FINAL REPORT

This Objectives of this script were to Identify malicious logs, Monitor system performance, send email alerts using Gmail’s SMTP server, Perform Network scans and Capture Network packets/ We use code in this manner in many aspects of our work, such as automating the study of system logs, monitoring system activity performance, implementing network security features that are positioned in between (a threat detection approach), and maintaining system stability through quick response times. Based on the system that has been put into place, this article outlines the goals, methods, outcomes, difficulties, and next work. Examining system logs to look for unusual activity, keeping an eye on CPU and memory utilization, warning of excessive CPU usage, scanning the network for verified vulnerabilities, and performing logs are all done with this Python script.

We start our analysis by looking at the system log file Linux_2k.log. In order to put the results in a summary report named summary_report.txt, the Python script searches for rows that contain the words "failure," "unknown," or other similar terms. In this instance, the psutil library is used to monitor the CPU and RAM by printing and logging performance metrics. It uses hardcoded credentials to send an email alert via Gmail's SMTP server when CPU use above 2%.

We begin our analysis by checking out Linux_2k.log, a system log file. The python script looks for rows containing "failure" or "unknown" or some other words and stores them into a summary report called summary_report. txt. In this case, CPU and memory monitoring is done with the help of psutil library which prints/logs the performance data. When the CPU usage exceeds 2%, it sends an alert through an e-mail via Gmail's SMTP server with hardcoded credentials.

The review is predicated on records from the Linux_2k.log system log file. The comments are generated into a summary file called summary_report.txt by the Python script, which searches for occurrences of the classes of such phrases (such as "failure" or "unknown"). Performance data is logged using the psutil package, which is used for CPU and memory monitoring. The Gmail SMTP server is then used with hardcoded credentials to send an email alert.

When network traffic is being observed, scapy examines any packet data, logs its IP address information, and provides network activity. Python's subprocess features allow the use of Nmap on network scans. Please take note that these features have proven quite helpful in monitoring hardware performance, cross-checking logs for errors, and sending out alarms when RAM spikes occur. The examination of anomalous traffic patterns can be better understood by observing the open ports, which are checked using Nmap scan and TCP packet listening.

However, there are issues with the system. For instance, its correlation analysis could miss novel attack types that static rules are unable to identify, even when it is perfect for well-known

patterns and common danger indicators. Additionally, limitations like keyword patterns for log identification and simple packet sniffer can make it less effective.

A number of significant obstacles stand in the way of these real-world difficulties. Because static log analysis depends on pattern-based logging, it may miss abnormalities that don't match recognized markers. When issuing CPU usage alerts, there may be an excessive number of false positives since the threshold may be set too low or too sensitively. Additionally, the script hardcodes the email credentials, which could pose a security concern in the event that they leak.

The system's flexibility is thus further constrained by reliance on certain tools like Nmap or the restricted capabilities of packet sniffing. Combining machine learning and dynamic log analysis with improved log processing and observability that could track other metrics like disk or bandwidth use could solve these issues.

[If you want a recommendation: Implement encryption or secure storage mechanisms for sensitive credentials to reduce the risk.] The system could also be improved by enabling admin to define a custom range for the network scan, a database with structured reports to make sense of the aggregated data, management, and improving packet sniffing by adding PCAP support for easier analysis with other forensic tools. Also, visualization tools like Grafana or Kibana can create nice dashboards where the user can see live trends and patterns in the performance of the system and network activity.

One way to mitigate the current system’s shortcomings would be to bring modularity into its design. This would allow us to add future capabilities like intrusion detection or even advanced scan features without the need for changes to the existing system. Training as well as reaction

testing. This would give practical expertise with threat detection methods and network security technologies.

By integrating log analysis, performance tracking and alerting capabilities (this) script creates a solid groundwork for monitoring both systems and networks. However, while these features effectively identify threats and manage system health, there is still room for enhancement. For example, the application of machine learning to analyze logs adaptively, the implementation of advanced packet sniffing techniques and the bolstering of security measures (to safeguard sensitive login information) represent valuable avenues for development. Although these enhancements are noteworthy, they could ultimately elevate the system to a cutting-edge cybersecurity solution. This transformation would offer in-depth insights into network dynamics, respond to emerging threats and ensure ongoing monitoring of system health.