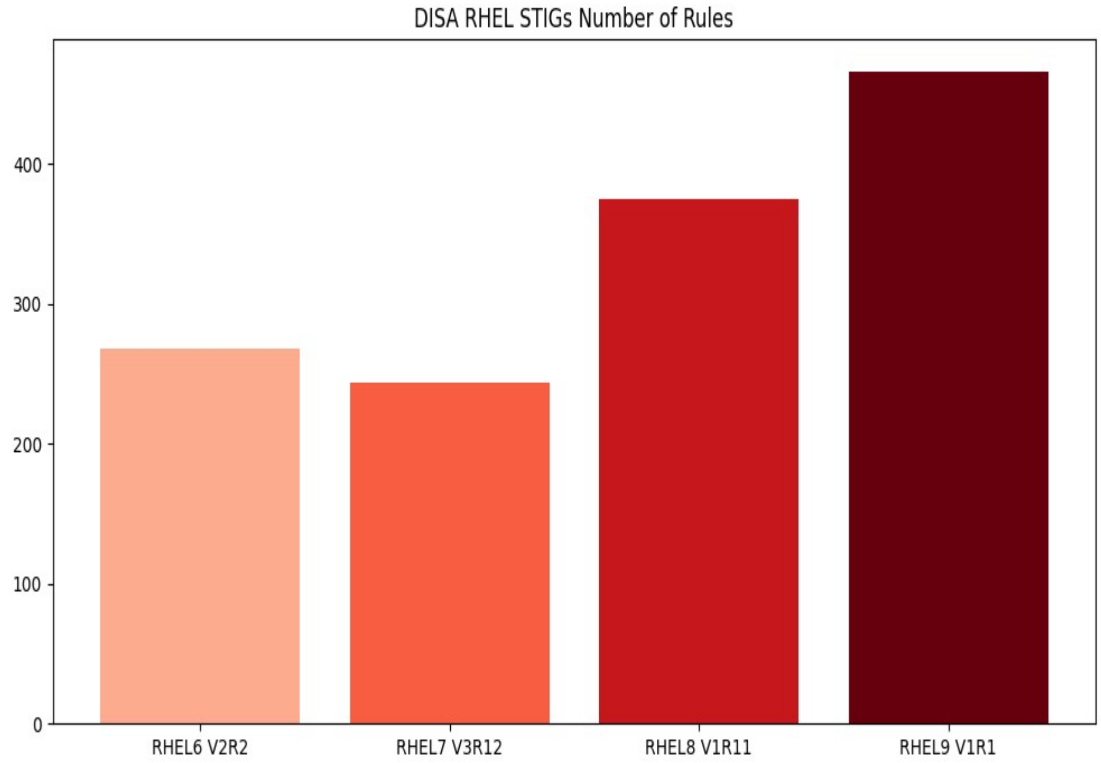
Reducing RHEL9 STIG Performance Impacts

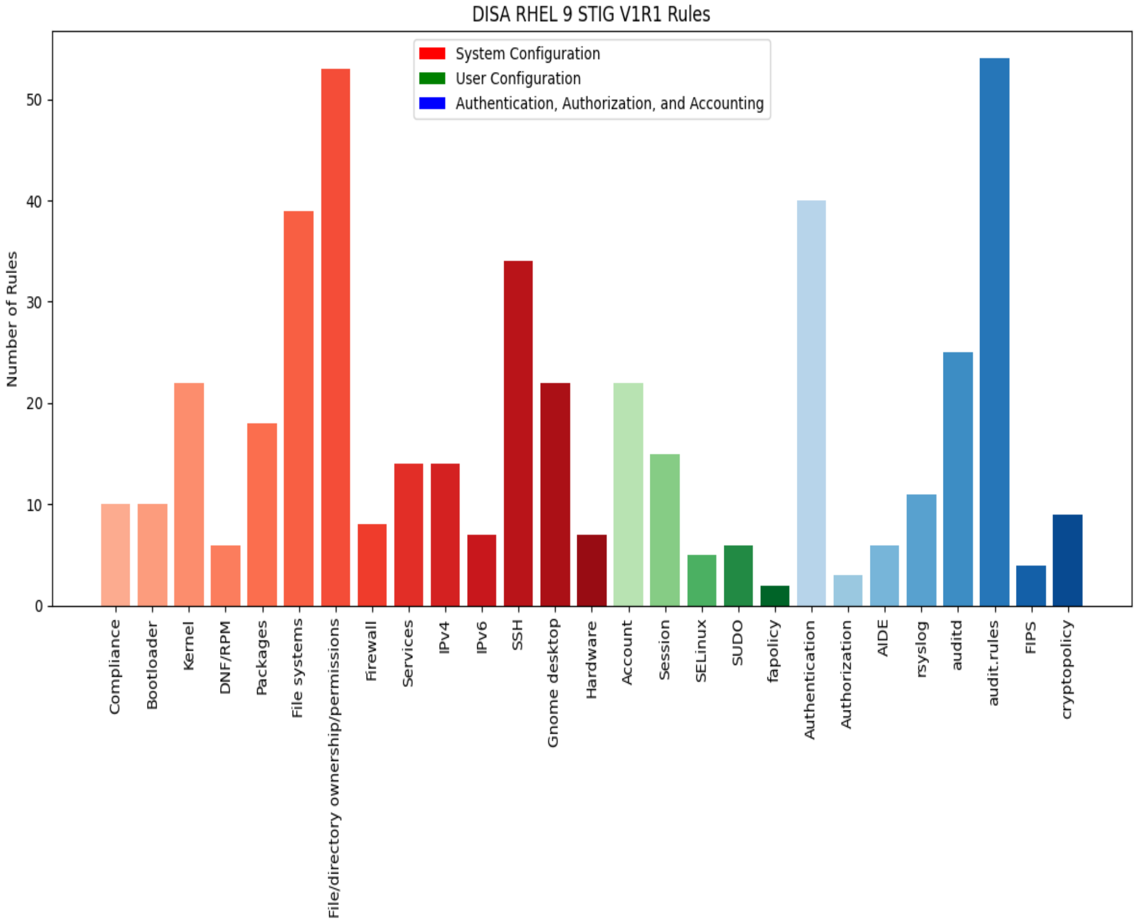
Jeremy Filizetti
jfilizetti AT ultrascale.net

RHEL STIG Overview



- Growing number of rules
  - Software is more featured but this isn't the only reason
  - Increasing micro-management items
    - 3 rules to make sure a file is owned by root:root with specific permission

RHEL 9 Rules by Category



- Organization has improved
  - Some items fit in multiple categories
- Bad recommendations remain
  - Carving your disk up like a thanksgiving day turkey when using HDD
  - File permissions still seem misunderstood by people creating it
  - Some have the potential to create security issues themselves

STIG Severity Guidelines

- Severity ratings on many rules continue to be inaccurate
- Some are downright off the wall

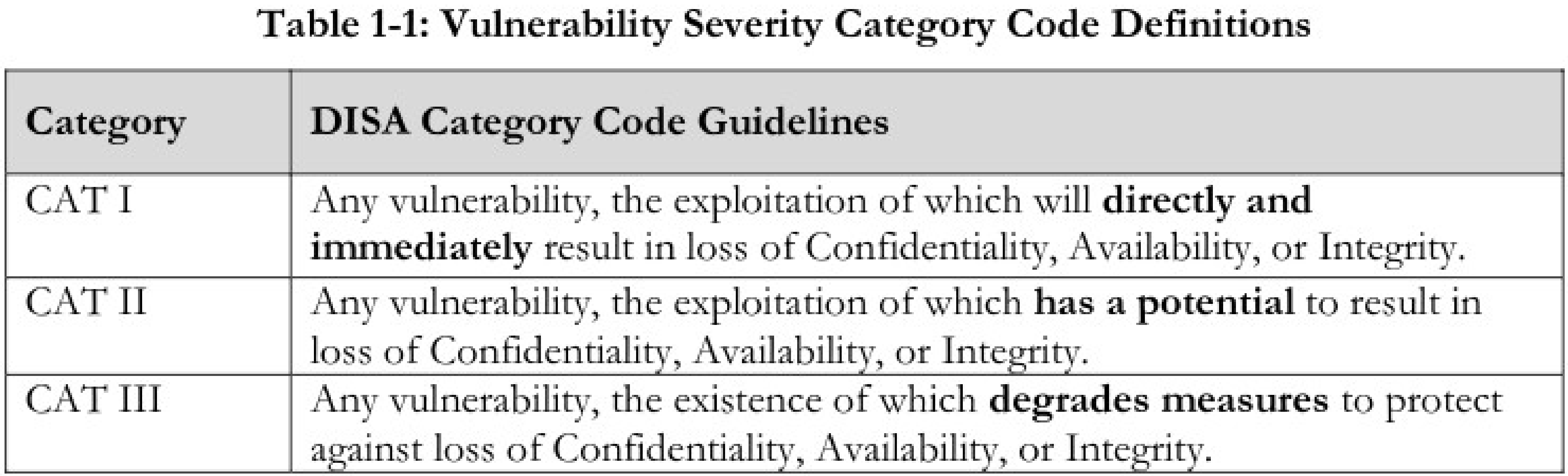


Table 1-1: Vulnerability Severity Category Code Definitions

Category	**DISA Category Code Guidelines**
CAT I	Any vulnerability, the exploitation of which will **directly and immediately** result in loss of Confidentiality, Availability, or Integrity.
CAT II	Any vulnerability, the exploitation of which **has a potential** to result in loss of Confidentiality, Availability, or Integrity.
CAT III	Any vulnerability, the existence of which **degrades measures** to protect against loss of Confidentiality, Availability, or Integrity.

RHEL9 STIG Performance Testing

- Many STIG configuration changes have minor impacts
- A few items have major impacts
- Testing was done to highlight several things
  - Increased Latency
  - Limiting throughput
  - Wasted CPU
- **Names HBSS/McAfee/Trellix synonymously throughout**

Equipment Used

- Hardware
  - Dell Poweredge R730
  - 2 – Intel E5-2620 v4
  - 128 GB RAM
  - Storage on Intel Optane 900P
- VM
  - 8 vcpu
  - 32 GB RAM
  - qcow2 (no compression, backed by ext4 file system on optane)

Software Used

- Linux perf
- Flamegraphs
- vmstat
- Various custom python scripts to graph with matplotlib
- LibreOffice for some graphs
- draw.io for sequence diagram

Testing

- Take a batch of small files and copy them
  - Used linux kernel source (6.7.2)
    - 1.5 GB, 5382 directories, 82375 files
  - Stored on a local file system
    - No exceptions from fapolicy or McAfee/Trellix ENSL
  - Copy files, copy extended attributes, set times (cp -a)
    - Triggers audit actions to stress audit system

Rough sequence of tests

- Tests ran for concurrency of 1 2 4 and 8

  -
    ```
    echo 3 > /proc/sys/vm/drop_caches
    ```

  -
    ```
    sudo perf record -o perf.cp.data -F 47 -a -g sudo -u jeremy
    /bin/time -f 'seconds %e  system: %S  user: %U' bash -c "seq 1
    $concurrency | xargs -P 0 -i cp -a ~/linux-6.7.2 ~/dest_{} 2>
    /dev/null"  2>&1 | tee results.log
    ```

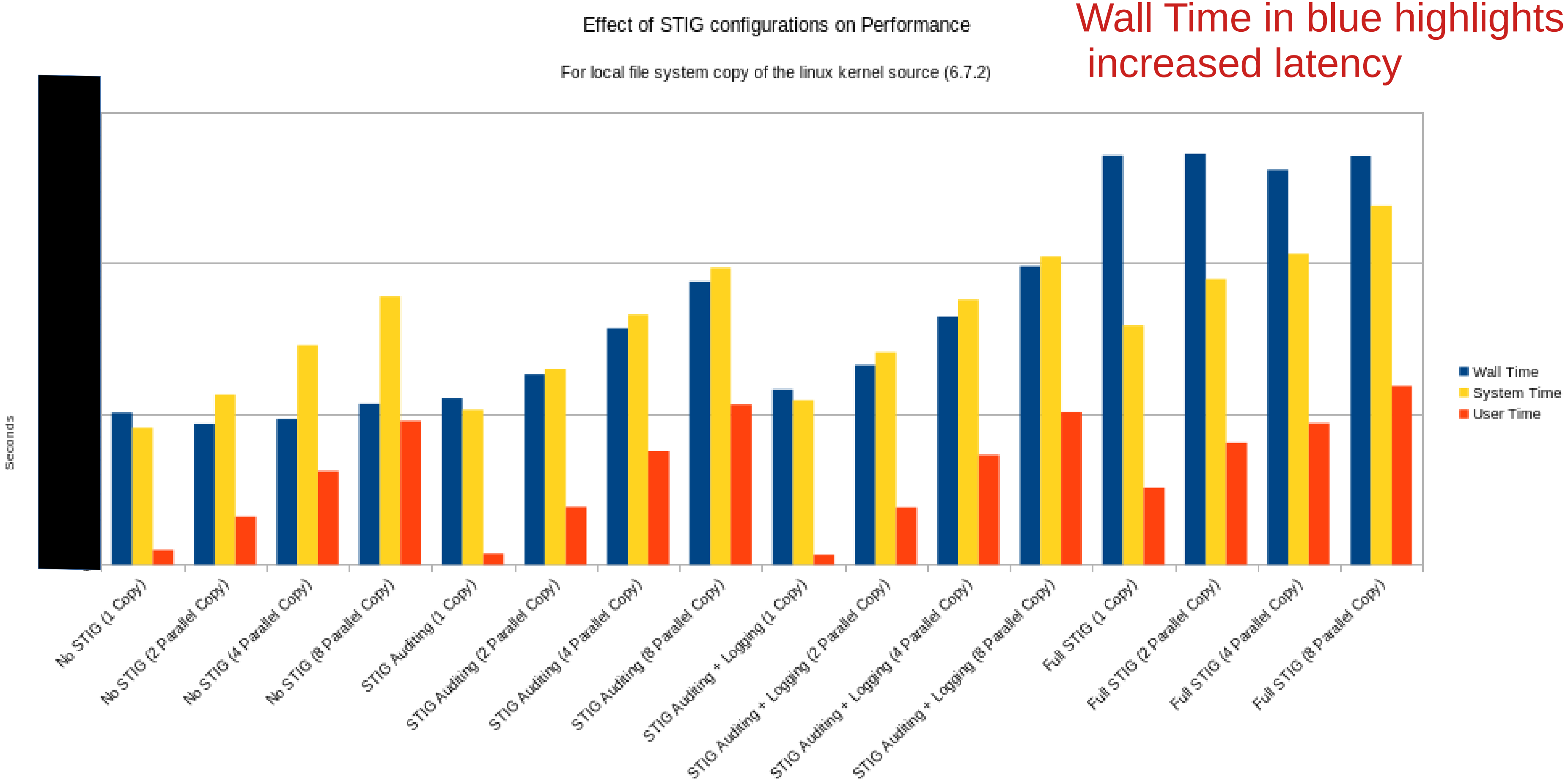
  -
    ```
    sudo perf script -i perf.cp.data > perf.cp.data.script
    ```

  -
    ```
    sudo perf report -i perf.cp.data --no-children --sort overhead,pid
    -F overhead,overhead_sys,overhead_us,pid  --max-stack=0 --stdio |
    tee perf-report.log
    ```

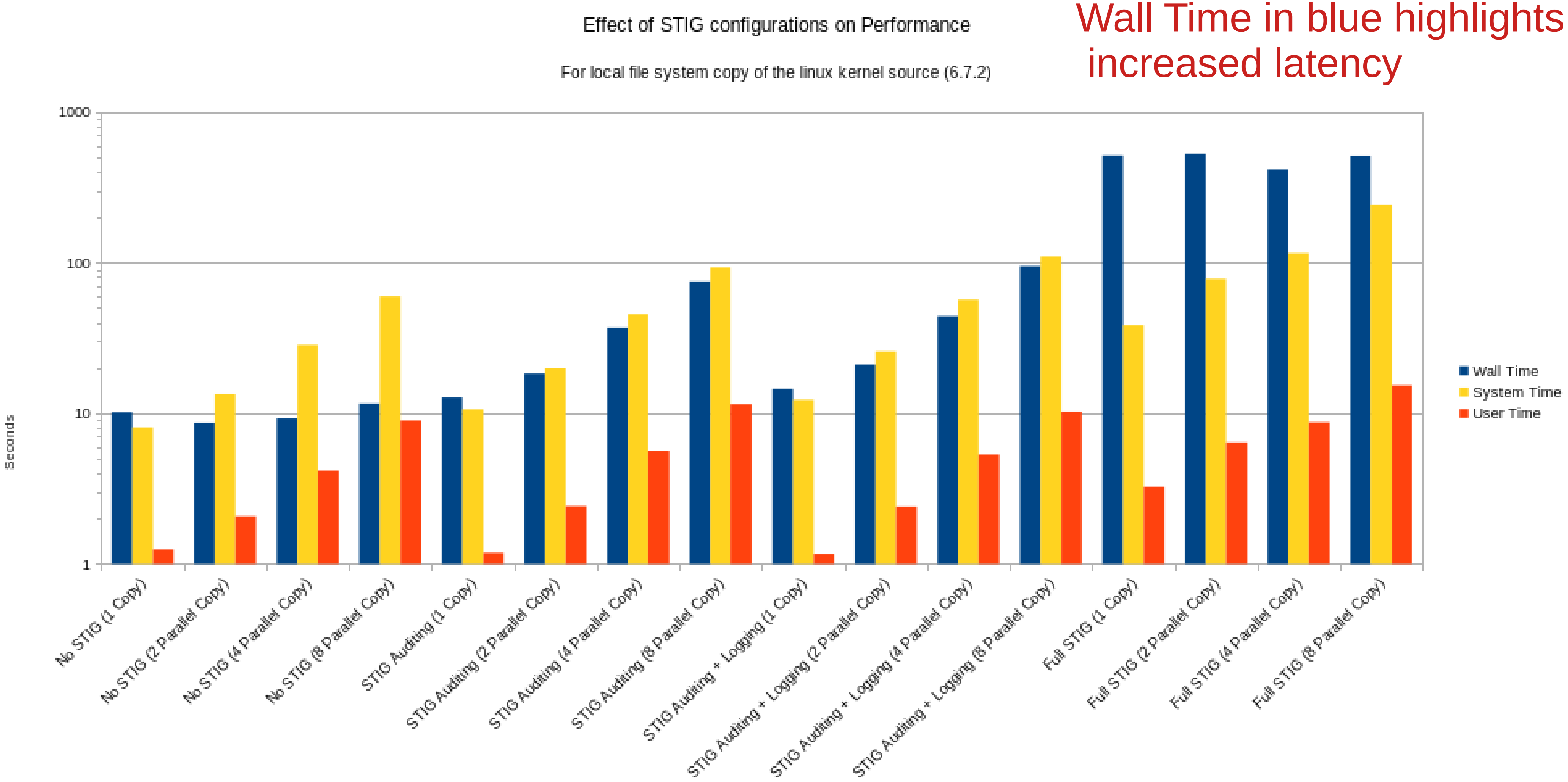
  -
    ```
    seq 1 $concurrency | xargs -P 0 -i rm -rf ~/dest_{}
    ```

Performance isn’t that bad?

Wall Time in blue highlights increased latency

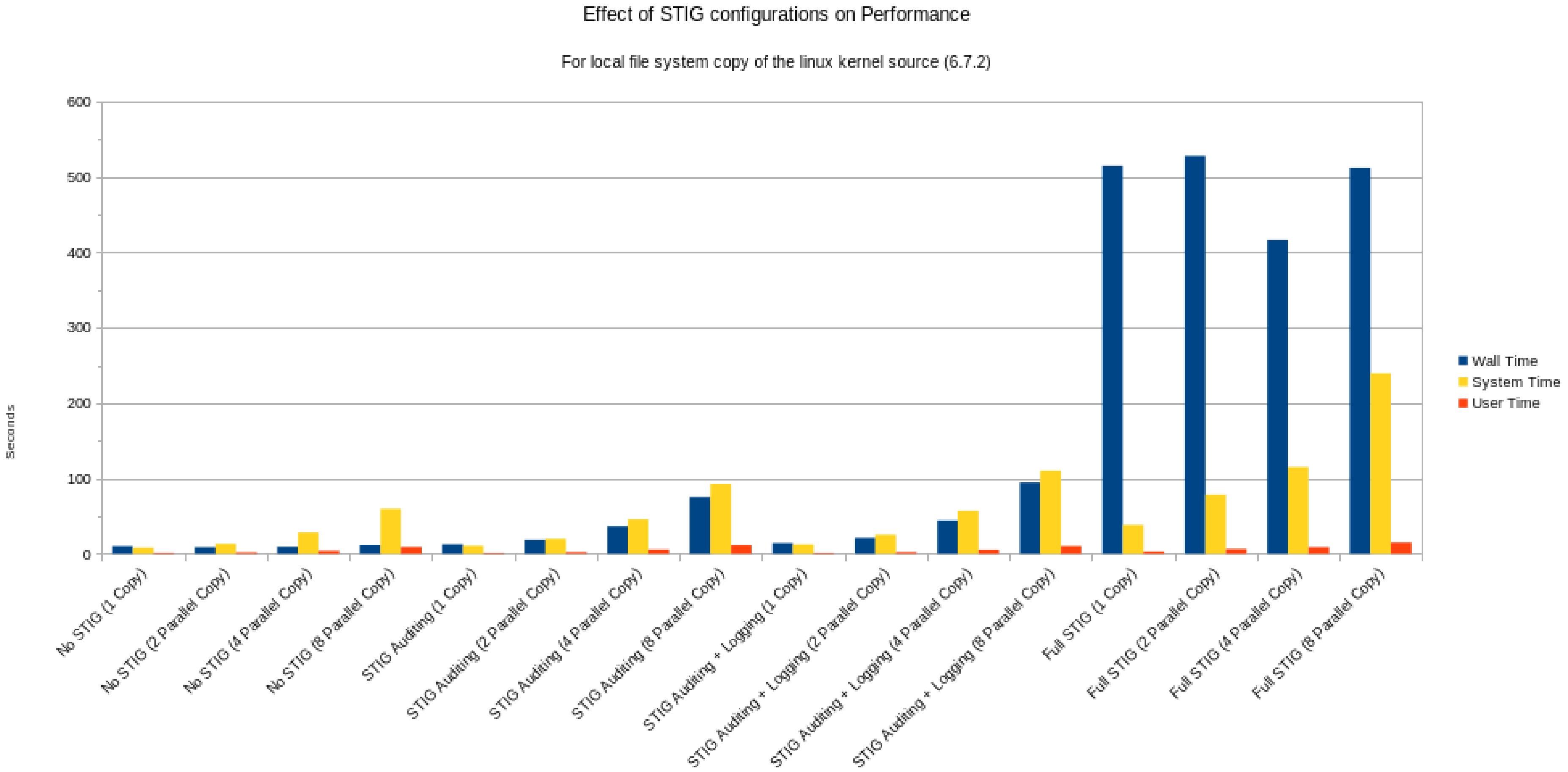


It is. Log scale is deceiving



Wall Time in blue highlights increased latency

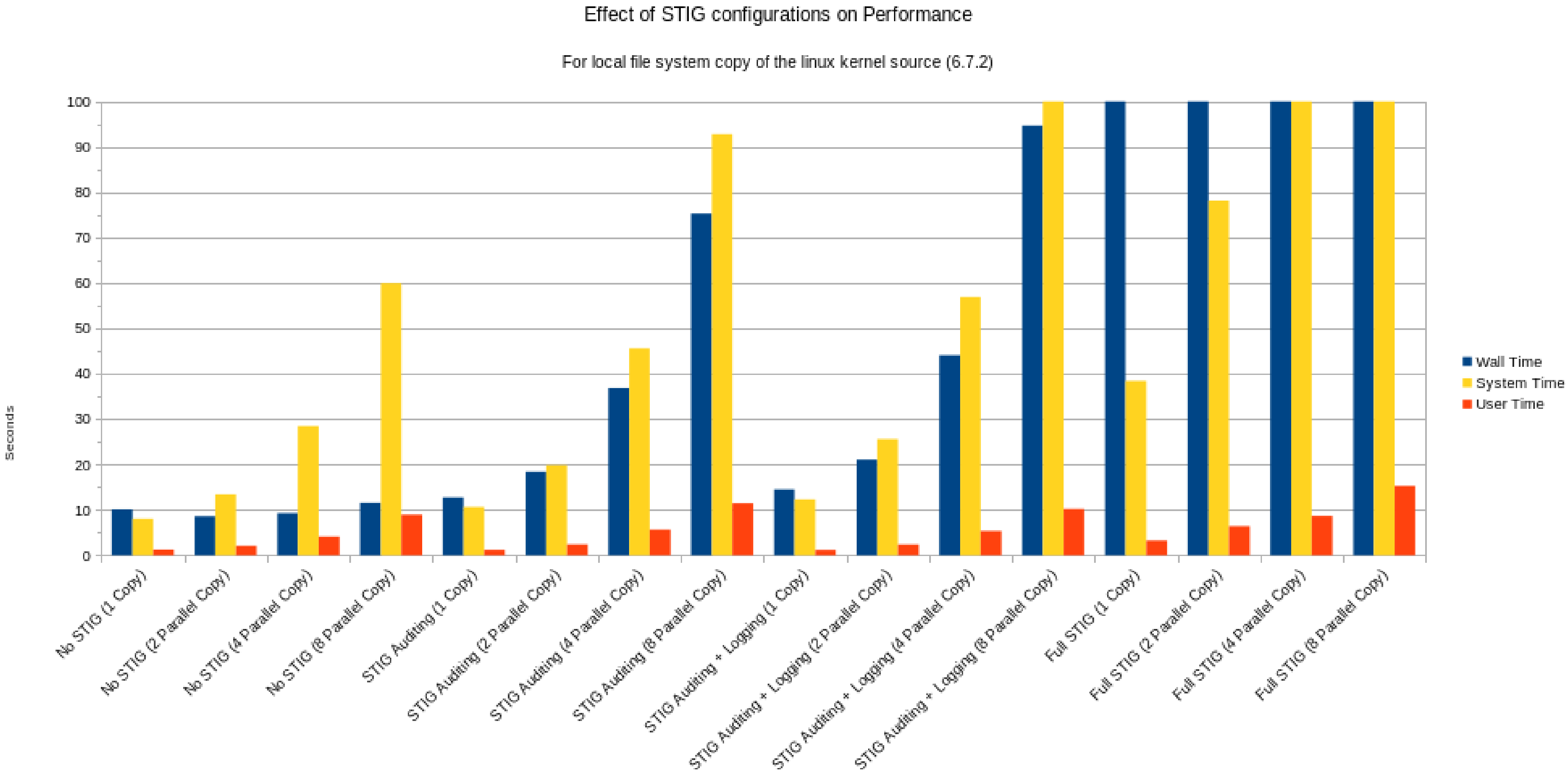
Performance is terrible



Effect of STIG configurations on Performance

For local file system copy of the linux kernel source (6.7.2)

Performance is terrible (zoomed)



Performance Impact

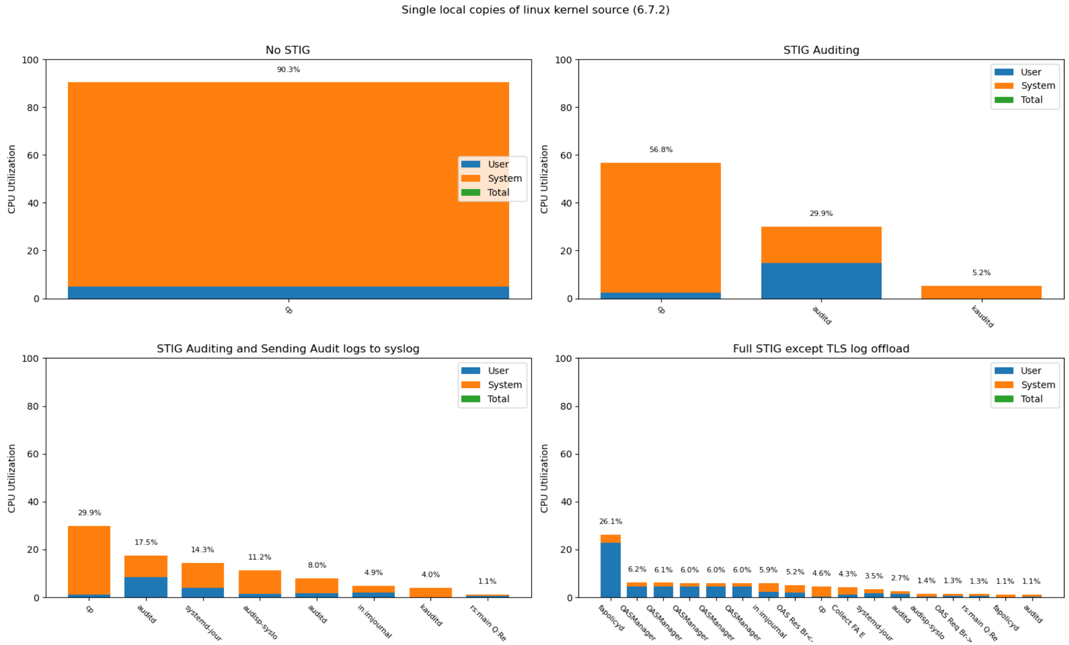
- Auditing
  - 1.5-6.5x longer wall time
  - Up to 1.5x more CPU utilization
- Auditing with audit logs sent to syslog
  - 1.4-8.2x longer wall time
  - Up to 2x more CPU utilization
- Full STIG
  - 44-61x longer wall time
  - Up to 5.8x more CPU utilization

Comparisons on following slides

Baseline (top left):
This is how things should run

Auditing and logging (bottom left)

Duplicative logging due to RHEL-09-652035.



Single local copies of linux kernel source (6.7.2)

But fixes deficiency of STIG log dropping due rate limiting. Added to /etc/rsyslog.conf:

```
$imjournalRatelimitInterval 0
$imjournalRatelimitBurst 0
```

Auditing impact (top right):

Auditing goes to audit daemon and auditd logging only.

Roughly the full STIG (bottom right):

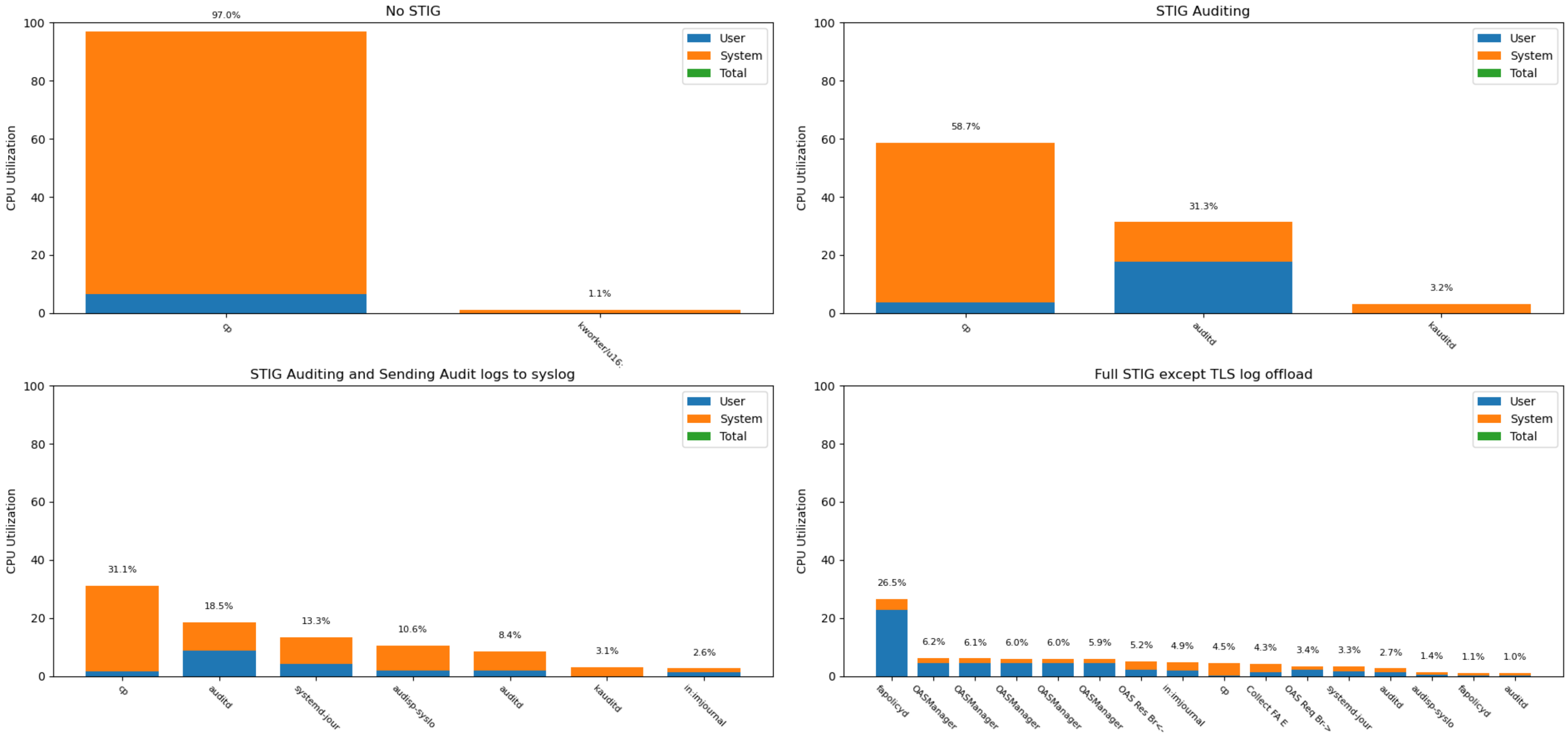
fapolicyd, McAfee impacts are visible in combination with auditing and logging.

By far the biggest impact is McAfee

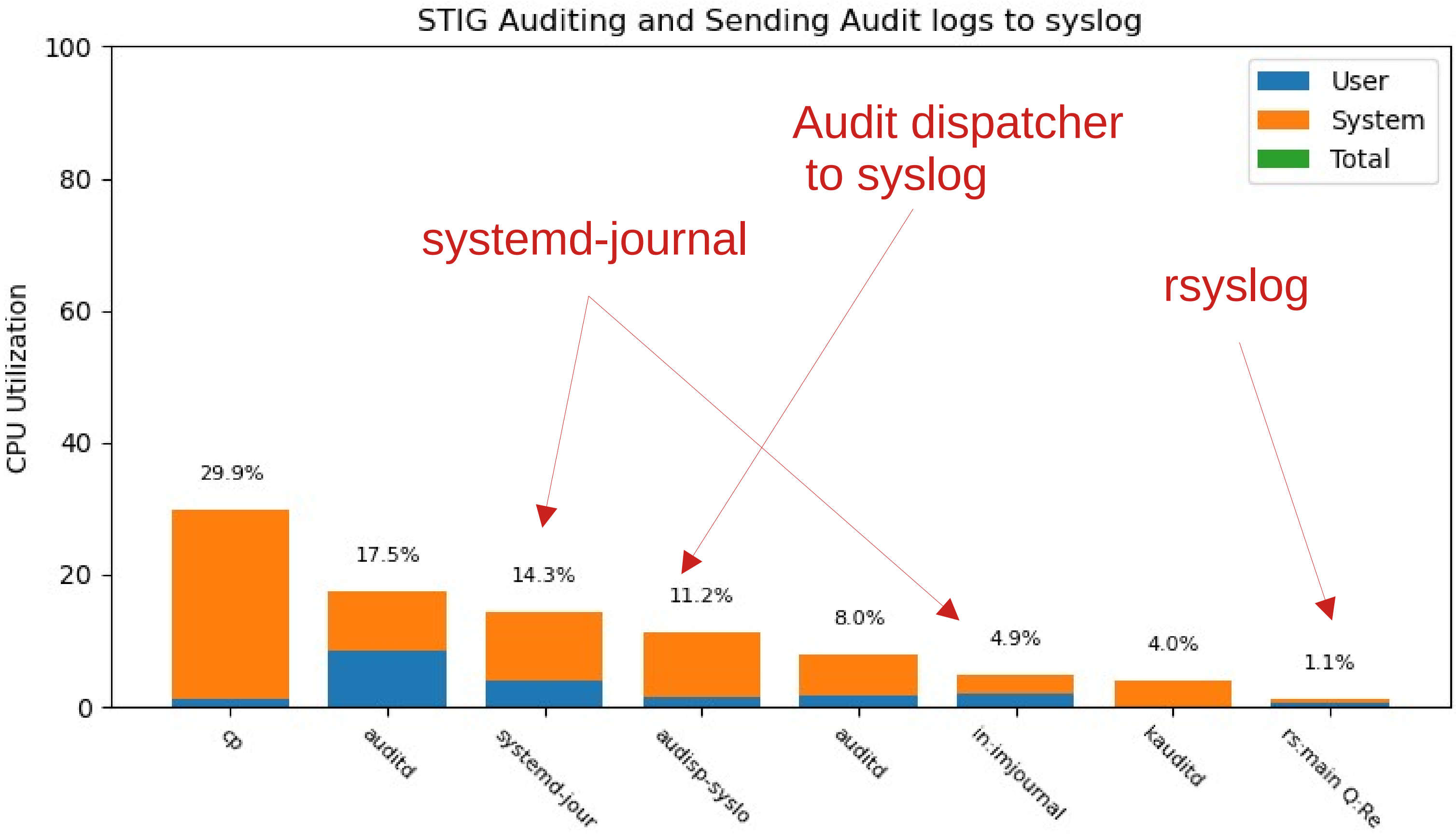
Idle/swapper thread removed from results

Single Copy (processes >1% CPU)

Single local copies of linux kernel source (6.7.2)



Duplicative Logging (RHEL-09-652035)

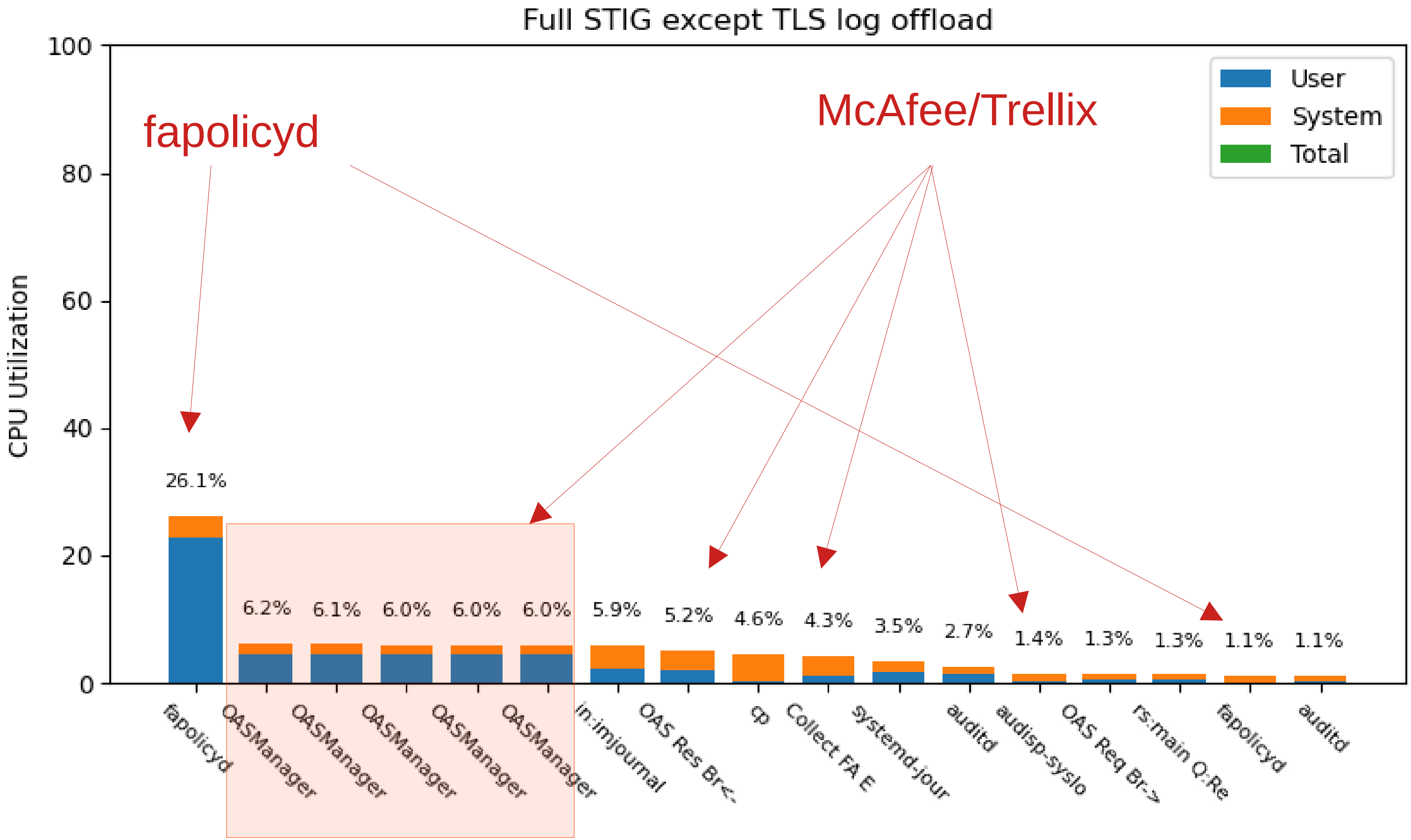


Audit dispatcher to syslog

systemd-journal

rsyslog

Full STIG consumed by “security” tools

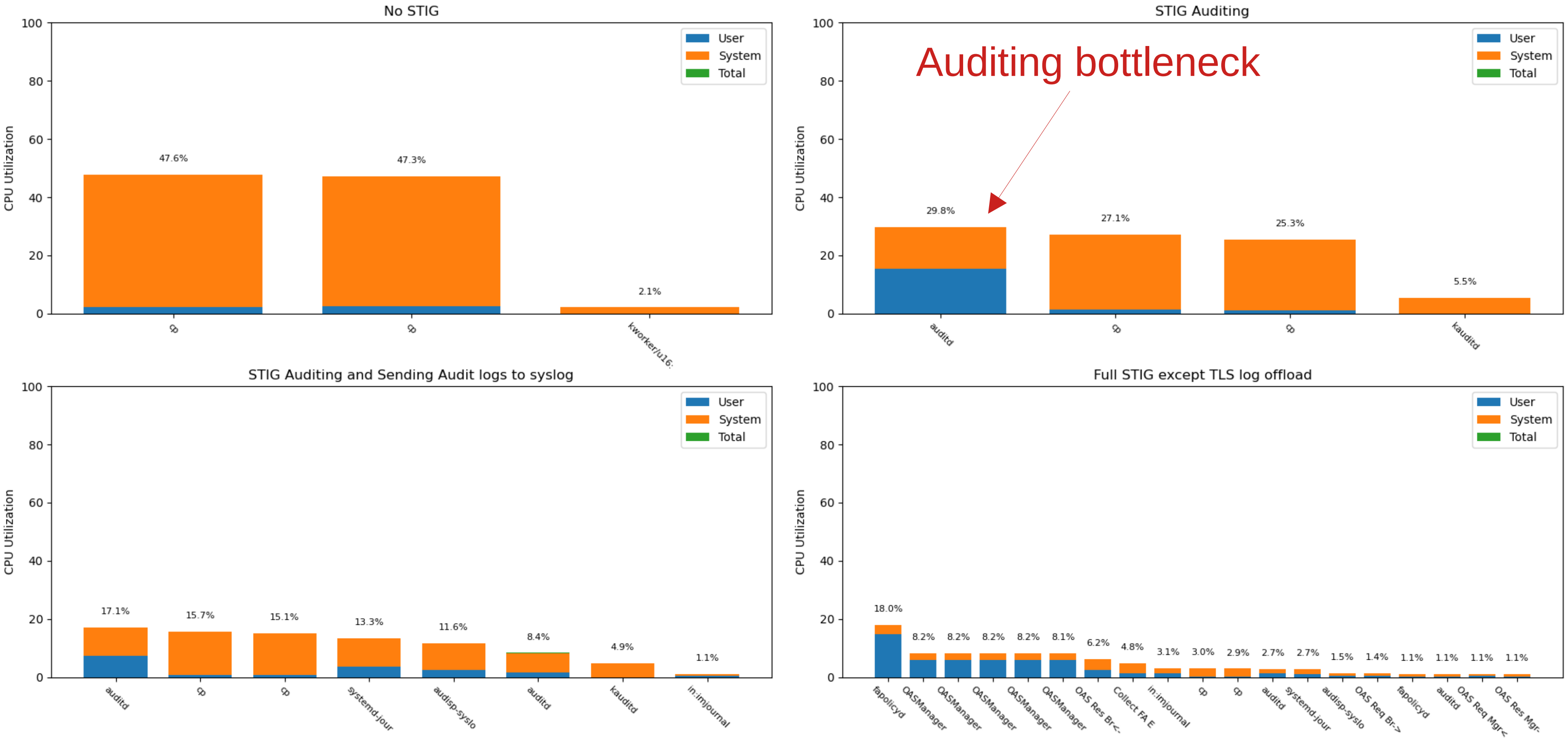


fapolicyd

McAfee/Trellix

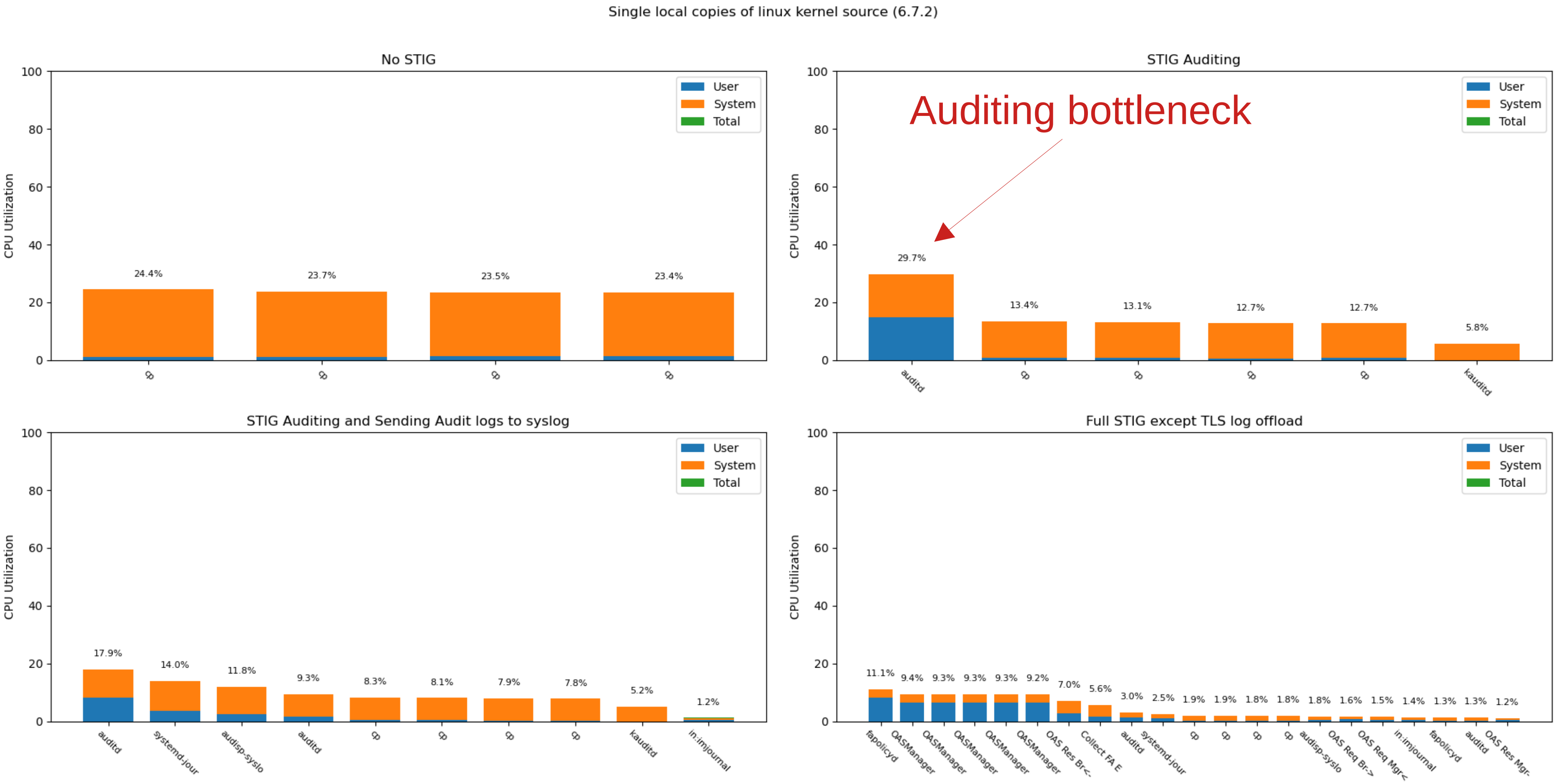
2 in Parallel (processes >1% CPU)

Single local copies of linux kernel source (6.7.2)



Auditing bottleneck

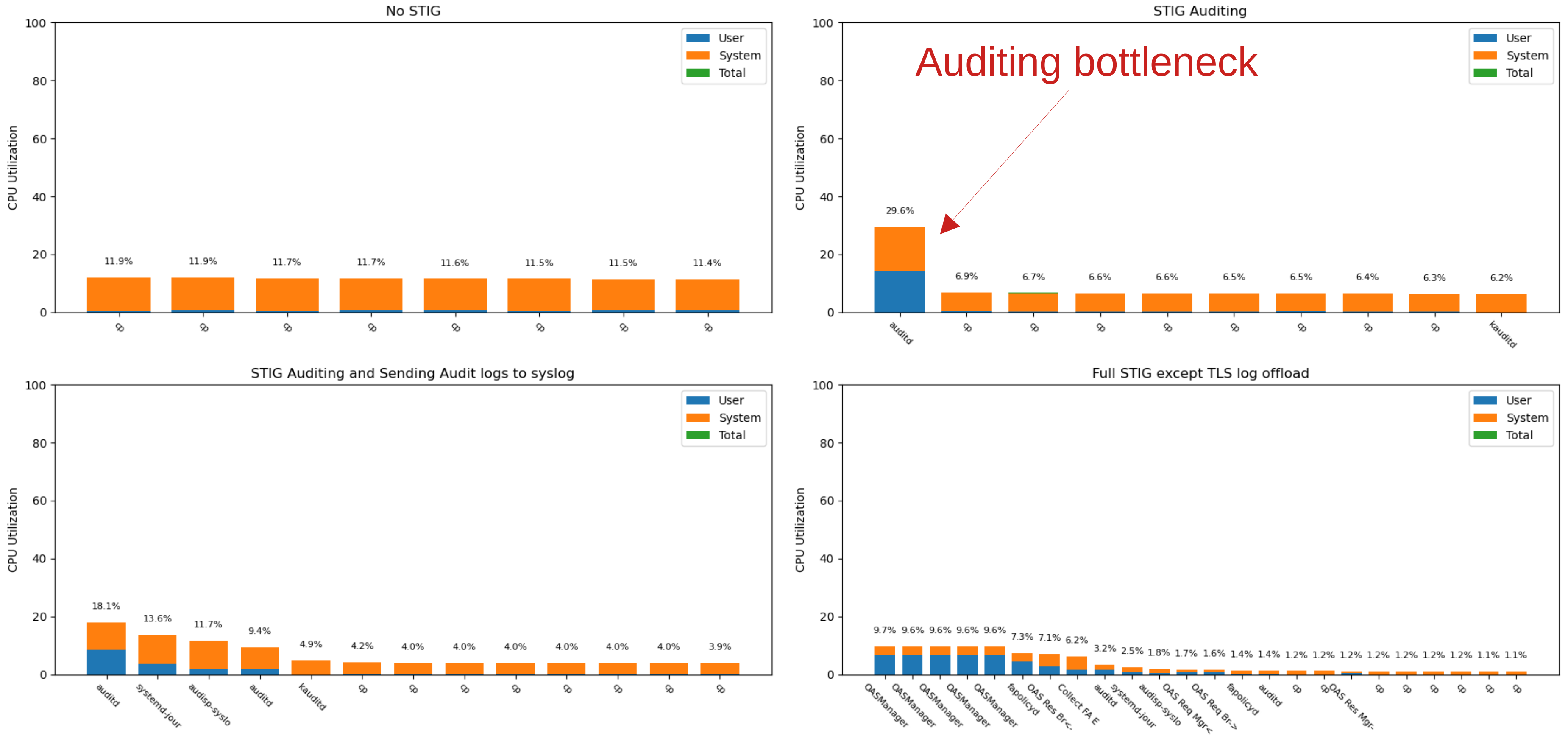
4 in Parallel (processes >1% CPU)



Single local copies of linux kernel source (6.7.2)

Auditing bottleneck

8 in Parallel (processes >1% CPU)



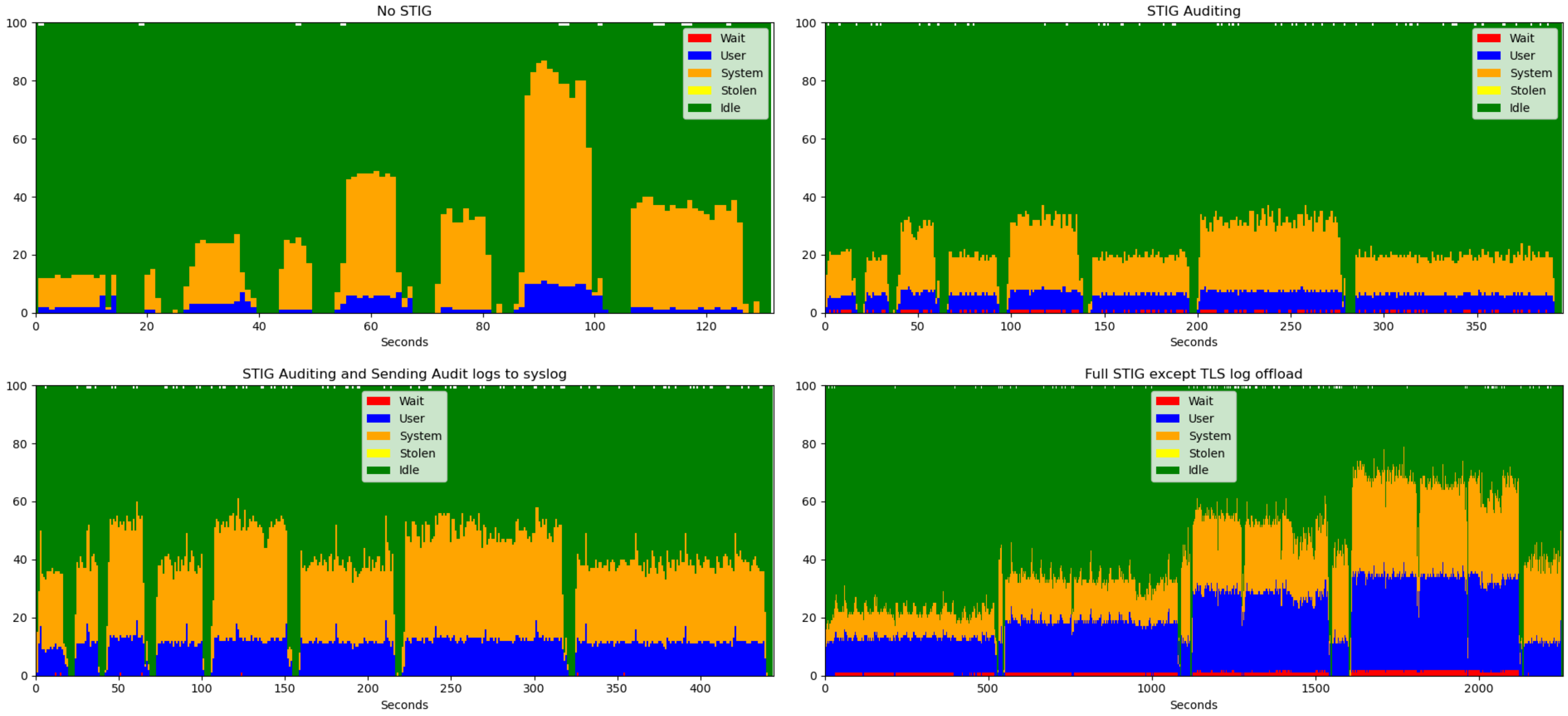
Single local copies of linux kernel source (6.7.2)

Auditing bottleneck

Tests in CPU Utilization graphs

1) Single copy
2) Remove files
3) Two copies in parallel
4) Remove files
5) Four copies in parallel
6) Remove files

All Tests CPU Utilization



No STIG

STIG Auditing

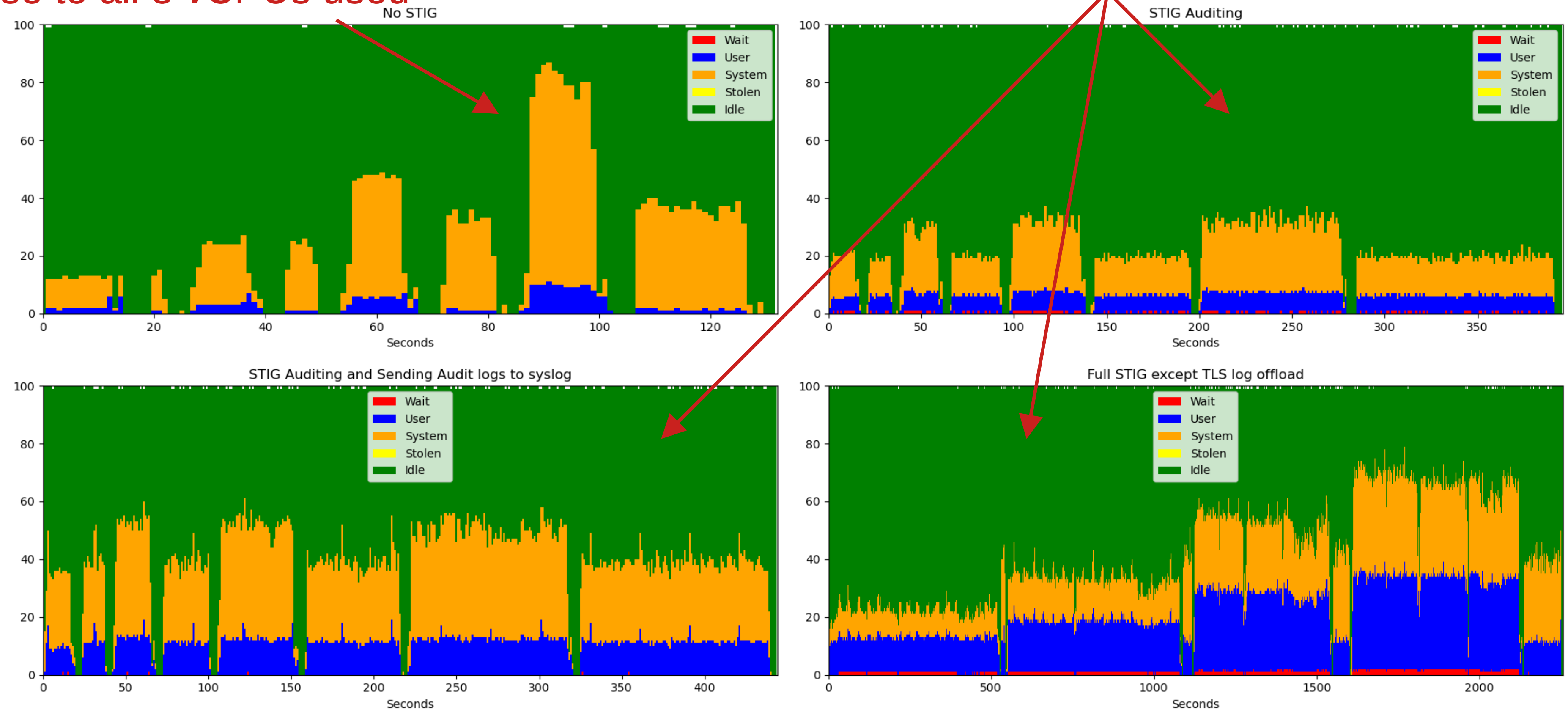
STIG Auditing and Sending Audit logs to syslog

Full STIG except TLS log offload

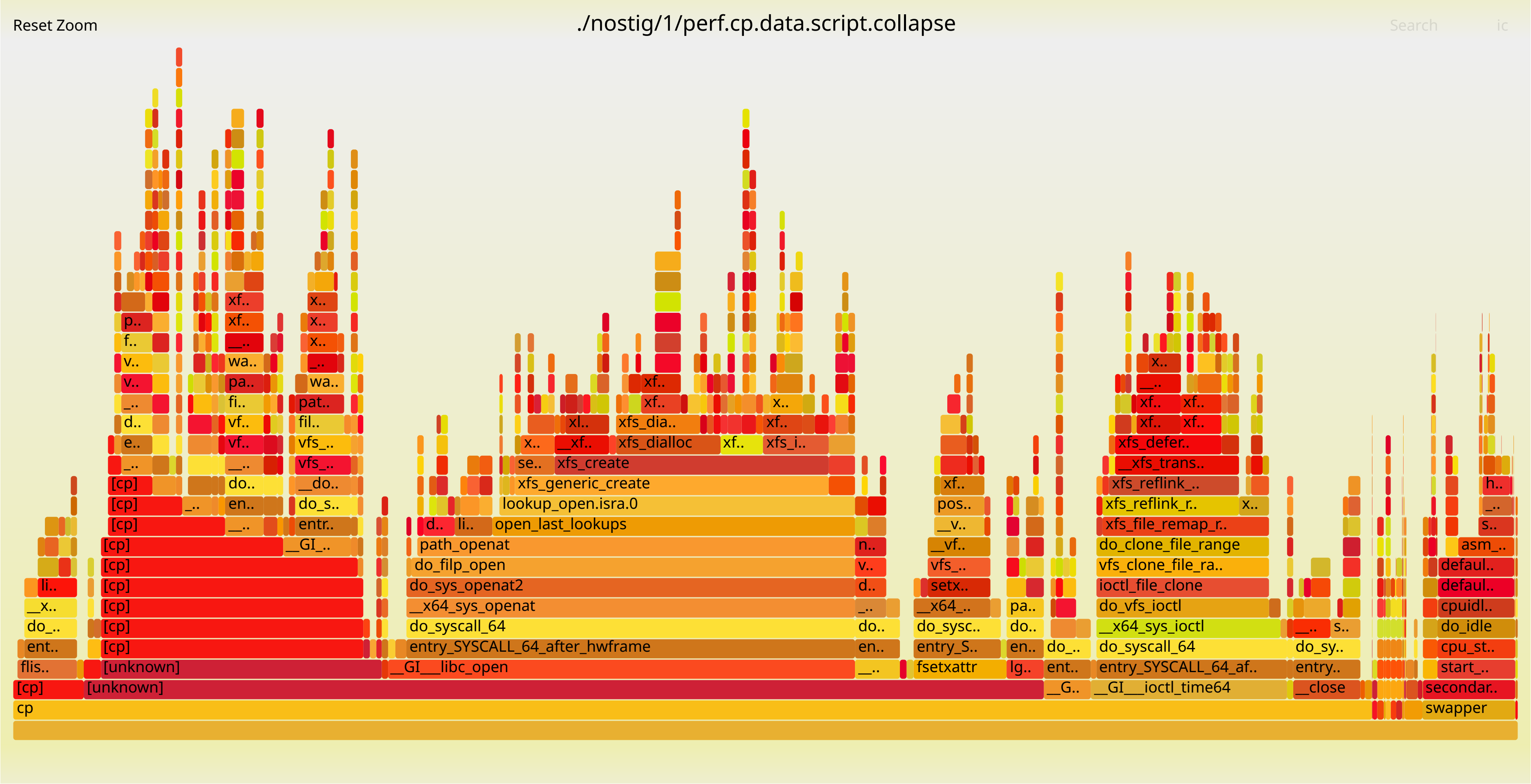
All Tests CPU Utilization (Annotated)

Copies are single threaded but with 8 close to all 8 vCPUs used

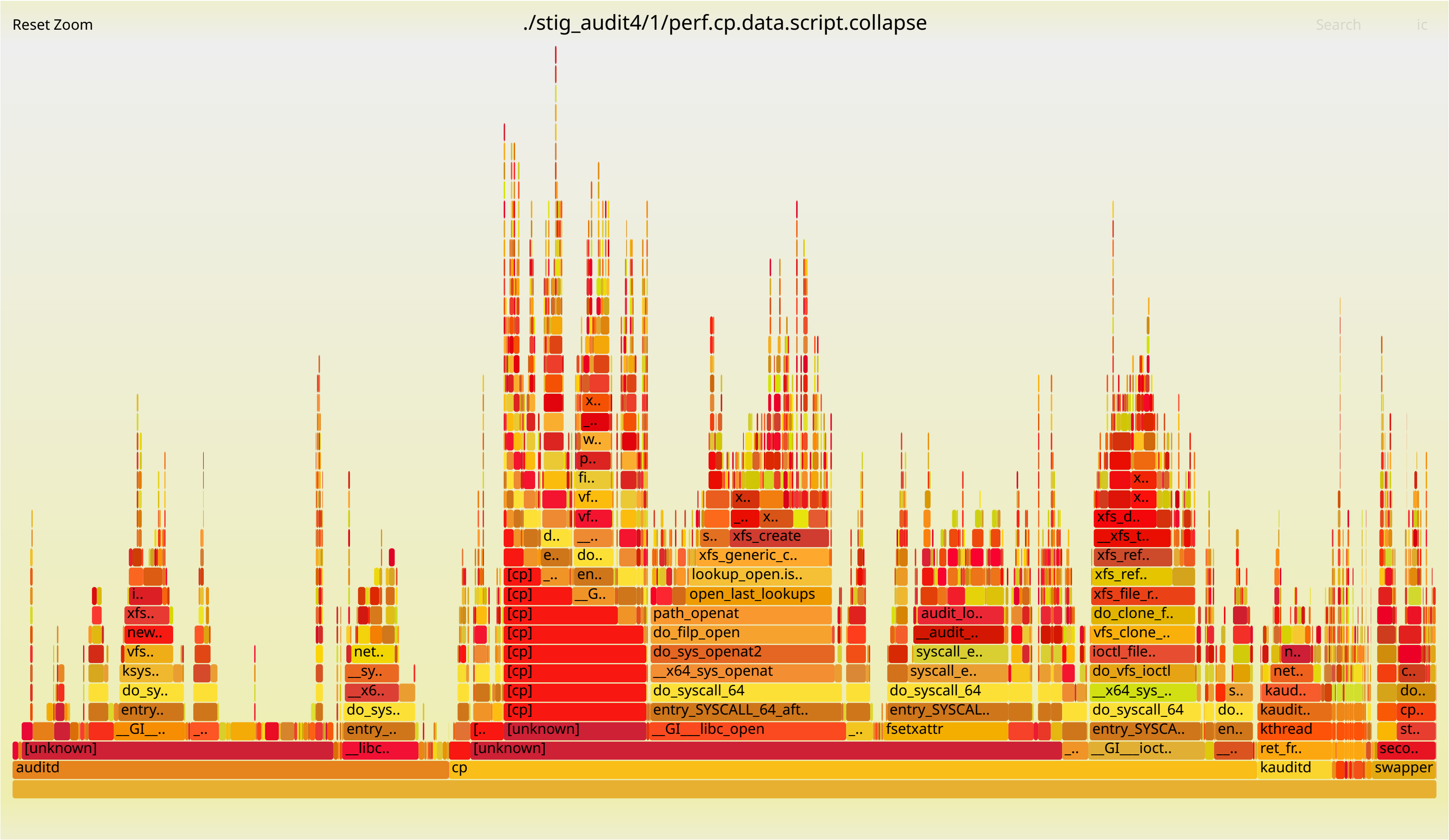
Performance starved by auditing bottleneck



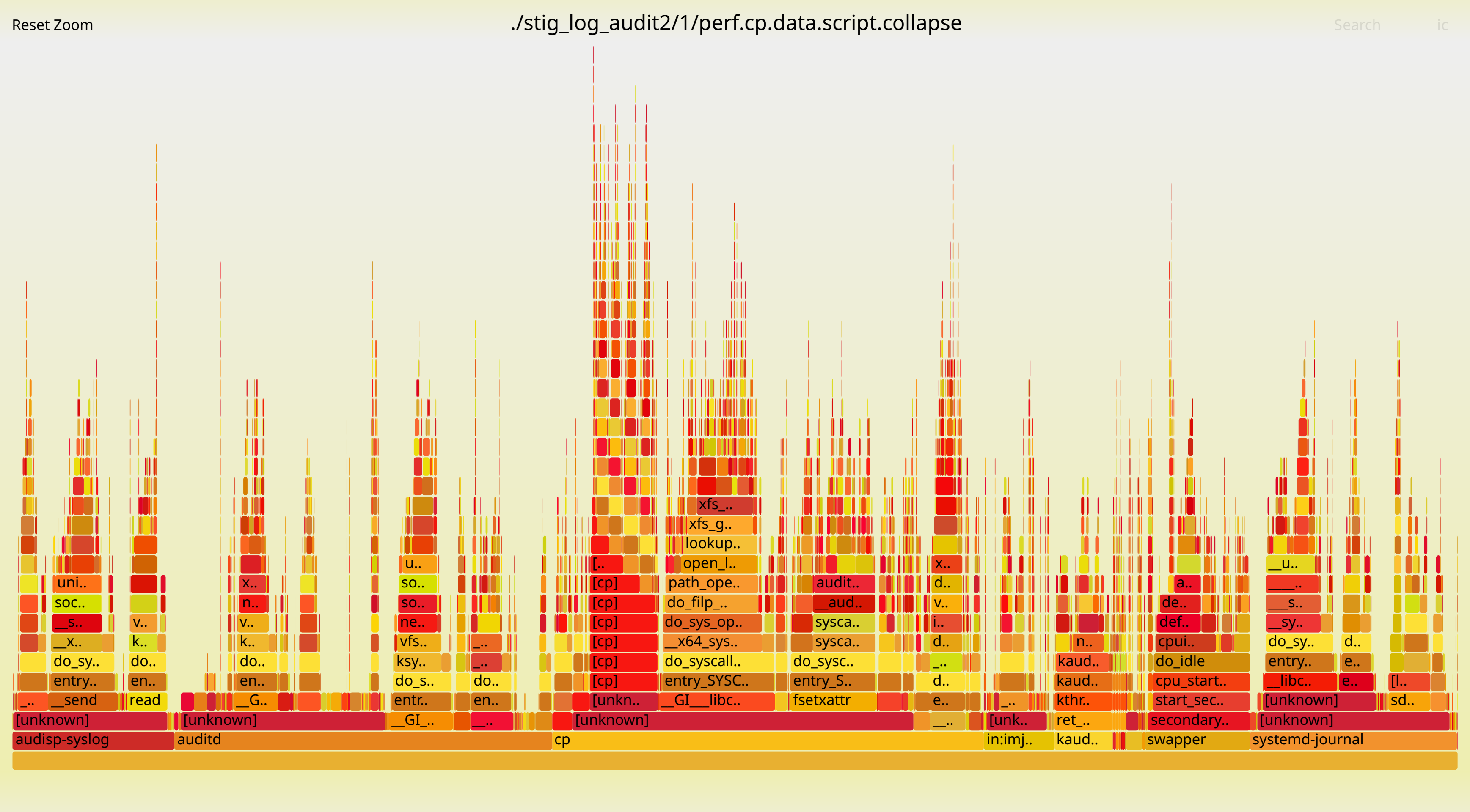
Single Copy No STIG



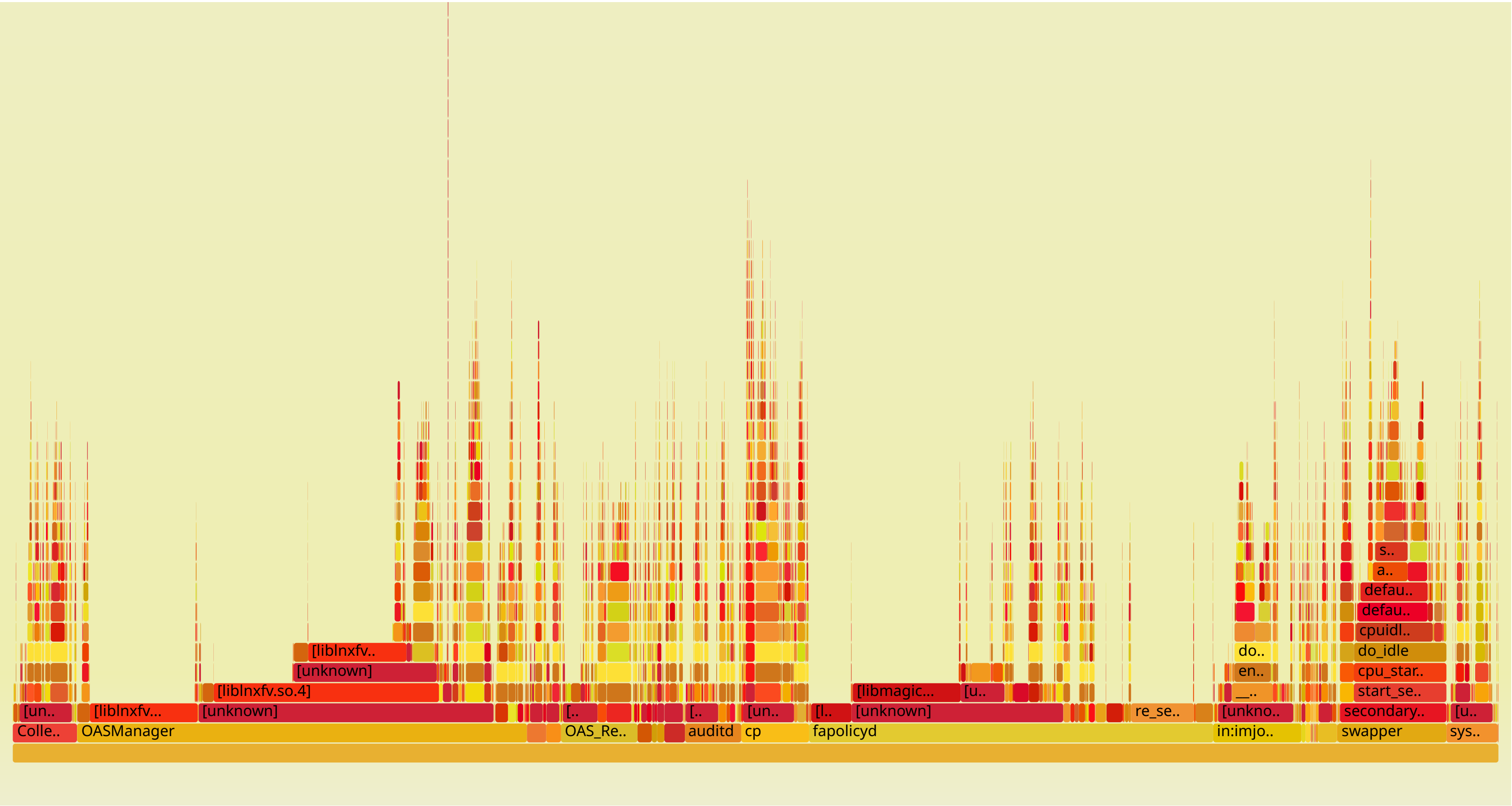
Single Copy STIG Auditing



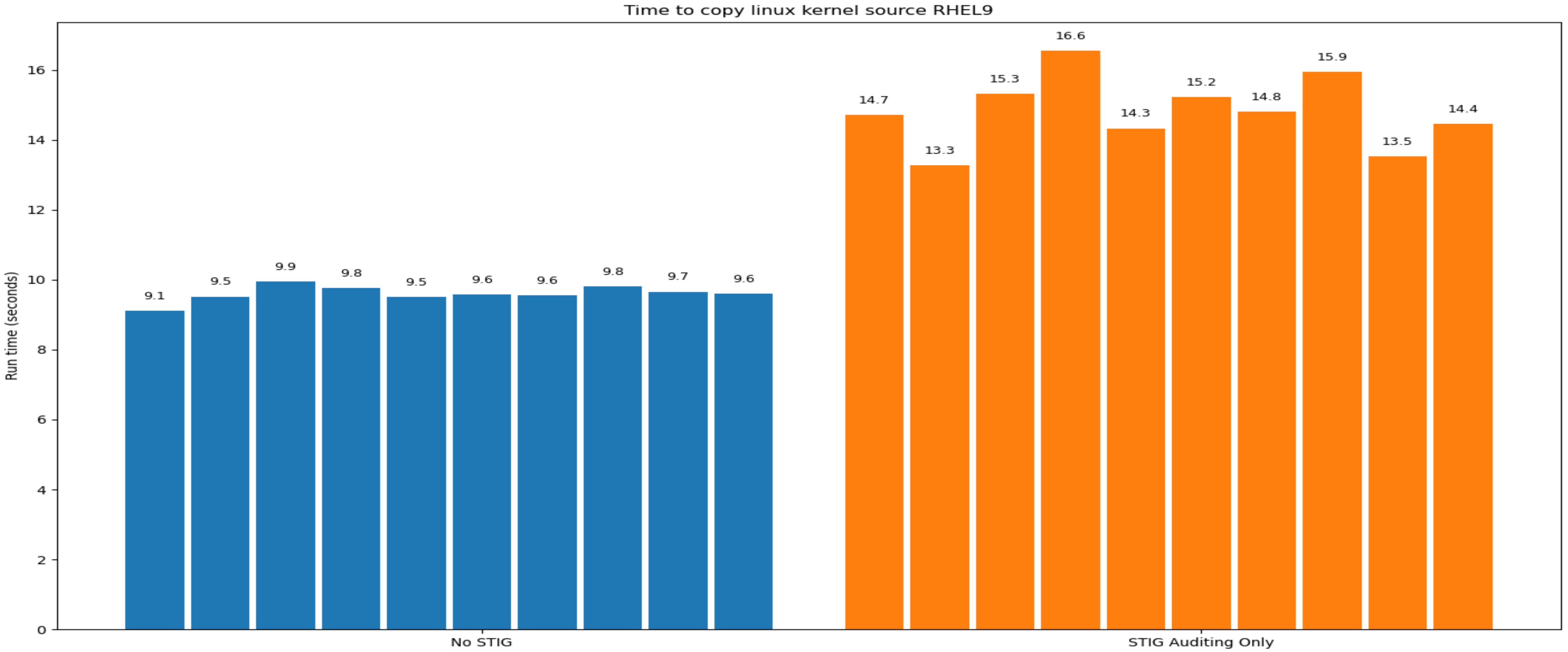
Single Copy STIG Auditing and Logging



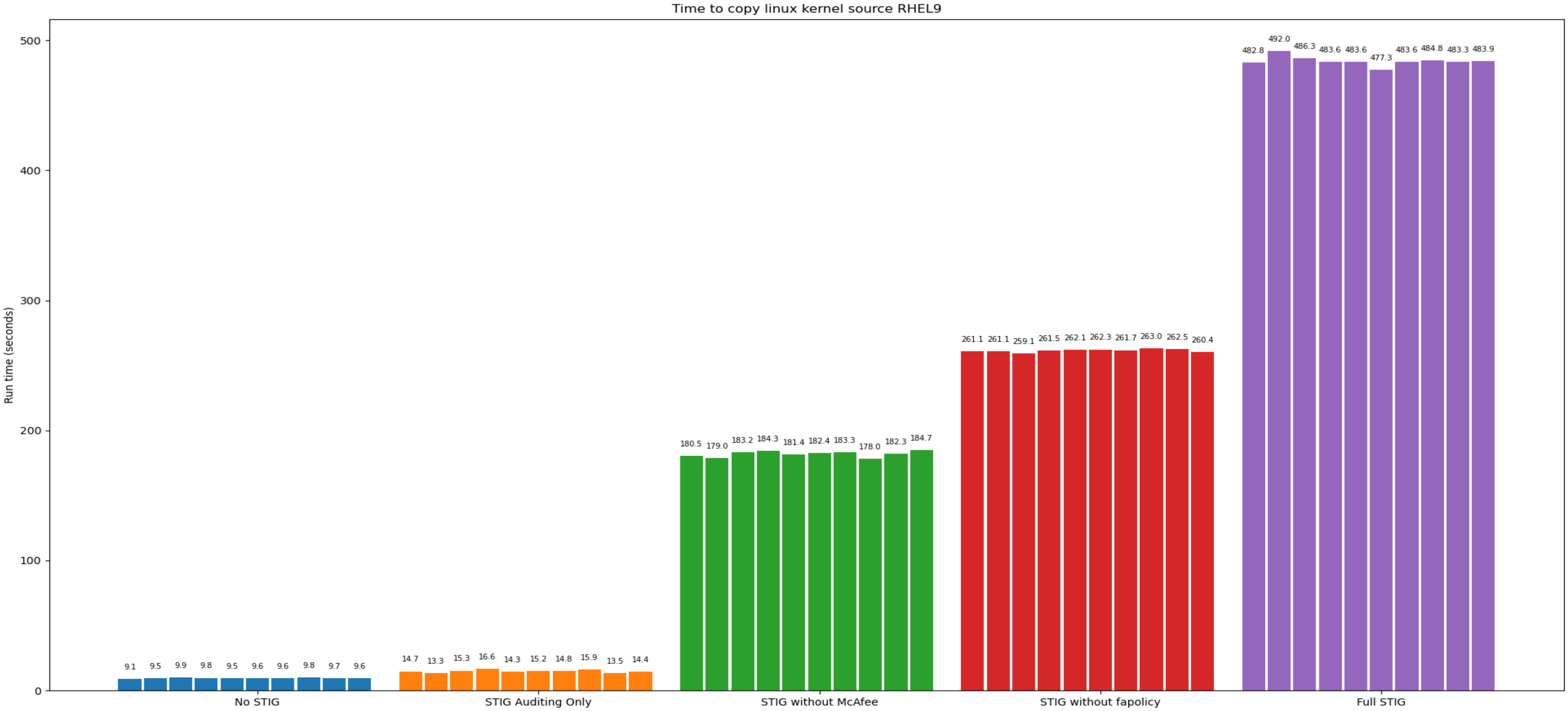
Single Copy Full STIG (without offload)



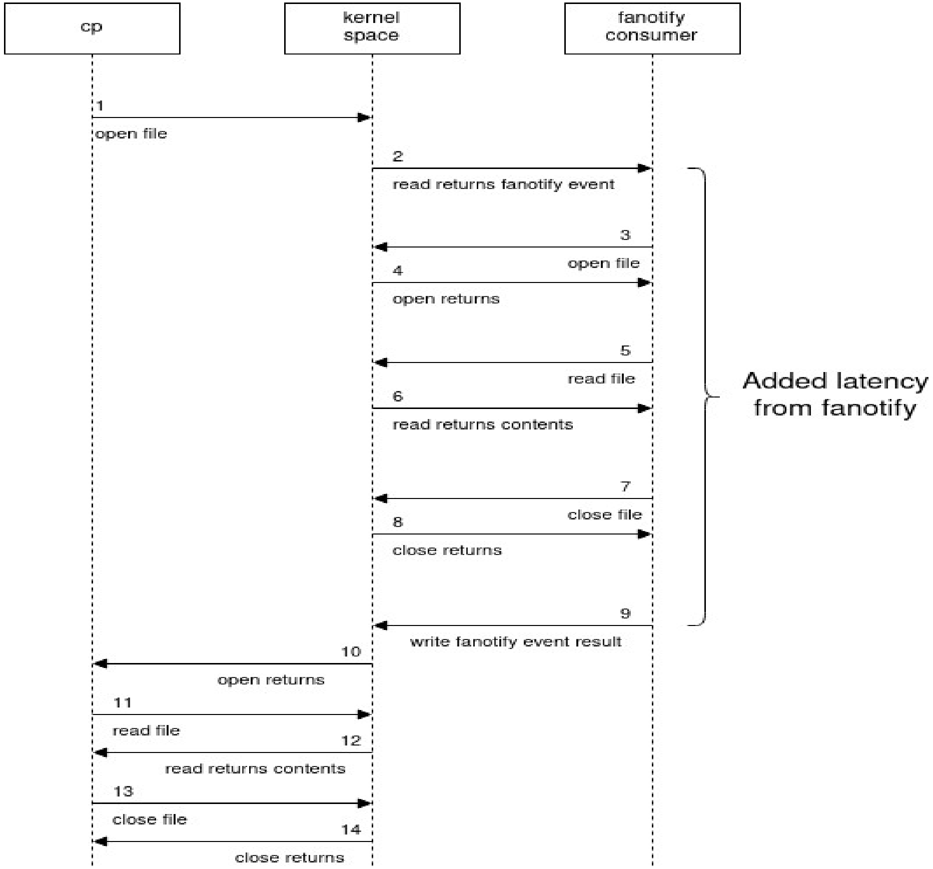
10 Runs (auditing impact)



10 Runs (fapolicy, McAfee impact)



fapolicy and HBSS problems



- Multiple fapolicy consumers
  - HBSS/McAfee/Trellix
    - Adds the most latency because on-demand scanning is being performed
  - fapolicyd
    - Default policy has several rules but is generally has less overhead then HBSS

Recommendations/Findings

- Reduce Auditing
- Don’t implement RHEL-09-652035
  - Use a different way to offload audit logs
- Don’t use fapolicy RHEL-09-433015
  - There are many trivial ways to bypass
  - Can’t ignore the cost
- Don’t use HBSS (McAfee/Trellix) RHEL-09-211025
  - Single biggest impact to system performance from the STIG
  - Also trivial to bypass
  - Keep systems up to date and run vetted software
  - Traditional AV is antiquated thinking

Recommendations/Findings

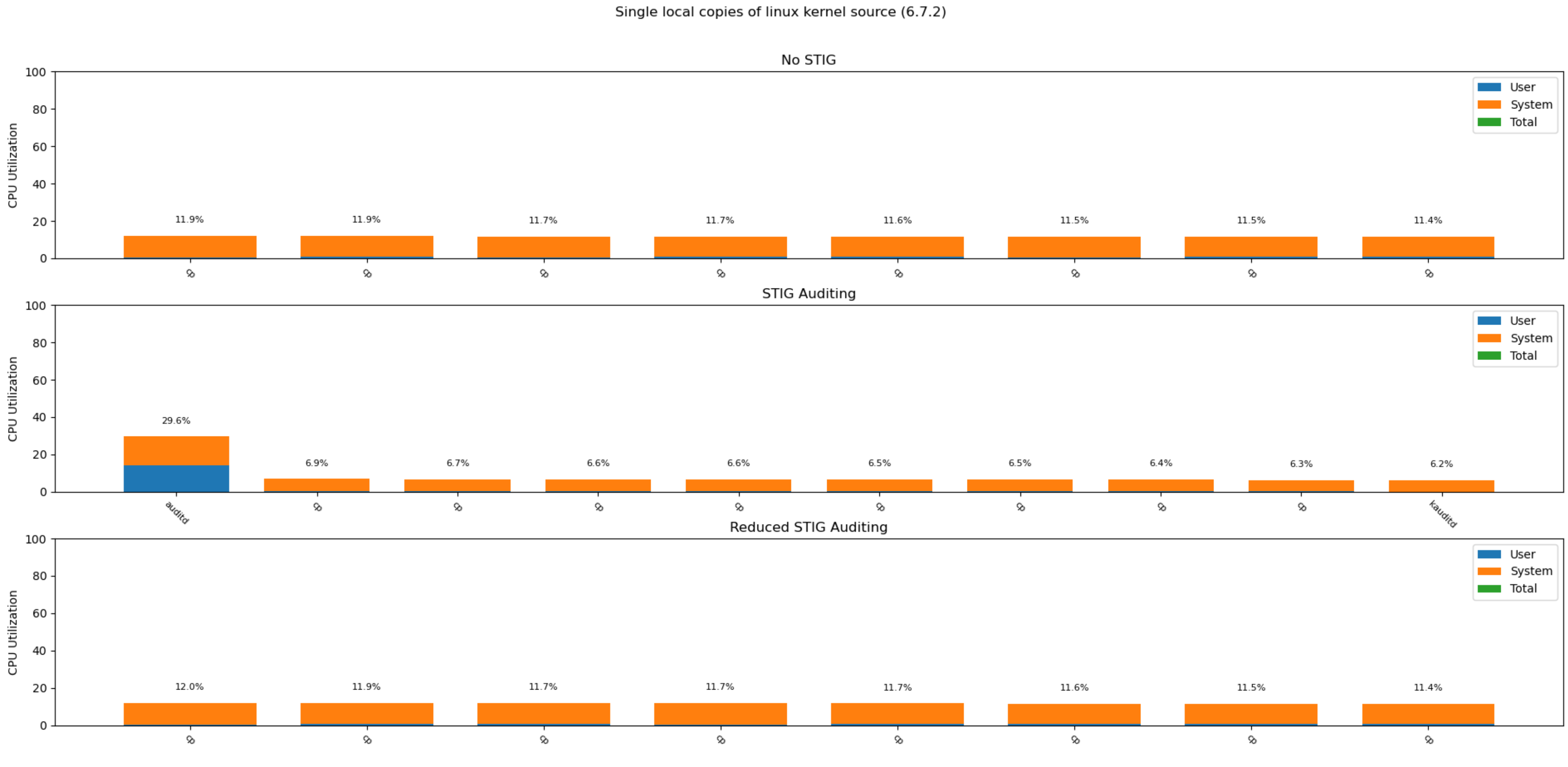
- Auditing in linux is a bottleneck
  - Reduce auditing by removing pointless item (next slides)
  - Auditing random DAC failures is just noise
- DO audit methods of privilege escalation
  - The STIG is incomplete here
  - Audit execution of all privileged binaries (capabilities and SUID/SGID)
  - Audit anything altering kernel modules
  - Even better use secure boot and kernel lockdown mode
- Use EDR (within reason) but only if it’s not a custom kernel modules
  - eBPF enables all the functionality a reasonable EDR should need
- Just because it’s in NIST SP 800-53 doesn’t mean its a good idea
  - Push back when it degrades your performance or usability

Reduced Auditing Rules

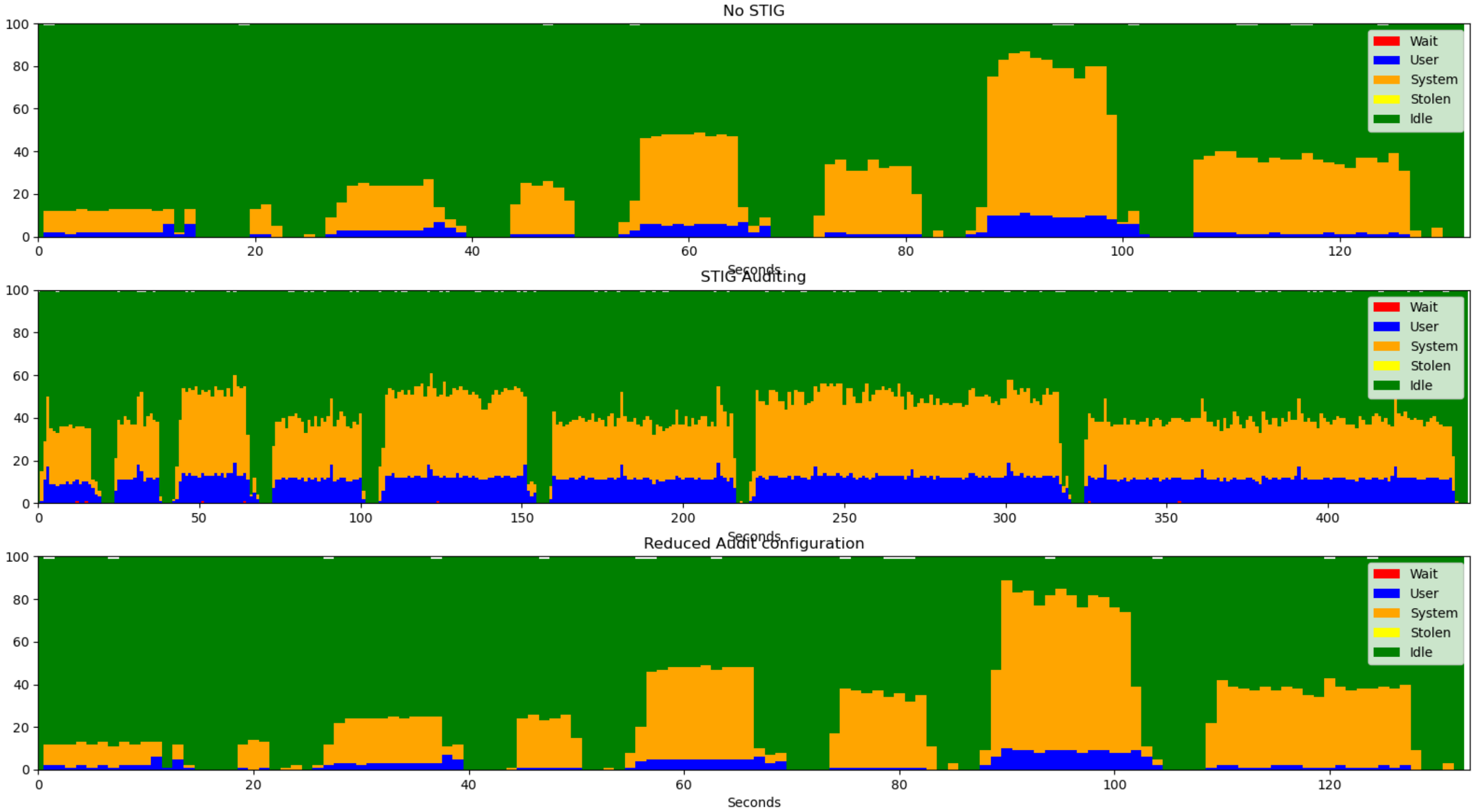
```
-a always,exit -F arch=b33 -S execve -C uid!=euid -F euid=0 -k execpriv
-a always,exit -F arch=b64 -S execve -C uid!=euid -F euid=0 -k execpriv
-a always,exit -F arch=b32 -S execve -C gid!=egid -F egid=0 -k execpriv
-a always,exit -F arch=b64 -S execve -C gid!=egid -F egid=0 -k execpriv
-a always,exit -F arch=b32 -S delete_module -F auid>=1000 -F auid!=unset -k module_chng
-a always,exit -F arch=b64 -S delete_module -F auid>=1000 -F auid!=unset -k module_chng
-a always,exit -F arch=b32 -S init_module,finit_module -F auid>=1000 -F auid!=unset -k module_chng
-a always,exit -F arch=b64 -S init_module,finit_module -F auid>=1000 -F auid!=unset -k module_chng
-a always,exit -F path=/usr/bin/umount -F perm=x -F auid>=1000 -F auid!=unset -k privileged-mount
-a always,exit -F path=/usr/bin/chacl -F perm=x -F auid>=1000 -F auid!=unset -k perm_mod
-a always,exit -F path=/usr/bin/setfacl -F perm=x -F auid>=1000 -F auid!=unset -k perm_mod
-a always,exit -F path=/usr/bin/chcon -F perm=x -F auid>=1000 -F auid!=unset -k perm_mod
-a always,exit -F path=/usr/sbin/semanage -F perm=x -F auid>=1000 -F auid!=unset -k privileged-unix-update
-a always,exit -F path=/usr/sbin/setfiles -F perm=x -F auid>=1000 -F auid!=unset -k privileged-unix-update
-a always,exit -F path=/usr/bin/chage -F perm=x -F auid>=1000 -F auid!=unset -k privileged-chage
-a always,exit -F path=/usr/bin/chsh -F perm=x -F auid>=1000 -F auid!=unset -k priv_cmd
-a always,exit -F path=/usr/bin/crontab -F perm=x -F auid>=1000 -F auid!=unset -k privileged-crontab
-a always,exit -F path=/usr/bin/gpasswd -F perm=x -F auid>=1000 -F auid!=unset -k privileged-gpasswd
-a always,exit -F path=/usr/bin/kmod -F perm=x -F auid>=1000 -F auid!=unset -k modules
-a always,exit -F path=/usr/bin/newgrp -F perm=x -F auid>=1000 -F auid!=unset -k priv_cmd
-a always,exit -F path=/usr/sbin/pam_timestamp_check -F perm=x -F auid>=1000 -F auid!=unset -k privileged-pam_timestamp_check
-a always,exit -F path=/usr/bin/passwd -F perm=x -F auid>=1000 -F auid!=unset -k privileged-passwd
-a always,exit -F path=/usr/sbin/postdrop -F perm=x -F auid>=1000 -F auid!=unset -k privileged-unix-update
-a always,exit -F path=/usr/sbin/postqueue -F perm=x -F auid>=1000 -F auid!=unset -k privileged-unix-update
-a always,exit -F path=/usr/bin/ssh-agent -F perm=x -F auid>=1000 -F auid!=unset -k privileged-ssh
-a always,exit -F path=/usr/libexec/openssh/ssh-keysign -F perm=x -F auid>=1000 -F auid!=unset -k privileged-ssh
-a always,exit -F path=/usr/bin/su -F perm=x -F auid>=1000 -F auid!=unset -k privileged-priv_change
-a always,exit -F path=/usr/bin/sudo -F perm=x -F auid>=1000 -F auid!=unset -k priv_cmd
-a always,exit -F path=/usr/bin/sudoedit -F perm=x -F auid>=1000 -F auid!=unset -k priv_cmd
-a always,exit -F path=/usr/sbin/unix_chkpwd -F perm=x -F auid>=1000 -F auid!=unset -k privileged-unix-update
-a always,exit -F path=/usr/sbin/unix_update -F perm=x -F auid>=1000 -F auid!=unset -k privileged-unix-update
-a always,exit -F path=/usr/sbin/userhelper -F perm=x -F auid>=1000 -F auid!=unset -k privileged-unix-update
-a always,exit -F path=/usr/sbin/usermod -F perm=x -F auid>=1000 -F auid!=unset -k privileged-usermod
-a always,exit -F path=/usr/bin/mount -F perm=x -F auid>=1000 -F auid!=unset -k privileged-mount
-a always,exit -F path=/usr/sbin/init -F perm=x -F auid>=1000 -F auid!=unset -k privileged-init
-a always,exit -F path=/usr/sbin/poweroff -F perm=x -F auid>=1000 -F auid!=unset -k privileged-poweroff
-a always,exit -F path=/usr/sbin/reboot -F perm=x -F auid>=1000 -F auid!=unset -k privileged-reboot
-a always,exit -F path=/usr/sbin/shutdown -F perm=x -F auid>=1000 -F auid!=unset -k privileged-shutdown
-a always,exit -F arch=b32 -S umount -F auid>=1000 -F auid!=unset -k privileged-umount
```

```
-w /etc/sudoers -p wa -k identity
-w /etc/sudoers.d/ -p wa -k identity
-w /etc/group -p wa -k identity
-w /etc/gshadow -p wa -k identity
-w /etc/security/opasswd -p wa -k identity
-w /etc/passwd -p wa -k identity
-w /etc/shadow -p wa -k identity
-w /var/log/faillock -p wa -k logins
-w /var/log/lastlog -p wa -k logins
-f 2
-e 2
--loginuid-immutable
```

Reduced Auditing Impact



Reduced Auditing Impact



Future work / Other issues

- There are plenty of issues I haven’t touched on
  - Not directly performance related
  - Captured in other documents
- Working on approval to release ansible playbook for those interested
- Working on a set of eBPF/bpftrace one liners to characterize the performance impacts

Conclusion

Don’t let bad “security” recommendations destroy your system’s performance