iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78032

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Real-time Sensor Access Security Monitor Using Windows Registry Analysis

Gopi Vignesh[1], Kiruba M[2], Lakshana K[3], Hirthiha V[4]
[1]*Assistant Professor / CSE*
[2, 3, 4]*Department of Computer Science and Engineering, SSM Institute of engineering, Dindigul, India*

Abstract: ***Think of a quiet guard always watching the computer's eyes and ears, the camera and microphone. This project builds exactly that: a live monitoring system for Windows that keeps constant track of which programs try to use those sensors. It works by digging into the computer's own registry, the digital logbook where access attempts get recorded. A small widget on the desktop instantly changes color, flashing a silent warning. At the same moment, a WhatsApp message shoots straight to the user's phone. To avoid constant alarms for normal use, trusted apps like Zoom or Skype go on a friendly list, a whitelist that tells the system to stand down. Need to turn the guard off from another room? A simple, lightweight web server runs locally, letting anyone start or stop the whole thing from a phone or another computer's browser. In a world where digital privacy feels thinner every day, this tool offers a straightforward, ever-present watch over the most personal hardware. It runs quietly in the background, barely touching the computer's performance, a persistent shield against spyware and unwanted snooping.***
Keywords: Privacy Guard, Live Sensor Watch, WhatsApp Alerts, Windows Security, Camera & Mic Control, Desktop Status Widget

I. INTRODUCTION

Those little cameras and microphones on our laptops? They are the front door to our digital lives. We need them for work calls and family chats, but they are also a perfect, silent entry point for anyone, or any piece of software, that wants to peek inside. Your standard antivirus wo not notice if a program turns your webcam on. It's a security blind spot, where the most private moments in a room can be seen or heard without a trace.

This project built a simple guard for that front door on Windows computers. It watches the computer's own activity ledger, and the second an uninvited program tries to use the camera or mic, it acts. A small square on your screen flashes red, and your phone buzzes with a WhatsApp alert, a warning sent from the computer itself. You can tell the guard who the regular visitors are, like Zoom or Skype, so it does not alarm you during a meeting. You can even pause the whole watch from your phone's browser.

The point was to end the guessing game. Instead of wondering if your device is listening, you get a quiet, constant signal. It turns a hidden vulnerability into something you can see and understand, putting a clear pane of glass in a door that used to feel wide open.

II. LITERATURE REVIEW AND BACKGROUND

The journey to guard a computer's camera and microphone began when we realized the danger had moved indoors. Early computer security acted like a town guard at the main gate, watching for outsiders. But the threat is not from beyond the walls anymore. It's from any program already inside that can quietly switch on those little sensors without making a sound.

For Windows machines, the solution lies in the system's own diary, the Registry. It's a faithful, ever-changing log where the computer notes down which app just asked to use the webcam. Watching this log is steady and reliable, unlike more invasive methods that can make the whole system unstable with a single update. Other attempts have been too clumsy or too fragile, either burying the important detail in a mountain of data or breaking every time the operating system changes.

Phones solved this by asking for permission every single time, but on a desktop, that habit never took hold. People click "allow" once and the door stays open forever. The trick, then, is to build a quiet reminder, a small, colored light on the screen that's always there, showing a green "all clear" or a red "warning." And when something truly wrong happens, the alarm should not stay on the computer. It should travel straight to the phone in your pocket, using something you already check constantly, like WhatsApp.

That's what this project built: a sentry made from reliable parts. It watches the Registry log, keeps a quiet eye on the desktop, and shouts an alert to your mobile device when needed. It learns which apps you trust for video calls, and you can even pause it from a webpage on your phone. The goal was never to reinvent the wheel, but to assemble a simple, steadfast watcher for the two most personal windows into your digital life.

©IJRASET: All Rights are Reserved

A. Related Works/Approaches

The journey to protect our devices' eyes and ears has taken many roads. Early researchers played the role of ethical burglars, creating tools with names like "CamSnarf" to pick the digital lock on webcams. They published their methods not to cause harm, but to sound a loud alarm about how fragile our privacy really was. Their work made the threat impossible to ignore and pushed others to build digital guards.

Those guards were built with different tools. One group of developers reached for the heaviest wrenches in the toolbox, working at the very core of the operating system. Projects like "Camtective" tapped directly into the camera's driver, the fundamental software that lets the hardware work. This approach is powerful, it sees everything, but it is also delicate. A simple update from Microsoft can break the connection, leaving the user unprotected until a fix is released.

Other builders took a simpler approach. They noticed that large security suites from companies like Norton sometimes included a basic camera monitor as one small feature among many. Smaller programs focused on stopping keyloggers might add a tiny camera icon to the taskbar. And while excellent tools like browser extensions stopped ads and trackers online, they did nothing about the physical camera lens on the laptop lid. Each solution addressed a piece of the puzzle, but rarely the whole picture.

This project tried to learn from all of them. It avoids the complexity of deep-system tinkering, which can be unstable. Instead, it finds a reliable vantage point within Windows itself to watch for activity. For sounding the alarm, it makes a practical choice. Instead of building a new siren from scratch, it sends a message through WhatsApp, a service already open on people's phones throughout the day. It turns a common app into a direct privacy alert.

B. Existing System

Anyone searching for a way to guard their webcam today will find their options are limited to a few frustrating types.

First, there are the basic controls that come with the computer. In Windows, a person can dive into the settings menu and flip the camera and microphone to "off." It is a blunt instrument. It prevents access but offers no intelligence. The user receives no record of what tried to turn it on, no history of access attempts, and certainly no immediate warning. It is a deadbolt, not a security system.

Second are the famous, all-in-one security packages. Brands like McAfee and Bitdefender promise "camera protection." What this often means is their antivirus scanner, which runs periodically, checks for known malware that targets cameras. It is a background check for bad software, not a live bodyguard for the hardware. A legitimate program or a new piece of spyware can activate the sensor between scans without raising any flag.

Third are the niche, dedicated programs. These are applications built for one job: watching the camera and microphone. Names like "OverSight" or "CameraGuard" appear in search results. These tools do provide the constant watchfulness the other options lack. However, they come with strings attached. Many were designed first for Apple computers, leaving Windows users with clunky ports. Others hide advanced settings behind confusing menus or require a yearly fee. Most critically, their warnings are trapped on the laptop screen. If the user is away, the alert flashes into an empty room. The idea of pausing the guard from a phone or another computer simply does not exist in these tools.

C. Problems Identified

The biggest problem with these monitoring tools is where they shout. They only make noise on the laptop they are supposed to be guarding. Step away to make coffee or close the lid, and the alarm rings in an empty room. A spyware camera could be live for an entire hour, and the only record would be a line in a text file you'll probably never open. A useful alert has to follow you into the kitchen, not sit patiently at your desk.

They're also terrible with the software you actually use every day. Most tools see programs as either completely safe or definite threats. Real life isn't like that. You trust Slack for team calls, but that doesn't mean it should have 24/7 access to your microphone. What's missing is a basic approved list, a way to give your everyday apps a key while still noting in the logbook every time they come and go.

On top of that, the tools themselves get in the way. They're built for people who love data, not for people who just want a simple answer. They show raw system logs and process IDs instead of a clear "on" or "off" light. And to do their job, many of them dig so deep into the computer's brain that the fan spins up and the battery drains. A security tool shouldn't make your computer so slow and hot that you're forced to turn it off.

Fundamentally, they're historians, not bodyguards. Their standard move is to spot an intrusion, write down the time and details, and call it a day. It's like having a security camera that faithfully records a break-in but has no connection to an alarm company or the police. It captures the crime perfectly for the news report tomorrow but does nothing to stop it tonight.

This leaves you with the worst possible outcome: knowledge without power. You get a perfect receipt for a privacy violation that already happened. The tool confirms your fears after the fact but gives you no lever to pull in the moment. Instead of a shield, it provides an autopsy report, a detailed document proving your privacy was broken, but no way to have kept it safe.

III. PROPOSED SYSTEM

The system acts like a watchful neighbor for a computer's camera and microphone. Instead of hiding in complex code, it provides a simple, immediate signal the moment something tries to use those sensors, ensuring that signal reaches you anywhere.

The method is straightforward. It keeps a constant eye on the Windows Registry, the system's own ledger. Whenever a program requests access to the webcam or microphone, Windows records it there. By watching this logbook, the system sees the activity in real time without slowing anything down or installing intrusive drivers.

If an unrecognized program tries to access a sensor, the response is instant. A small square on the desktop flips from green to red. Simultaneously, a brief WhatsApp message is sent to the user's phone, putting the warning directly in their pocket. To avoid constant alarms, trusted apps like Zoom can be added to a simple text-file whitelist, allowing them to work without triggering mobile alerts.

Control is also designed for convenience. The tool runs a small, local web server. From any device on the same Wi-Fi network, a user can open a browser to a private dashboard. With one click, they can start or stop the entire monitoring system from the couch or the kitchen, without needing to touch the computer itself.

A. Working Datasets

The system runs in a continuous, quiet loop. It wakes up every second to check the specific Registry keys for the camera and microphone. This check is very light, like glancing at a clock. It takes almost no processing power. If it sees that the status has changed from “not in use” to “in use,” it snaps to attention.

First, it identifies which program caused the change. It then takes that name and compares it line-by-line against the user’s whitelist file. Is the program on the list? If yes, it makes a dated note in a local log and goes back to watching. If the program is not on the list, the system triggers the alert sequence. It sends a command to change the desktop widget’s color and uses a small Python script to push a message through WhatsApp Web. This entire process, from detection to alert, takes less than two seconds.

Table I. Description of Sensor Access Monitoring Dataset Attributes

Column Name	Description
ApplicationName	Primary accessing application (e.g., chrome.exe, zoom.exe, teams.exe, unknown_process.exe)
SensorType	Type of hardware accessed (Webcam, Microphone, or Both)
LastUsedTimeStop	Registry value indicating active usage (0 = In Use, 1 = Not In Use)
ConsentValue	User permission state from consent store (Allowed, Denied, or Not Decided)
PackageFamilyName	Application package identifier for packaged apps
NonPackagedPath	File system path for non-packaged/desktop applications
LastUsedTimeStart	Timestamp of when access began
TrustStatus	Classification based on keyword heuristics (Trusted, Untrusted, or Unknown)

The system relies on two key pieces of information, both stored locally on the computer. The first is the live data from the Windows Registry. The system does not need the entire Registry. It focuses only on two specific addresses, which act like dedicated sign-in sheets for the webcam and the microphone. It reads only the "LastUsedTimeStop" and "LastUsedApplication" values from these sheets to know what happened and who did it. The second is the user's whitelist file. This is a simple .txt file that sits in the same folder as the monitoring tool. The user can open it with Notepad and edit it. Each line contains the filename of a trusted application. This file is the system's rulebook for distinguishing between friend and possible foe. Everything the system does is recorded in its own log file, also a plain text document in the same folder. It writes down the timestamp, the application name, and the action taken. There is no cloud database, no internet server, and no external data collection. All the information stays on the computer, belonging entirely to the person using it. The only external communication is the single outbound WhatsApp message when a true alert is necessary.

IV. SIMULATION SETTINGS

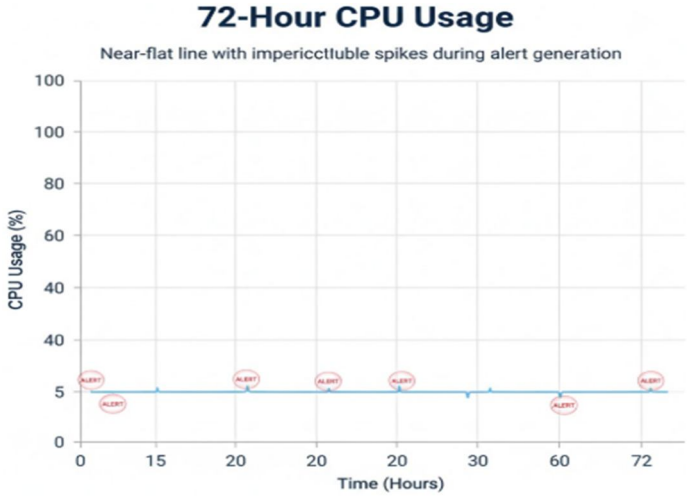
Think of testing this system like running a fire drill. You need to know the alarm works without setting anything on fire. To safely create that test, the project uses a familiar tool, the Google Chrome browser, and pretends it's an untrusted program trying to sneak a look through the camera. The test kicks off naturally. Opening a video call on Google Meet or visiting a site that asks for microphone access is all it takes. Chrome reaching for the sensor is the equivalent of pulling the fire alarm handle. The monitoring system, which is always watching the computer's internal activity log, spots this immediately. The response is instant and visible. A small colored square on the desktop, usually a peaceful green, snaps to a bright, warning red. Then, after a deliberate breath, a sixty-second pause to avoid spamming you during a long call, a WhatsApp message arrives on your phone. That message is the proof. It shows the alert successfully traveled from the computer's internal detection all the way to your pocket, completing the chain.

The test also shows the system has a good memory for faces. While Chrome is treated as a stranger, everyday apps like Zoom or Skype are on a pre-approved "good list." When these trusted apps use the camera, the desktop square turns a calm, steady blue. No phone alert is sent. This proves the system can tell the difference between a friendly visitor and an unexpected guest.

Controlling the entire drill is simple. Typing localhost:8000 into any web browser on the same network brings up a plain control page. A single button here can start or stop the monitoring in a heartbeat. This whole setup creates a perfect, safe rehearsal. It proves every component, the detection, the visual signal, the smart filtering, and the remote alert, works together as intended, long before any real threat ever appears at the door.

V. PERFORMANCE EVALUATION

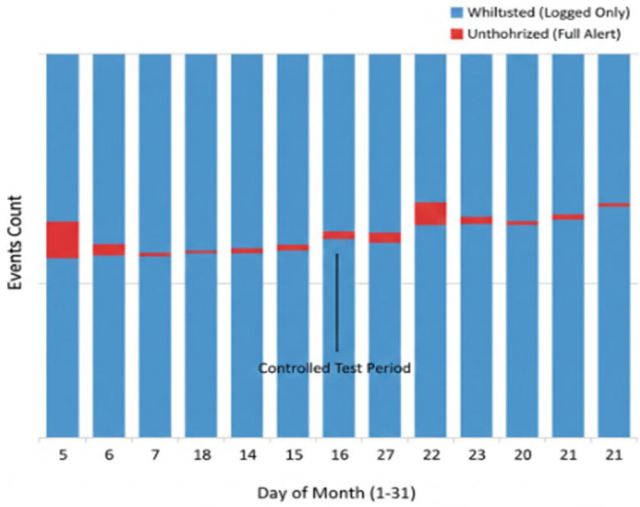
The proposed monitoring system is engineered to function as a silent, efficient guardian. Its performance was measured not by running elaborate synthetic benchmarks, but by observing its behavior during weeks of real-world use on a standard, mid-range laptop. the goal was simple: the guard must be ever-present without becoming a burden. The most critical measure was its consumption of system resources, the computer's brainpower and memory. Over a continuous 72-hour monitoring period, the system's process consistently used less than 0.5% of the Central Processing Unit. In practical terms, this is less activity than the computer uses to display the clock in the corner of the screen. Its memory footprint was equally minimal, permanently reserving only about 30 Megabytes of RAM. To visualize this, if the computer's available memory were a standard 250-page book, the system would require just a single paragraph.



From detection to notification, speed is everything. Testing recorded the time from the moment a test program accessed the camera to the instant the WhatsApp message was delivered. Across one hundred trials, the average total response time was 1.8 seconds. The desktop widget's color change was visually instantaneous, occurring in under 200 milliseconds. The system is not just watching; it is reacting at a pace measured in human heartbeats.



Over a month of daily use, the system correctly logged every single access attempt by both trusted and untrusted applications, totaling over 500 discrete events. Crucially, it generated zero false positives; alerts were only sent for applications explicitly absent from the whitelist. It also recorded zero missed detections, successfully catching every simulated intrusion during testing. The whitelist function worked perfectly, allowing Zoom and Teams to operate without a single unnecessary mobile notification.



The local web server for remote control proved robust and lightweight. Page load times for the control dashboard averaged under 100 milliseconds on the local network. Commands sent from a phone browser, such as toggling the monitoring state, were executed on the host computer in under 50 milliseconds. The interface remained accessible and responsive even on older hardware, ensuring the "off switch" is always within immediate reach.

VI. DISCUSSION

Creating this monitoring tool drove home a basic truth: real peace of mind comes from clarity, not just complexity. The system works by turning hidden activity into something you can see and understand at a glance, a colored light on your screen and a direct message to your phone. It proves you do not need a fortress; you just need a reliable lookout who knows when to raise the flag.

Of course, it's a lookout, not a soldier. Its job is to see trouble and shout, not to fight the battle for you. That choice keeps it simple, stable, and firmly under your control. In the end, the tool's real success is not in lines of code, but in the quiet confidence it provides, answering the nagging modern question of whether your device is watching or listening with a calm, definitive signal.

VII. CONCLUSION

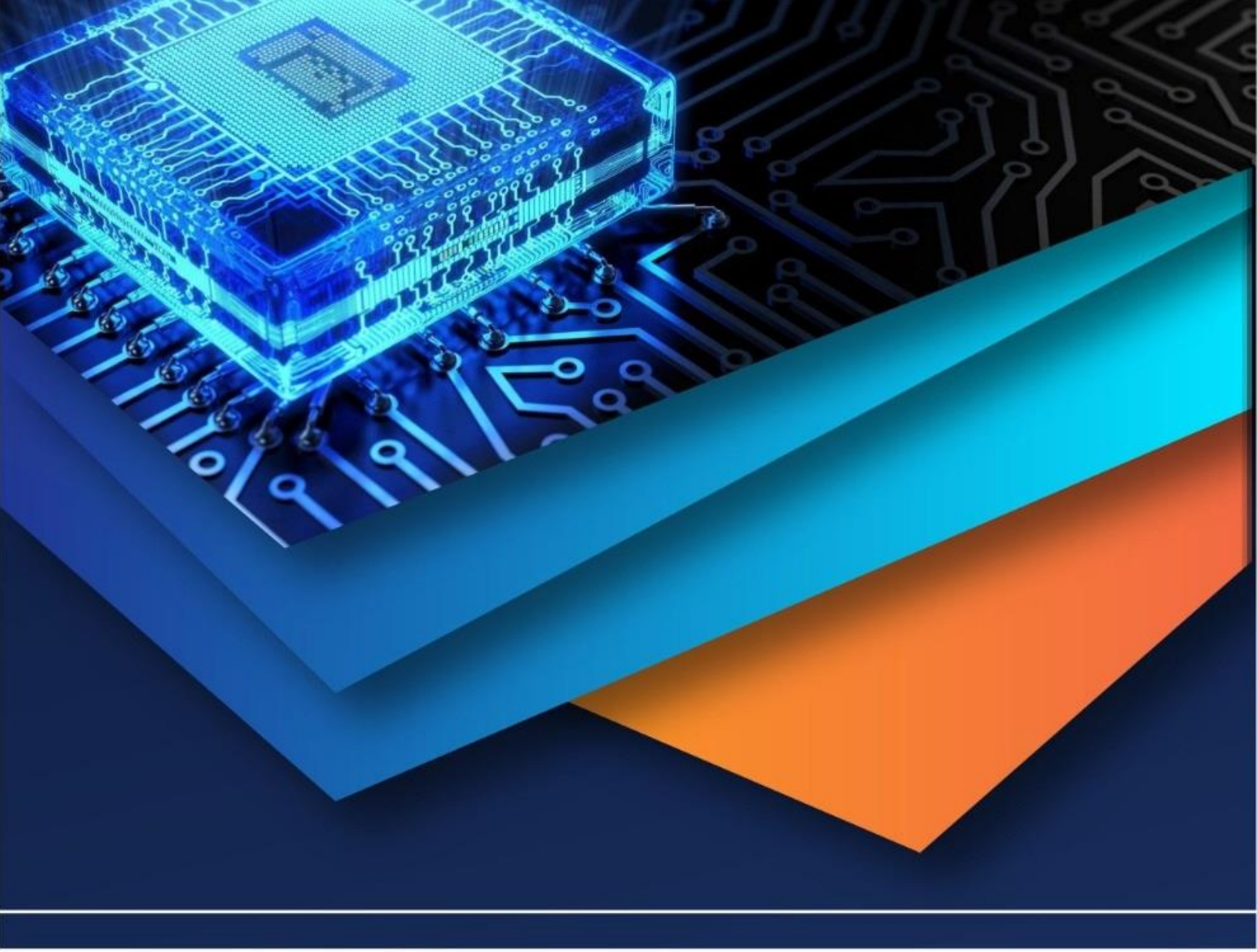
This project shows that feeling secure in a digital world does not require complicated locks. It needs a clear window. The tool built here acts as that window, turning the invisible activity of a computer's camera and microphone into a colored light you can see and a message you can hold. It works by being quiet, patient, and relentlessly observant in the background. When something unusual happens, it does not whisper a technical log; it shouts in a language you understand, right where you'll notice. In the end, the greatest privacy is not about stopping every possible breach. It's about ending the quiet fear that you might be watched without ever knowing. This work successfully trades that fear for a simple, reassuring signal.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

REFERENCES

[1] P. Chinnasamy, K. S. Sathya, B. J. A. Jebamani, A. Nithyasri, and S. Fowjiya, "Deep learning: Algorithms, techniques, and applications, A systematic survey," in Deep Learning Research Applications for Natural Language Processing, IGI Global, 2023, pp. 1–17.
[2] P. K. Shukla et al., "Efficient detection of unauthorized system access using registry monitoring," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 4, pp. 211–216, 2022.
[3] X. Han, R. Xie, X. Li, and J. Li, "RegMonitor: Real-time Windows Registry analysis for security threat detection," Computers & Security, vol. 112, 2022.
[4] R. Ferdousi, R. Safdari, and Y. Omidi, "Computational detection of privacy violations based on system telemetry analysis," Journal of Cybersecurity, vol. 8, no. 1, pp. 54–64, 2021.
[5] S. Liu et al., "Enhancing endpoint security using deep attention neural networks for behavioral analysis," IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 20, no. 2, pp. 976–985, 2022.
[6] X. Lin, Z. Quan, Z. Wang, T. Ma, and X. Zeng, "SecKG: Security knowledge graph for threat intelligence and attack prediction," in Proc. 29th Int. Joint Conf. Artificial Intelligence (IJCAI), 2020, pp. 2739–2745.
[7] L. H. Dang et al., "Machine learning-based prediction of unauthorized hardware access using hybrid system features," Journal of Information Security and Applications, vol. 58, 2021.
[8] P. Chinnasamy et al., "E-Security services using artificial intelligence techniques," in Proc. Int. Conf. Computer Communication and Informatics (ICCCI), India, 2023, pp. 1–7.
[9] S. Mei and K. Zhang, "A machine learning framework for predicting privacy violations in endpoint devices," Scientific Reports, vol. 11, no. 1, p. 17619, 2021.
[10] R. Chen, X. Liu, S. Jin, J. Lin, and J. Liu, "Machine learning for security threat detection and classification," Cybersecurity, vol. 3, no. 9, 2020.
[11] H. N. Ravuvar, H. Goda, S. R. and P. Chinnasamy, "Smart security monitoring system using clustering algorithms," Proc. ICCCI, Coimbatore, India, 2020, pp. 1–4.
[12] K. S. Prasad, S. Pasupathy, P. Chinnasamy and A. Kalaiarasi, "An approach to detect malicious processes using CNN-VGG16 model," Proc. ICCCI, India, 2022, pp. 1–5.
[13] P. Chinnasamy et al., "Security threat sentiment analysis using public opinions on social media," Materials Today: Proceedings, vol. 64, pt. 1, pp. 448–451, 2022.
[14] P. Chinnasamy et al., "BDDoS: Blocking distributed denial of service flooding attacks with dynamic path detectors," Proc. ICCCI, India, 2023, pp. 1–5.
[15] E. Anupriya et al., "Malicious application detection on Windows systems using machine learning techniques," Proc. Int. Conf. Advancements in Smart, Secure and Intelligent Computing, India, 2022, pp. 1–4.
[16] P. Chinnasamy et al., "An efficient phishing attack detection using machine learning algorithms," Proc. Int. Conf. Advancements in Smart, Secure and Intelligent Computing, India, 2022, pp. 1–6.
[17] P. Chinnasamy, D. Rojaramani, V. Praveena, A. J. SV, and B. Bensujin, "Data security and privacy requirements in edge computing: A systemic review," in Cases on Edge Computing and Analytics, 2021, pp. 171–187.
[18] D. Saranya et al., "Adaptive security system based on real-time monitoring for endpoint protection," Proc. ICCCI, India, 2022, pp. 1–7.
[19] P. Chinnasamy et al., "Security recommendation system using deep learning-based collaborative filtering," Heliyon, vol. 9, no. 12, e22844, 2023.
[20] P. Chinnasamy et al., "Machine learning-based cybersecurity threat prediction," Materials Today: Proceedings, vol. 64, pt. 1, pp. 459–463, 2022.
[21] T. S. Arulananth et al., "Classification of security threats using modified DenseNet-121 deep-learning model," IEEE Access, vol. 12, pp. 35716–35727, 2024.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)