SOC Copilot: An AI-Powered Security Operations Assistant for Automated Threat Detection and Intelligent Incident Response

Dr. C V Madhusudhan Reddy, Dr. G K V Narsimha Reddy, Karu Praneeth Kumar, Aarupalli Karthik, Kanike Vinay, Pavadi Bharath

Dept. Of Computer Science and Engineering (Artificial Intelligence), St. Johns College of Engineering and Technology, Yemmiganur, 518301, India

Abstract- ***Heavy cyber attacks have pushed security teams into constant crisis mode. Day after day, they face endless warnings - each needing attention, each eating up time. Too many logs come in different shapes, too much noise clouds judgment, false alarms pile up fast. Analysts grow numb. That dullness slows everything down: spotting danger takes longer now, fixing it even more so. Old tools collect data but stick to rigid rules, unable to shift when hackers change tactics - they just add clutter instead. Enter SOC Copilot - a smart helper built with two types of algorithms working together. One spots odd behavior without knowing what's coming; the other sorts real threats into categories using past examples. Together, they cut through confusion. It pulls records from various sources like JSON, CSV, Syslog, Windows EVTX files - not missing a beat. From those entries, it builds 78 unique traits based on patterns, timing, actions, connections. Then ranks urgency levels P0 to P4. Every result ties back to known attack methods via MITRE ATT&CK - and explains why clearly, plainly. Running without internet access, SOC Copilot puts governance at its core. Oversight stays with analysts every step of the way. Every move gets recorded - full traceability built in. Data never leaves local systems, meeting strict control standards. Testing shows it sorts threats correctly more than 99 times out of 100. Workload drops sharply because routine sorting happens automatically. Alerts arrive packed with context, cutting down decision time. Guidance comes clear, pointing straight to next steps. Speed improves across the entire reaction cycle, Cybersecurity automation improves threat detection with machine learning***

I. INTRODUCTION

More businesses, governments, and vital systems now live online - that means more openings for cyberattacks than ever before. Last year, a typical data breach cost around 4.45 million dollars on average, based on findings from IBM; it also took companies nearly nine months just to spot and stop those breaches [1]. With stealthy long-term intrusions, unknown software flaws, constant ransom demands, and hacks spreading through suppliers, today's risks call for sharp, fast, and smart defenses in place.

Housed within many organizations, Security Operations Centers act like a central hub for cyber defense. Watched nonstop, these teams spot threats, look into breaches, then guide how to react. Instead of working blind, they pull data through tools known as SIEMs - systems that gather alerts from firewalls, IDS or IPS units, EDR software, apps, and networking gear. Even so, today's operations run into deep-rooted issues that quietly slow things down.

Every day, enterprise security teams face a flood of warnings - sometimes numbering in the tens of thousands. Because so many turn out to be mistakes, roughly 95 percent by some estimates, people start tuning them out. When too much noise fills the screen, real dangers slip through without proper attention. Workers grow numb after hours of sorting weak signals from messy data streams. Judgment wavers when one shift ends and another begins, simply because minds tire. What feels like vigilance can quietly become routine dismissal.

Hours pass before replies come. Sorting through logs by hand takes ages. Because one mistake hides in many records, workers check system after system. When clues appear, they link them using experience. Each step drags detection further into the future. Fixing problems starts late as a result. Intruders stay hidden longer simply because responses crawl.

Most older security tools gather logs well, yet they miss subtle clues that separate normal hiccups from real dangers. Instead of understanding behavior, these systems rely on fixed triggers that fail when attacks change shape. Because of rigid logic, odd events get ignored or wrongly flagged every day. Without learning from surroundings, alerts often point nowhere useful.

Because old ways of checking data by hand or using fixed rules often fall short, researchers have turned to artificial intelligence and machine learning to strengthen cyber defenses. Instead of relying only on set conditions, smart systems now sort through mountains of logs automatically. These tools catch quiet signs of danger that simpler methods miss. By weighing how risky each warning is, they help decide what needs attention first. Responses come with added background details, making them more useful. Human insight stays central, while the system handles heavy lifting behind the scenes.

A fresh look at SOC Copilot begins here. Built on artificial intelligence, it tackles current security hurdles using a mix of machine learning methods stitched together. Instead of relying on just one technique, this tool uses outlier spotting without labels - thanks to Isolation Forest - alongside labeled pattern recognition via Random Forest for broader coverage. Unusual behaviors get flagged even when they've never been seen before. One piece feeds into another: logs come in first, then cleaned up before deeper inspection kicks in. Threats are analyzed by models trained to tell types apart, not just spot differences. Alerts show up with clear reasons why, tied directly to real-world tactics from the MITRE framework. Analysts respond, their input loops back to sharpen future results. Over time, if performance slips, the system notices shifts and adjusts. Control stays central, designed around oversight needs from day one.

What comes next unfolds like this. Following part two looks at earlier work on the topic. Problem definition takes shape in section three. A new system enters the picture within section four. Architecture appears through section five. Method steps fill out section six. Findings get attention in section seven. Benefits show up first, uses follow after - sections eight and nine handle those. What might come later gets room in section ten. The final thoughts settle into place across section eleven.

II. LITERATURE REVIEW

Starting with older rule-driven methods, research into using computer tools for spotting cyber threats covers a wide range. Moving beyond those, newer studies explore how complex neural networks can identify risks too.

Older security tools like Splunk, IBM QRadar, or ArcSight form the base layer many teams still depend on today. Instead of scattered records, they pull logs into one place while spotting activity links as things happen. Because these setups run on fixed logic patterns, though, fresh threats often slip through without warning. Updating them means constant human effort just to keep up with known behaviors. When attacks unfold slowly across several steps - yet avoid familiar traces - they rarely trigger alerts. Research by Zuech and others confirms how weak such rigid methods are against stealthy intrusions.

Signature-based tools like Snort and Suricata scan network data by matching it to stored patterns of past attacks. Because they rely on pre-existing records, their strength lies in spotting familiar dangers. Yet, as Roesch pointed out, they lag when facing new threats - arriving too late to catch what has never been seen before. Without prior examples, these methods miss unknown vulnerabilities entirely. Zero-day breaches slip through, just as shape-shifting viruses do. Fresh tactics also go unnoticed, simply because no rule yet exists to flag them. Their design waits for history to repeat, leaving blind spots wide open.

Out in the world of digital security, spotting strange behavior matters a lot. One way researchers tackle this uses machines that learn patterns on their own. A method built by Liu and team splits data points at random until odd ones stand out fast. Instead of needing known examples of attacks, it learns what normal looks like first. When something moves too far from that norm, alarms go up silently. Because of this trick, never-before-seen threats might get caught early anyway.

One way to sort data involves using decision trees grouped together. These collections, known as Random Forest models, work well when identifying different kinds of cyber threats. Research published under reference [7] supports this idea. When tested on a standard set of network records called CICIDS2017, such systems ranked high in precision. That test was detailed by Sharafaldin and team [8]. Instead of just spotting normal behavior, they correctly flagged harmful actions like forced access attempts, malicious website probes, hidden intrusions, zombie device signals. Grouped tree strategies handled these separations better than many alternatives.

Deep learning now plays a role in cybersecurity through tools like autoencoders, RNNs, and transformers used to spot threats. Shone and team presented an autoencoder system able to detect intrusions with strong results on standard test data. Still, these models can be hard to interpret - problematic when experts need clear reasons behind alerts. Because of this opacity, trust becomes difficult even if performance looks good. Understanding model choices matters just as much as catching threats. Without transparency, adoption in real security operations faces hurdles.

Here's what we miss so far. Even with progress, current studies leave holes - ones SOC Copilot steps into. Not many tools blend different detection methods; instead they stick to just one path. Some skip clear reasoning entirely, treating explanations as an afterthought. Control features? Rare. Human guidance built into the system? Hardly ever seen. Most suggested setups rely on internet access, so they fail where networks are isolated. Yet SOC Copilot works without connection, combining multiple models, clear reasoning steps, strong oversight design, while running entirely on local machines.

III. PROBLEM STATEMENT

What happens when too many pressures stack up inside today's security hubs? Detection slows. Response weakens. The whole system stumbles under its own weight. Alerts pile high while teams stretch thin. One misstep leads to another. Overload becomes routine. Speed fades just when it is needed most.

Suddenly, logs multiply nonstop. Security tools like firewalls, intrusion detectors, and login systems cough up endless streams every day. Cloud apps, servers, and device monitors add their own chunks too. Picture a single medium company - millions of entries pour in by noon. Each one must be decoded, shaped into standard form, then weighed against background clues. Patterns shift without warning. The load never stops growing.

Most alerts from rule-driven tools aren't real threats - research shows over 95 percent get flagged by mistake. Because of this, security teams waste hours on harmless events instead of chasing down true dangers. What slips through is the quiet buildup of ignored warnings piling up like unread mail.

Too many alerts pile up. Triage work feels endless, one case after another. Digging into tough threats takes serious mental effort. That grind wears people down over time. Right now, there are around 3.4 million open cybersecurity jobs worldwide. Fewer hands make each task heavier. Pressure builds when teams run thin. Empty roles mean more load on those who stay. Burnout creeps in quietly, then hits hard. The shortage isn't just numbers - it shows in tired eyes and slower decisions. Gaps in staffing deepen stress across shifts. Experience fades fast when people leave. Keeping pace gets harder every month. Work stretches further than bodies can go. Minds grow dull under constant noise. Help does not arrive quickly enough. Each missing person leaves a hole others must fill. Stress stays longer than solutions do.

When alerts pop up, they often miss key details about what is actually happening. Without clear background, sorting out how serious an event might be gets tricky. Pulling together info from different places becomes a regular task. Connecting dots means using outside threat data alongside personal experience. Making sense of it all takes time because pieces are scattered across systems.

Starts with an alert, then someone has to sort it out by hand. From there, gathering proof takes time because people do each step themselves. This slows things down when checking how bad a problem might be. Someone must figure out what went wrong without help from automation. Decisions about what to do next come later than they should. Each delay gives threats more room to cause harm inside the company.

One way to put it: when heaps of messy security logs come in every format imaginable, what mix of smarts could sift through them, spot real dangers, sort by urgency, then lay out clear reasons why - helping human analysts choose fast without breaking rules or losing control over where data goes or how choices get made.

IV. PROPOSED SYSTEM - SOC COPILOT

A smart helper for security teams, SOC Copilot grows step by step, fitting together like puzzle pieces. Layer by layer it tackles challenges, shaped around real needs. Each phase adds strength without overload. Built to adapt, not force. Problems meet structure, one stage at a time.

A. Log Ingestion

A fresh start comes with how logs enter the system - ready to handle different shapes without favoring one. One way deals with JSON or JSONL, pulling structured entries from cloud tools and web interfaces. Another path opens up CSV files, often saved by older platforms in row-and-column layouts. Devices like routers speak Syslog, a common tongue the system decodes on arrival. Then there is Windows EVTX, a packed binary form that holds alerts from Microsoft environments. Every type gets its own method to pull out what matters - time stamps show when, IPs point where it came from and went to. Ports and protocols slip into place beside event labels and risk levels, all shaped into something consistent.

B. Log Parsing and Preprocessing

Logs start their journey through several steps before becoming usable. First up, timestamps get shifted into a single time zone - UTC - and shaped uniformly using ISO 8601 rules. Instead of keeping different naming styles from various sources, field names are reshaped to fit one standard structure. When it comes to categories like status or type, words turn into numbers for easier handling later on. Each record then faces checks based on predefined formats, where errors trigger logging but keep flawed data out of results.

C. AI/ML-Based Analysis Engine

One part of the system uses Isolation Forest, which learns what normal network activity looks like by studying only harmless traffic, then gives each event an oddness score between zero and one. This method builds patterns without labels, spotting outliers quietly. On another track, Random Forest steps in after learning from labeled examples in the CICIDS2017 collection, where every kind of behavior gets fair attention during training. Instead of lumping logs together, it sorts them carefully - into seven types, including quiet traffic, flood attacks, guessing breaches, harmful software bursts, secret data leaks, stealthy scouting moves, and database probing stabs. Each model runs its own way, yet both feed conclusions to the main decision hub.

D. Threat Classification and Ensemble Logic

Starting with two model results, the Ensemble Coordinator mixes them via a weighted system. A math formula brings together scores: 0.4 times the Anomaly Score plus 0.6 multiplied by Threat Severity and Classification Confidence. Out comes one number - the Combined Risk Score. That figure shapes how alerts get ranked, from P0 being most urgent down to P4 least urgent. Levels like Critical, High, Medium, or Low follow next based on severity. Because it uses two signals, familiar risks and fresh ones aren't missed. While classifiers catch recognized dangers, anomaly detectors spot odd patterns never seen before. Each piece feeds into final judgments without leaning too hard on just one source.

E. Explainability and Recommendation Engine

Sometimes it talks through why an alert happened, showing how sure it is. What makes something seem off gets spelled out using real behavior clues. A list pops up with what mattered most in raising the flag. It checks where limits were crossed and says so plainly. Tactics hackers use appear mapped out the way experts classify them. Next steps show up only if they fit the type of danger found. These hints stick to official roles like stop, check, fix, and rebuild. Every piece connects without flashing lights or drama.

F. Dashboard Visualization

A live feed of threats shows up on screen through PyQt6, built right into the desktop view. Moving number displays pulse with fresh stats, catching attention without flash. Alerts line up by urgency, sorted high to low, changing shade based on risk level - updated every two seconds like clockwork. Click one, and deeper details unfold nearby, showing what tipped the scale and why it matters. Colorful dots blink along the bottom edge, each tracking a different part of the workflow behind the scenes. Settings hide at the side, ready for fine-tuning limits or adjusting oversight rules when needed.

G. Technology Stack

Component	Technology
Core Language	Python 3.10+
ML Framework	Scikit-learn (Isolation Forest, Random Forest)
Desktop UI	PyQt6

Data Processing	Pandas, NumPy
Database	SQLite
Configuration	YAML
Testing	Pytest (208+ tests)
Packaging	PyInstaller
Version Control	Git

V. SYSTEM ARCHITECTURE

Beginning at the core, SOC Copilot uses separate levels that unfold in stages. Not just stacked, these layers split into three working phases. Each level connects through clear transitions instead of blending together. Five main sections make up the structure, each with its own role. Built this way, the system allows changes without disruption. Compliance needs shape how parts are arranged. Flexibility comes from separation, not complexity.

A. Frontend Interface Layer

Out of the box comes a desktop setup built on PyQt6, bringing together a sidebar that moves with keystrokes leading the way. Hidden down below sits a status strip lit by tiny LEDs whispering what's happening under the hood. Pages pile behind one another - Dashboard up front, then Alerts, Investigation sliding in next, followed by Assistant and finally Settings tucked at the back. Messages pass only one direction into this world, guided by a bridge tied to a controller locked from changes. Communication stays lean, watching silently without reaching back.

B. Backend API Layer

From start to finish, the Application Controller handles every step of log handling - taking in data, cleaning it up, pulling out key details, running predictions, then sending alerts. Batching pieces together on the fly, it processes information in small chunks, adjustable by size and timing based on settings.

C. AI Processing Engine

When the app starts up, it pulls in trained models through joblib, locked as read-only for inference. From raw inputs, feature work builds 78 numbers grouped into four kinds - no mixing allowed. Statistical traits show up first, twenty-two strong: things like how many packets move, data size, time a connection lives. Then come eighteen that track timing patterns, such as what hour traffic hits. Behavior shapes the next set, twenty markers deep - one tracks how often requests fire off, another watches failed attempts pile up. Last stretch holds network signals, eighteen of them: oddities like irregular port usage, count of distinct endpoints reached.

D. Database Layer

Inside SOC Copilot, data sticks around using SQLite. Analyst decisions go into a Feedback Store - later used to check how well models perform. A separate Governance Database keeps every change logged, never erasing anything added. Feature patterns get saved over time inside Drift Monitoring, capturing shifts quietly behind the scenes.

E. Response Generation Layer

A signal appears after analysis finishes. This output bundles a severity tag - ranging from P0 to P4 - with links to known attack patterns under MITRE ATT&CK. Each entry sorts risks by category while listing key data points that shaped the outcome. Reasons unfold in clear steps anyone can follow. Suggestions for next moves come attached, grounded in surrounding conditions.

F. End-to-End Data Flow

Out of order comes structure - log files start it all. Right after, format detection kicks in without delay. Once that finishes, parsing begins immediately afterward. Following close behind, validation checks every piece carefully. Then normalization adjusts values into alignment. Feature extraction pulls exactly 78 markers right at that point. A bit later, isolation forest scoring weighs anomalies silently. Not long after, random forest classification labels each event quietly. Their results meet during ensemble coordination just once. From there, risk score calculation adds up consequences slowly. Alerts appear only when thresholds break unexpectedly. Explaining each alert happens before anything else next. The dashboard displays everything moments later always. Analysts review what shows up eventually. Finally, feedback gets stored for good measure.

VI. METHODOLOGY

A. Log Data Processing

A fresh start happens when the system checks file types by looking at endings and headers. After that comes breaking down records, handled differently depending on their shape. Errors fall away during structure checks, leaving only what fits. Moments get lined up in one global rhythm, set to midnight-based labels from an international rulebook. Pieces shift into standard slots, guided by fixed blueprints. Categories turn into steady codes, drawn from unchanging wordbooks.

B. Feature Extraction

Seventy-eight numerical traits get pulled out by the feature building part, grouped into four types. Statistics make up twenty-two of them - things like how many packets move through, total bytes, session length, along with gaps between arrivals. Time-driven ones add eighteen more, tracking rhythms across hours plus shifts in timing between events. Behavior shows up in twenty indicators tied to how often requests happen, what share fail, and sudden spikes in use. Another eighteen come from network layout, measuring spread of ports used and which protocols show up most. Sequence stays locked down once set, saved right next to the model files so inputs line up every time.

C. AI Inference Workflow (Pseudo-Algorithm)

One way to spot threats uses a mix of methods. It takes a data table X with n rows and seventy eight columns from logs. The result is a list A of warnings, ranked by urgency, each with risks spelled out plus reasons why. First step loads two ready made tools: M_IF for oddity checks, M_RF for sorting types. Each row x_i gets scored oddly using M_IF's built in judge scaled between zero and one. Then M_RF guesses what kind it might be along with how sure it feels about that guess. Highest certainty among those guesses becomes its confidence level. Risk blends four tenths of weirdness score with six tenths of severity times belief strength. When this number crosses a line we draw ahead of time, an alert wakes up - gets details written down like possible harm source, attack pattern link, next move idea - and joins group A. At the end, only these flagged items get handed back.

D. Threat Classification Logic

Class	Description	Severity
Benign	Normal network activity	0.0
DDoS	Distributed denial-of-service	0.8
Brute Force	Authentication attacks	0.7
Malware	Malicious software execution	1.0
Exfiltration	Unauthorized data transfer	1.0
Reconnaissance	Network scanning/probing	0.5
SQL Injection	Database exploitation	0.9

E. Risk Scoring

Risk level totals come from two parts. One part checks how far actions stray from usual patterns, counting for 40 percent of the total. Instead of adding directly, it blends with another piece that matches known threats, making up 60 percent. When joined, their result fits into one of four bands. If the number hits 0.8 or more, it lands in critical. Between 0.6 and just under 0.8 means high. A value from 0.4 to below 0.6 reads as medium. Anything beneath 0.4 shows low risk.

VII. RESULTS AND DISCUSSION

A. System Performance

A fresh look at SOC Copilot began with tests on the CICIDS2017 data - real traffic marked by hand through various breach attempts. While not every condition matched live environments, patterns emerged clearly over time. Because labels guided the process, spotting differences became easier without extra tools involved. Though some noise crept in, results still lined up close with expected outcomes. From start to finish, evaluation stuck strictly to recorded behaviors seen during simulated break-ins.

Metric	Value
RF Classification Accuracy	99.99%
IF Anomaly Separation	Confirmed
Feature Extraction	78 features, consistent
Model Loading Time	1–2 seconds
Single Record Latency	< 10 ms
Batch (1,000 records)	2–5 seconds
Large-Scale (100K records)	2–5 minutes

B. Example Log Analysis

Out of nowhere, a strange pattern shows up in the network data. Port activity feels off - entropy hits 0.92, way outside normal ranges. Destinations? Forty-seven different ones, more than expected. Timing between packets lacks rhythm, uneven and unpredictable. The system flags it sharply - a score of 0.85 from Isolation Forest points to clear oddness. A second check using Random Forest says Malware, nearly certain at 92.5%. Weighted together, anomaly strength and classification certainty build a risk number: 0.895. That lands squarely in critical territory, ranked P0. Behind the scenes, logic traces back - the label sticks because evidence aligns so tightly. "Malware," it concludes, "with strong backing." A spike in the anomaly score often points to odd activity. Port entropy stands out, along with how many different destinations appear, plus timing since last event. Look into running processes after cutting off the device

C. Comparative Evaluation

Approach	Known	Novel	FP Rate
Rule-Based SIEM	High	None	High
Isolation Forest Only	Moderate	High	Moderate
Random Forest Only	High	None	Low
SOC Copilot Ensemble	High	High	Low

D. Response Time and Workload Improvement

What happens when alerts get sorted by software instead of people? Review time drops because machines handle the first look. Context appears fast, cutting what used to take minutes down to just seconds. Suggestions show up ready to apply, so there is no waiting around to plan next steps. Attack patterns match known behaviors quickly, thanks to embedded framework tagging. Work gets lighter not by doing more but by filtering out false signals early. Confidence grows when multiple detection models agree on a result. Critical items stand apart from routine ones using clear urgency labels. Thousands of logs move through checks in groups, not one at a time. Explanations come already written, saving effort after detection. Mistakes teach the system how to improve - each outcome feeds back into future judgments. Length stays fixed, detail remains intact, nothing added, nothing lost.

VIII. ADVANTAGES

One big plus? Less work for security teams because alerts get sorted automatically. Quick checks happen in less than 10 milliseconds for every entry. Smarter conclusions come from digging into details and linking findings to real-world attack patterns. The system grows step by step without breaking a sweat. Safety stays front of mind - automated actions sit turned off until needed. Runs cut off from the internet, keeping data under local control. Spots familiar dangers along with brand-new unknown attacks. Every result can be checked again later thanks to fixed scores and unchangeable logs.

IX. APPLICATIONS

Out here, SOC Copilot fits into many kinds of security setups. Picture large company security teams leaning on it to back up their analysts. Instead of reinventing the wheel each time, service providers bundle it to keep threat spotting consistent for every customer. When systems stretch between clouds or mix old and new tech, tossing different log types at it works just fine. Some government hubs insist everything stays inside their walls - that’s where its offline mode becomes key. Places like power plants, hospitals, banks operate cut off from broad networks, yet still run tight checks using this. Even classrooms teaching cyber defense pull it into labs for real-world practice. Length holds steady. No extras tagged on.

X. FUTURE SCOPE

One path moves toward live data flow into tools like Splunk, Elastic SIEM, and QRadar through stream-based links. Instead of waiting, alerts come as events unfold across systems. A different angle uses deep autoencoders - already outlined in code - to spot odd behavior machines might miss. This method learns normal patterns then flags outliers without clear rules. Working alongside it, automated fixes plug into existing workflows where actions follow policy checks. Each fix runs only when approval paths confirm safety. For organizations serving many groups, setups allow separate spaces with unique settings under one roof. Tenants stay isolated while sharing core infrastructure efficiently. Shifting to cloud formats brings flexibility by packaging services in containers managed by Kubernetes. Scaling happens quietly based on load changes behind the scenes. Another step examines event chains using transformer models that trace complex attack sequences over time. Patterns emerge from long strings of logs once invisible at glance. Lastly, searching incidents feels more natural since queries accept plain questions thanks to large language model support. Responses form conversation-style instead of rigid syntax demands.

XI. CONCLUSION

This study introduced SOC Copilot, a smart helper built to ease major pressures on today’s security teams.

Instead of just spotting familiar threats, it uses Isolation Forest to flag odd behaviors never seen before. On top of that, Random Forest steps in to sort different kinds of attacks with precision. Together, these methods allow consistent identification of dangers - common ones included. The system handles new risks while staying reliable on established patterns.

What stands out first? A mix of machine learning models working together, decisions balanced by weights, hitting over 99% correct classifications. Next up - seventy-eight carefully shaped features pulled from stats, timing patterns, actions, and network signals. Clarity matters here. Built-in explanations show why choices were made, highlight what factors mattered most, point to known threat behaviors. Control comes before convenience. Automation stays off unless flipped on, emergency stops exist, every action gets logged without deletion. Running completely disconnected isn't an afterthought - it's required. Data never leaves the local environment.

A fresh approach to security tools shows promise, blending smart automation with real human insight. When attacks grow more complex and frequent, helpers such as SOC Copilot start mattering more inside company defenses. These systems support teams by keeping pace with rising dangers, even when there are too many alerts and too few experts to review them.

REFERENCES

[1] IBM Security, "Cost of a Data Breach Report 2023," IBM Corporation, Armonk, NY, USA, 2023.

[2] Ponemon Institute, "The Economics of Security Operations Centers: What is the True Cost for Effective Results?," Ponemon Institute LLC, 2020.

[3] A. Chuvakin, K. Schmidt, and C. Phillips, Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management, Syngress, 2012.

[4] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," Journal of Big Data, vol. 2, no. 1, pp. 1–41, 2015.

[5] M. Roesch, "Snort — Lightweight intrusion detection for networks," in Proc. 13th LISA, USENIX, 1999, pp. 229–238.

[6] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. IEEE ICDM, 2008, pp. 413–422.

[7] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[8] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. ICISSP, 2018, pp. 108–116.

[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," IEEE Trans. Emerg. Topics Comput. Intell., vol. 2, no. 1, pp. 41–50, 2018.

[10] M. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset," IEEE Access, vol. 10, pp. 40281–40306, 2022.

[11] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," J. Netw. Comput. Appl., vol. 60, pp. 19–31, 2016.

[12] G. Apruzzese et al., "On the effectiveness of machine and deep learning for cyber security," in Proc. CyCon, 2018, pp. 371–390.