**Detector:**

**Introduction:**

Tag-based detection uses tags to identify processes that have suspicious performances by analyzing system audit logs. Such a procedure includes three main steps: tag initialization, tag propagation(transfer), suspicious process judgment.

At the tag initialization stage, processes or files that are potentially related to an attack scenario will be labeled specific tags based on pattern matching.

At the tag propagation stage, processes or files that have interaction with tagged processes or files will be labeled specific tags based on pattern matching.

At the suspicious process judgment stage, processes that are labeled certain combinations of tags will be recorded as suspicious processes.

**Contribution:**

**1. Implementing detecting points for 7 attack scenarios (download and execution, shell scripting, sourcing, local job scheduling, trap, bash history, credentials in files) on the detector.**

a) download and execution

Processes that download files from the Internet and then execute them.

b) Scripting

Processes that use “sh” command to execute suspicious script files.

c) Sourcing

Processes that use “source” command to load suspicious files.

d) Local job scheduling

Processes that schedule local jobs that execute or load suspicious files.

e) Trap

Processes that schedule which execute or load suspicious files jobs on interrupt signals.

f) Bash history

Processes that read bash history.

g) Credentials in files

Processes that search files for credentials.

Those 7 detecting points successfully detected malicious process behaviors that existed in testing log data.

**2. Changing the implementation of current Linux detector on logic level which allows it to detect various suspicious behaviors for any process instead of only one suspicious behavior for each process.**

**3. Transfer the “Drawing attack scenario graph” feature from Windows Detector to the current Linux detector. Such a procedure extracts all related events for suspicious attack scenarios and those extracted events are then used to draw attack scenario graphs.**

**Collector:**

**Introduction:**

LTTng, *Linux Trace Toolkit: next generation, is a* light-weight tracing framework for Linux. Log data collected by LTTng is small in size and can be used for real-time attack detection.

Since the system log collected by LTTng is a little different from the one collected by SPADE, in case of format, event names and parameters, we need to make some mappings to fit it to the original data parser on our Linux detector.

**Contribution:**

**1. Implemented a data parser that formalizing LTTng log into a SPADE-like vertex-edge log.**

**2. Found the mapping from file descriptor to file path for some important events like File\_Read, File\_Write, etc.**

**3. Made a table for socket related tracepoints that listed all their parameters.**

**4. Found the mapping from (INET/INET6) socket file descriptor to IPv4/IPv6 & port for both itself and the peer socket.**

Other works:

1. did research on lttng-analyses.

2. did research on LTTng syscalls and tracepoints.