NAME

cap-audit − discover the minimal capability set for a program

SYNOPSIS

cap-audit [*options*] **--** *command* [*args*...]

DESCRIPTION

cap-audit traces a target process and its descendants to record every kernel capability check it performs. The tool uses eBPF to hook capability verification paths in the kernel and libcap-ng to format the results. Events are filtered to only the traced process tree to reduce overhead.

The auditor fork/execs the requested command, registers its PID for tracing before exec, and automatically tracks child PIDs. At the end of execution it reports which capabilities were granted, which were attempted and denied, and presents deployment snippets showing how to grant the minimal set.

Runtime tracing requires root or the CAP_BPF, CAP_PERFMON, and CAP_SYS_PTRACE capabilities, along with kernel BTF data in */sys/kernel/btf/vmlinux*.

OPTIONS

-h, **--help**
: Show a help message and exit.

-v, **--verbose**
: Print each capability check as it occurs.

-j, **--json**
: Emit the analysis as JSON.

-y, **--yaml**
: Emit the analysis as YAML.

--
: End option parsing and treat the rest of the command line as the program to execute under audit.

EXIT STATUS

The program returns 0 on success. Non-zero values indicate an error occurred while setting up tracing or executing the target command.

NOTES

1. This utility observes the application behavior to decide what is needed. If you do not exercise all functionality, it can result in an incomplete inventory of needed capabilities. If it can’t see it, cap-audit can’t record it.

2. There are hard memory constraints for event collection. The limit for the number of children that can be traced is 8192. If an application is fork heavy, such as shell scripts or spawns a lot of short lived helper applications, the tracer will stop collecting events on new children once the buffer fills. Also, the tracer only has space to hold 20k unique stack traces. This is not as problematic as the other constraint as the same code paths often repeat. If this fills up, new code paths will not be collected. This buffer is used to map a capability request to the syscall needing it. Knowing the syscall causing the needed capability is a "nice to have". What’s more important is the capability needed - which is in a different memory area and safe from the described issue. In both cases, there is no indication to user space that the tracer is out of memory.

3. When auditing a daemon, pass options to keep it in the foreground. If it forks and exits, that will end the session and you will only get the capabilities traced up to the exit. Here’s an example keeping the daemon in the foreground:

cap-audit /usr/sbin/sshd -D

EXPERIMENTAL: cap-audit output is experimental, but very close.

FILES

/sys/kernel/btf/vmlinux

SEE ALSO

capabilities(7), **filecap**(8).

AUTHOR

Steve Grubb