RWS-IDS: Read/Write-Sensitive HIDS for Container Runtime Security

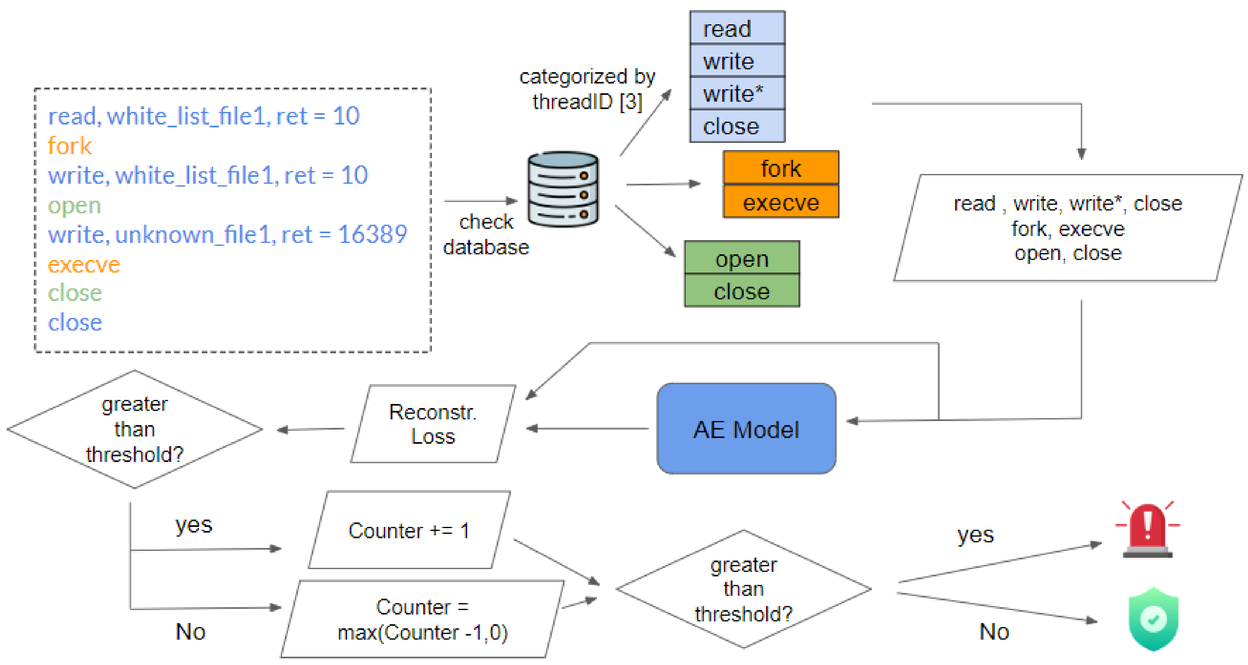
Graduate Degree Program of Cyber Security
National Yang Ming Chiao Tung University

Student: Wen-Sheng Lo Advisor: Tsern-Hui Lee

Abstract

Many recent papers have shown that using system call arguments can help to improve the performance of system call-based HIDS. Therefore, we use the latest LID-DS dataset containing system call arguments to conduct the experiments. In response to the current trend of containerization in the industry, we also use the container-supported tools - Sysdig to monitor applications running on containers and collect the corresponding system call data. Finally, our system achieves the best performance on the LID-DS dataset: 98.7 % detection rate and 0.15% false positive rate on average, and it also successfully detected the recently most serious Log4Shell vulnerability.

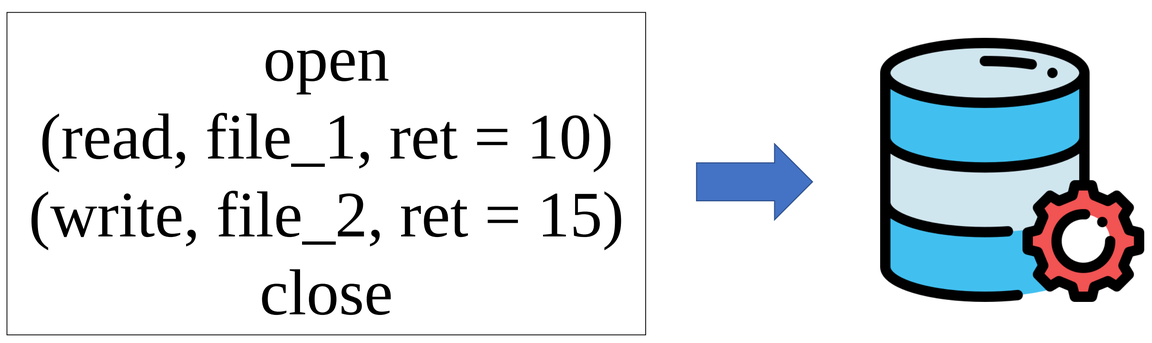
System architecture



Method

While preprocessing training data, build a syscall database

- For syscall read/write
  - Store its sysname + file_name + return_value into database
- For other syscall
  - Only store its sysname into database



While preprocessing testing data

- For syscall read/write
  - Check if its sysname + file_name + return_value is in database, if not, label it as unknown read/write, otherwise, label it as read/write
- For other syscall
  - Check if its sysname in database, if not, label it as unknown syscall, otherwise, label it as its sysname

Conclusion

The arguments of syscall are rarely used in the previous syscall-based HIDS research. To detect the above-mentioned two attack scenarios in LID-DS, we proposed a read/write-sensitive HIDS to mitigate the problem. It cannot only successfully detect the two attacks but also perform well on the other attack scenarios. our proposed system achieves a state-of-the-art result on the latest HIDS dataset - LID-DS and successfully detects the most serious vulnerability - Log4j recently. For practical use, the future work includes testing our proposed system on a server with real-world traffic of ‘read’/’write’ operations and combining our system with the container orchestration tools (e.g., k8s).