OPEN RESOURCE CONTROL AND SURVEILLANCE TOOL

Project Reference No.: *45S_BE_3868*

College ***:*** *K.L.S. Gogte Institute of Technology, Udyambag, Belagaavi*
Branch ***:*** *Department of Computer Science and Engineering*
Guide(s) ***:*** *Dr. Umesh M. Kulkarni*
Student(S) ***:*** *Mr. Shreyas Shivakumar*
Mr. Sharan Jamanani
Mr. Shreyas Kapale
Mr. Sahil Shet

Keywords: Real time Resource Monitoring and Control, Resource Surveillance, IAM, Identity access management, Cross-platform, App banning, Node-exporter, System metrics, Fine-grained control.

Introduction:

With the number of computer and expensive cloud droplets used in organizations, it is becoming increasingly important to ensure that these resources aren't being exploited or misused in any way and to ensure data security across systems to protect sensitive data. There is a huge requirement in the direct consumer space market for tools that provide smart fine grain control over system resources, which is not available in present tools. Most tools are focussed on resource control over virtual machines, DevOps and cloud computing systems, and not general physical systems which are used in institutes like schools, colleges, companies etc.

A tool is required which can both smartly provide system metrics and has a IAM policy-based grouping, monitoring and controlling. A decision was made to create an open resource control and surveillance tool (ORCS) to fulfil the aforementioned requirements. ORCS is a robust toolkit meant for real time monitoring of statistics of the Memory and CPU utilization of the system and various background processes running in the system. ORCS also comes with a powerful feature of letting the admin define user groups and perform app banning according to different user group policies. ORCS is a truly multi-platform toolkit and can work seamlessly on Operating systems like Windows, MacOS, Linux as well as monitor cloud droplets.

This makes ORCS a great toolkit in the field of Computer and network security as it not only monitors the system statistics but also allows the admin to block certain applications which could be a threat to data security or could run in the background and compromise system security to do irreversible damage.

The market has a few popular products that do similar things. AWS CloudWatch is a monitoring and observability service built for DevOps engineers, developers, site reliability engineers, IT managers, and product owners. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events. Its major drawback is the fact that it only monitors AWS cloud droplets.

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. Prometheus works well for recording any purely numeric time series. It fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures. Drawbacks of using Prometheus include a Subpar UI and lack of long-term storage. ORCS was conceived and designed to overcome these drawbacks.

Objectives:
The objectives of this project are to develop a toolkit named as "Open Resource Control and Surveillance Toolkit" that displays the real-time statistics of the Memory and CPU utilization of the system and various background processes running in the system. The toolkit must be capable of restricting Apps on the system according to policies defined by the admin for different user groups through Identity access management.

The master server should be flexible to add systems and display separate dashboards for each system connected to it. Toolkit should have a feature to add/modify multiple policies in order to have higher scalability. The toolkit must be able to perform all the said tasks on Windows, MacOS and Linux. ORCS should be able to monitor the usage statistics of computer hardware and cloud droplets, thus overcoming one of the shortcomings of several similar products in the market.

ORCS admin should be able to create users and policies, thus ensuring a better and appropriate use of the resource. ORCS should be able to provide a graphical representation of active memory utilization for better understanding and easier data interpretation.

Methodology:
A thorough study was conducted to find out about the existing solutions in the market that perform similar tasks. Some of these products were actually used to understand their features firsthand and to try and identify their drawbacks. Deliberations were held to decide the project's problem statement and basic functional and non-functional requirements were established. Features such as app banning and IAM access were penciled in for implementation.

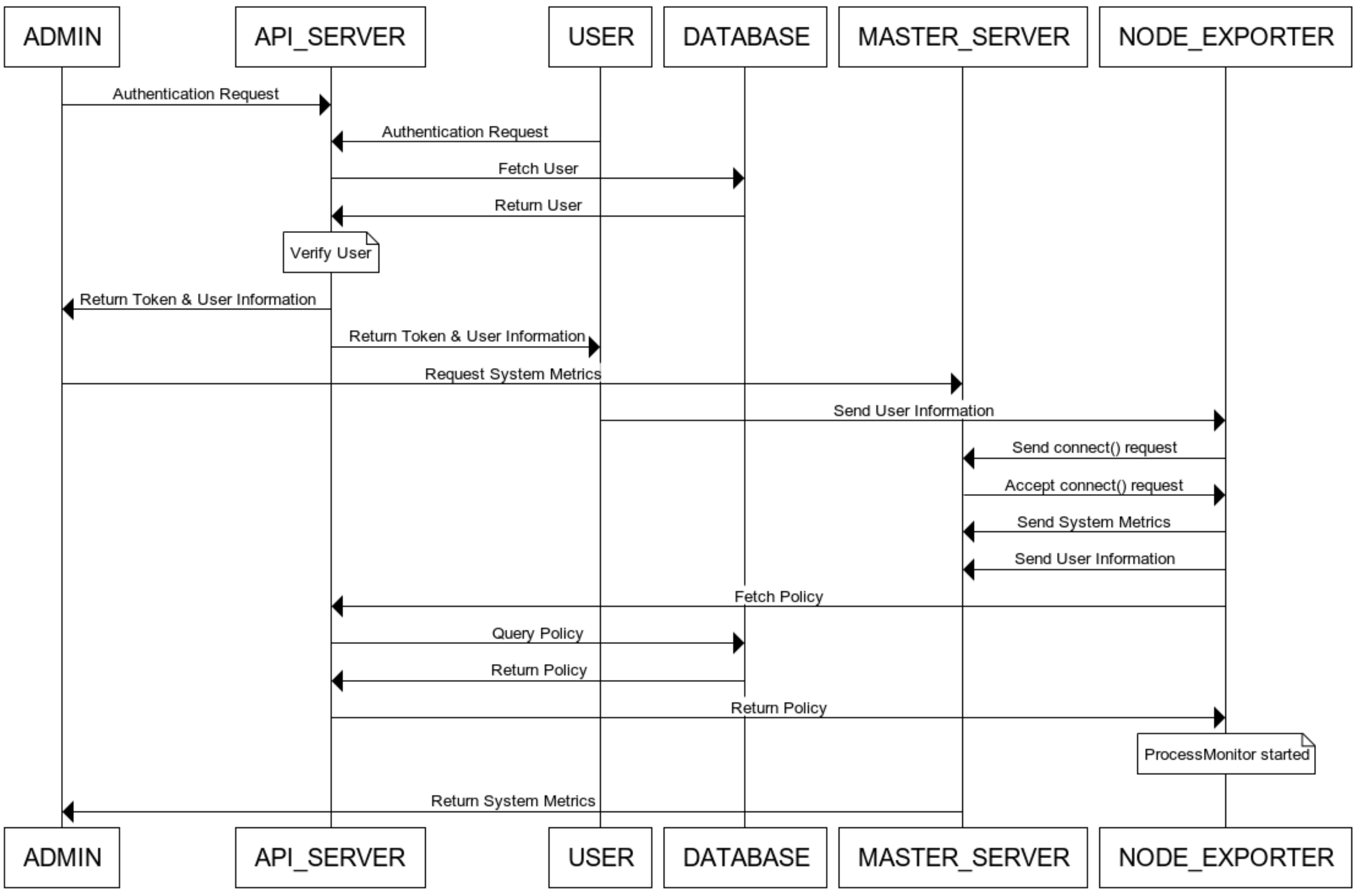
Upon setting up the problem statement and requirements, a basic structure of the product was designed along with important software artifacts to make the development process much easier and to support it. The artifacts were further enhanced through the guidance of the mentor to make them more comprehensive.

Upon successfully designing all the software artifacts such as ER diagrams, DFSs, Use case diagrams, sequence diagrams etc they were validated. A robust tech stack was chosen for the development in order to make the toolkit a multi-platform, scalable and fast application.

An ambitious project deadline was set with meaningful phases and the timeline was strictly followed to ensure that the toolkit is developed within the right time frame to allow a good amount of testing and bug fixing time.

In the initial phase of development, the basic functionality to monitor a system was implemented with an easy-to-understand user interface. This was done with close monitoring and guidance of mentors in order to ensure good programming practices and avoid issues right at the start. Every phase of code writing was accompanied by careful and comprehensive documentation of the same in order to closely monitor the progress. Subsequent phases saw the implementation and integration of IAM, User Authentication, Policy and user group creation and app banning with the initial system to make ORCS and robust System security toolkit.

Upon the completion of the coding phase of the project, rigorous testing was carried out to test the feature to ensure industry standards and validate the output of the toolkit. Testing was carried out on all the operating systems to ensure cross-platform support. Furthermore, each phase of coding and testing was documented to keep tabs on the development and avoid slip-ups. The project was made while following all the right coding practices and principles of object-oriented development.



(ORCS Sequence diagram)

Results and Conclusion

Upon the completion of the implementation of the toolkit, we were able to achieve all the goals and targets we set out to achieve. The application worked seamlessly on all three test operating systems (Windows, MacOS and Linux). We were able to create all the necessary UI elements to integrate them with our features and make the application much easier to use and understand. The application exhibited the below-mentioned behaviour, which was the expected and intended output.

- The application was able to monitor all the important system information such as no of active processes, total memory, used memory, CPU temperature, etc
- The information was displayed in an easy-to-interpret dashboard
- The toolkit has features to add new users, create new policies for users, ban applications, notify users about any banning, etc.

We can conclude that the said targets can be achieved satisfactorily using knowledge of core Computer science principles and good programming practices. We also conclude that it is necessary to have a toolkit like ORCS running on a system in order to ensure appropriate usage, prevent exploits and have fine-grain control.

Scope for future work

With its scalability and the fact that ORCS uses a robust tech stack, and has good scope for future work and development, the ones achievable in near future are:

- Creating alerts on the minimum threshold of the system's computing resources
- Bulk user registration through spreadsheets
- Code optimization for efficient consumption of system resources
- Mobility solutions for the admin, like an SMS or a mobile app for getting updates anywhere
- Failsafe mechanisms, like daemon process to restart the server when stopped, also to notify the admin when a process fails due to unknown circumstances.