

RUDRAS: A COGNITIVE IMMUNOLOGICAL DEFENSE FIREWALL FOR ZERO-TRUST, AI-NATIVE NETWORK PROTECTION

Deepak P S

Department of Computer Science and Engineering,
C Byregowda Institute of Technology, Kolar, Karnataka, India
deepakkrishnark@gmail.com

Abstract: Modern attackers rarely follow obvious paths. Old- style firewalls and tools that only look for known attack patterns often miss new kinds of attacks, quiet intrusions, or abuse by people inside the system especially when this activity is hidden in encrypted traffic or unusual network routes. Instead of relying on those methods alone, Rudras takes a different approach: built in Rust, it combines smart packet filtering, machine-learning methods to spot odd behavior, tight access controls, real-time threat updates, and tamper-resistant logs, all working together in one system. Every device gets its own baseline of “normal” activity and a tailored response plan, so defenses shift as usage evolves rather than staying locked to fixed settings. Over thirty-four safeguards such as deep inspection of messages, checks on web requests, detection of break-in attempts, and monitoring of user actions run in parallel, processed quickly in a single streamlined flow designed for memory safety and low delay. Learning happens locally across machines without sending all raw data to a central hub, yet policies stay consistent thanks to a clear, structured rule model. The paper then explains how Rudras was designed, implemented, and tested in lifelike enterprise conditions, and shows that it fills key gaps left by older security layers while creating room for future work on privacy-preserving sharing of warning signs and collective validation of threats across large networks

Keywords: Network security, Intrusion detection systems (IDS), Intrusion Prevention systems (IPS), AI-based anomaly detection, Cognitive immunological defense, Zero Trust architecture, Micro-segmentation, Rust-based firewall, Behavioral baseline modeling, Advanced persistent threats (APTs), Federated learning for security, Moving target defense, Immutable forensic logging.

I. INTRODUCTION

All Security once focused on fixed boundaries now faces fluid setups cloud systems, distant devices, and operational tech interwoven. Traditional tools built for known threats falter when new tactics emerge even as systems endure heavy traffic from denial-of-service disruptions, they still detect unusual sequences, hidden channels used for remote control, along with shifts across internal networks. Meanwhile, zero-trust models gain traction validating user identities, device status, and ongoing actions constantly instead of assuming safety due to where access originates, a framework outlined clearly in NIST SP 800-207 [2][18][19].

What if security worked like an immune system? Rudras does exactly that built in Rust, it runs as one compact program enforcing protections at full speed across devices and entry points. Instead of stacking separate tools, this platform weaves together over forty defensive functions into a shared structure where data

flows through a single analysis chain. Each part recognizes normal behavior for every device, spotting anomalies much like antibodies found invaders. When something seems off, alerts spread quietly among connected units without waiting for people to respond.

Most systems juggle mismatched components, firewalls here, analytics there which often fail because they do not speak the same language. Not so here; everything uses the same memory map, cutting confusion caused by disjointed tech. Coordination happens naturally, almost silently, shrinking weak spots created when tools work against each other.

This study offers four key outcomes. One outcome: a cohesive network defense framework built to resist buffer overflows and race conditions without sacrificing speed typical of C-based systems, designed with memory safety at its core. Another advance comes through a behavior modeling system tailored for AI methods that combines traffic patterns, user analytics, and structural data from networks to spot unknown threats and malicious insiders [3][4][5][6].

A third piece demonstrates how zero-trust principles like fine-grained segmentation and device health checks can be embedded straight into low-level packet handling, following standards set by NIST. In closing, real-world test cases illustrate effectiveness, while unresolved challenges point toward further inquiry using this setup as a foundation for instance, secure threat intelligence exchange, dynamic infrastructure changes, or mathematically proving policy correctness [14][15]. As you can see in Fig. 1, Rudras splits into clean data, control, and observability planes that work together seamlessly.

II. Literature Review

A. Signature-Based IDS/IPS and Next-Generation Firewalls:

Open-source tools like Snort and Suricata were among the first widely adopted systems for spotting intrusions by scanning packets and protocol details through predefined rules [8][9][10][11][12].

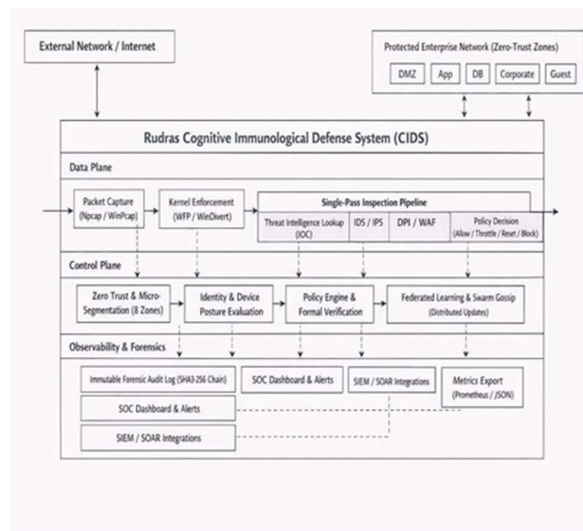


Fig. 1: High-level architecture of the Rudras Cognitive Immunological Defense System (CIDS), showing data, control, and observability planes.

Although tests reveal strong accuracy in detecting familiar attacks, their effectiveness drops sharply with new or disguised threats especially under heavy data loads [8]. Instead of working alone, they usually operate alongside external layers: firewalls, intelligence updates, and logging systems which often leads to tangled setups and mismatched security policies.

Every packet moves through a unified stream where threat checks happen at once signature analysis sits among more than forty combined methods. Rather than rely heavily on static rules, Rudras reduces blind spots tied to unknown exploits emerging before updates exist. Most next-gen firewalls stack features separately, yet here inspection flows within one continuous pass across detection systems. Signatures appear alongside web protection, real-time data feeds, and behavior-based risk assessment without breaking rhythm. Detection gaps shrink when multiple techniques overlap instead of standing alone.

B. AI-Driven Intrusion Detection and Anomaly Detection:

Because static signatures fall short, researchers turned to machine learning for spotting intrusions, relying on measures like traffic summaries, communication traits, or encoded packet contents. Instead of traditional methods, techniques including Isolation Forest and autoencoders process standard test sets from older benchmarks like NSL- KDD to newer smart device data catching unknown attacks without complete labels [5][6][7][13]. While LSTM networks and one-class SVMs perform well in these settings, success often hinges on how thresholds are set. Detection of forced-entry attempts and odd behaviors improves under unsupervised setups, yet shifts in network behavior over time still complicate real-world use [3][5].

Beginning with behavior patterns, Rudras uses a mix of artificial intelligence methods. Each IP and process holds a fixed reference point that does not change over time. Flow data gets updated constantly using Exponential Moving Averages, adjusting sensitivity through layered alert levels. When activity shifts too far from normal, automated reactions begin. On top of this foundation, user and entity behaviors are mapped individually. Detection of sideways network spread relies on graph-based neural models. Learning across devices happens without sharing raw data, thanks to decentralized training techniques. Modern intrusion systems recommend such strategies. What sets this apart is how deeply these tools are woven into the system's core actions not sitting outside as add-ons.

C. Zero Trust Architecture and Micro-Segmentation:

Starting with distrust inside network boundaries, zero trust defined by NIST SP 800-207 requires verifying each access attempt individually, relying on user identity, device status, and ongoing data streams instead of location-based signals like IP or VLAN tags [2][18][19][20]. Although many enterprise-grade tools and cloud-focused security systems apply these ideas using identity-conscious gateways, fine-grained segmentation, and persistent verification loops, they typically function as add-ons coexisting with legacy firewall setups [20].

Because Rudras works down to the packet layer, it applies zero trust through calculated device trust levels paired with an eight-part segmentation model. Each connection gets assessed using code-based identification alongside how recently updates arrived, active flaws detected, shifts in typical actions, and past interactions. Decisions unfold inside the operating system itself, guided by live firewall adjustments and responsive data protection checks shaping traffic flow. Session after session, verification repeats no assumed safety ever sticks around long. Trust never stays fixed; access tightens or loosens based on constant monitoring inputs. Every check supports core principles outlined by NIST: minimal permissions granted only when needed, repeated automated inspections run nonstop, plus rules adapting fast when signals change.

D. Research Gaps Addressed by Rudras:

Instead of tight integration, newer intrusion detection systems using artificial intelligence tend to keep monitoring and response steps apart - machine learning operates separately, passing findings to analysts or automated security tools [17]. In similar fashion, zero-trust setups frequently split core functions like user authentication, rule evaluation, and access control across distinct modules, creating gaps due to mismatched settings. Few efforts aim at bringing together secure memory practices, built-in behavioral analysis powered by AI, mathematically sound policies, encrypted collaboration on threats, and resistance to future quantum attacks within one system mature enough for real-world deployment [26].

One way Rudras addresses this space is through a standalone Rust executable combining intrusion detection and prevention, zero trust controls, anomaly spotting, decoy systems, audit trails, besides record keeping - all tied directly to published studies or established benchmarks. What sets it apart? It outlines fresh investigative paths: real-time clustering blended with probability modeling to catch unseen traffic patterns, dynamic defense layered with trap networks, encrypted matching methods for threat data exchange, along with alignment of verified policies and machine-driven threshold adjustments. Far from just another tool, Rudras opens doors serving as a testbed where next-gen defenses can grow past classic perimeter models.

III. Stages of Preparation

Problem Definition and Threat Modeling:

Beginning the process means facing how old firewall methods fail against today's hidden attacks like advanced persistent threats, ransomware, or internal risks that thrive inside encrypted networks. What stands out is a basic imbalance: breaking in takes one lucky attempt for hackers, yet protection demands total coverage, a challenge made worse when systems rely only on fixed signatures [16]. Instead of reacting blindly, the system builds profiles of likely intruders from novice tinkerers to state-backed groups and even those ready for post-quantum exploits and links each type to specific behaviors and defenses.

A detailed threat model lists potential threats like large-scale denial-of-service, command-and-control signaling, repeated login attempts using stolen credentials, spreading between network areas, misuse of industrial communication protocols, data theft hidden within encryption noting how each connects to specific defenses. Instead of relying on one layer, it spreads detection

and response through intrusion monitoring systems, adaptive algorithms, shared threat insights, user behavior analysis, tools that track sensitive information, and strategies that shift system configurations unpredictably. Rather than a static blueprint, the framework shapes what components are required, defines their coordination patterns, pinpoints where overlapping safeguards make breaches harder to carry out unnoticed.

Architectural Principles and Design Requirements:

Getting ready means checking every Rudras part against current studies and guidelines so accuracy and growth stay possible. Because they rely on Snort and Suricata logic, along with MITRE ATT&CK links, intrusion detection tools hold firm ground. With DNS, messaging, and operational tech safeguards, rules come alive through RFC documents, industrial communication blueprints, and up-to-date methods [8][9][21]. When trust is assumed never, frameworks lean into NIST SP 800-207 plus identity management benchmarks; encryption pieces instead draw from NIST's quantum-safe drafts and PKCS models [2]. Because of this careful structure, Rudras functions not through random rules, yet as a precise system built for testing and growth within both university and company settings.

IV Stages of Construction

Core Infrastructure and Packet Pipeline:

Starting off, construction centers on a strong foundation for handling packets that includes capturing them, following their paths, and managing active states. Instead of generic tools, Rudras links up with Npcap or WinPcap to grab traffic, while leaning on WFP and WinDivert to control rules at the system level and catch data in user space. Within this setup, several tasks run separately but together: one thread captures incoming data, another routes it onward, others handle intrusion detection, update artificial intelligence models, pull updated threat feeds, report performance stats, and watch running processes - all powered by Tokio's async engine. Communication between these pieces relies on fast, lock-free message passing alongside lightweight synchronization designed for frequent reading.

Beginning at Layer 2, each packet enters a fixed sequence where faulty ones get dropped early. Following that step comes identification via threat data checks instead of waiting until later phases. Signature analysis occurs alongside deep packet inspection or web application filtering depending on traffic type. Behavior-based scores generated by machine learning influence outcomes but do not control them outright. Decisions made at the endpoint include permitting transit, slowing delivery speed, cutting connections, or rejecting entirely. Every examination happens once only content gets analyzed one time then shared forward. Parsing tasks aren't repeated because results feed every detection module downstream.

This method skips inefficiencies found in layered systems doing similar work more than once. Take a look at Fig. 2a it shows how every packet zip through capture → validation → IDS → DPI/WAF → AI scoring → IPS, all in one smooth pass and Fig. 2b captures a live Windows run: packet capture kicking off, kernel rules loading, IDS/IPS humming, AI engine online, and real-time logs scrolling.

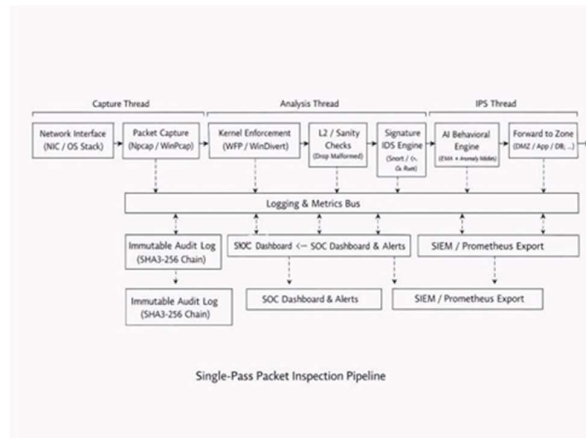


Fig. 2a: Rudras packet processing pipeline: each packet passes once through capture, fast-path validation, threat intelligence, IDS, DPI/WAF, AI behavioral scoring, and unified IPS enforcement, with structured logging at every stage.



Fig. 2b: Rudras runtime instance on Windows, showing initialization of packet capture, kernel enforcement, IDS/IPS, AI behavioral engine, and zero-trust policy modules, followed by live packet processing logs.

Detection Engines and AI Subsystem:

Following construction steps add signature-driven intrusion detection and prevention systems, introducing rule interpretation alongside structured grouping by protocol behavior. A Bloom filter screens alerts early, while dynamic thresholds apply using moving time windows for traffic volume checks. To sustain packet handling within sub-millisecond delays, certain rules target specific protocols, reducing unnecessary evaluations. Simple tests run first to skip heavier analysis when possible. Counters tracking individual IP addresses rely on concurrent buffer designs that avoid locking bottlenecks.

At the same time, the AI component tracks fixed behavior patterns using Exponential Moving Averages on indicators like packet volume, byte throughput, variety of ports, session lengths, and entropy adjustable tolerance levels trigger responses in the intrusion prevention system [5][6]. Built into this layer are user and entity behavior analytics, enabling profile creation; graph-based neural models spot unusual network traversal by accounting for structure, while decentralized learning allows models from different nodes to combine without moving sensitive data to a central point [14][15][30][31][32]. Fig. 3 breaks down our behavioral baseline magic using EMAs to track normal vs. weird activity, then escalating from 'watch' to 'block' based on thresholds.

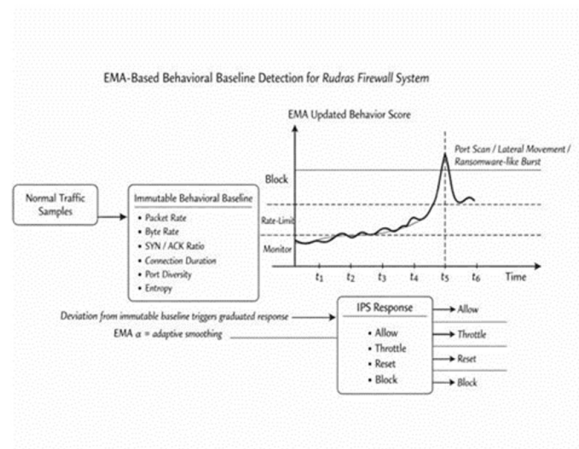


Fig. 3: Rudras behavioral baseline and EMA-driven anomaly detection, showing threshold-based escalation from monitoring to rate-limiting and blocking.

4.3- Zero Trust, Micro-Segmentation, and Enterprise Modules:

Enterprise security layers build around functional zero trust practices along with policy enforcement structures. Eight standard areas emerge under Rudras such as finance, research, guest access, databases, applications, DMZs, corporate networks, and administrative systems each isolated through strict

interaction logic where only approved cross-zone traffic moves forward. Rules apply based on who accesses resources: identity data, machine health ratings, service account tags, plus situational details pulled from authentication artifacts shape what is allowed instead of fixed network locations. Unexpected pathways fail unless explicitly cleared beforehand.

Though less common, some tools check device health before granting network access. One feature tracks signs of intrusions by analyzing behavior patterns across systems. Connection to security event platforms happens through standardized interfaces. Protection inside containerized environments relies on tight coupling with orchestration layers. Rules for wide-area networking adjust automatically based on traffic origin. Reports needed for audits follow structures defined in global standards. Experimental components include fake assets meant to mislead attackers.

Shifting addresses and ports at intervals makes targeting harder. Indicators of compromise can be shared without exposing raw data. Mathematical proofs help confirm policies behave as intended. Hardware roots of trust validate machine identity securely. Early-stage encryption methods anticipate future quantum attacks [22][23][24][25][26].

V. Testing Methodology

Functional and Module-Level Testing:

From the start, testing focuses on individual components, unit checks and combined interactions verify how well packets are parsed, rules assessed, scores calculated by AI, then policies applied, using both generated and captured network flows. Detection logic within IDS/IPS systems gets tested through selected rule collections and repeated attack scenarios, ensuring reliable identification of probing behavior, floods, web-based intrusions, malformed protocol usage, internal spread attempts, remote control signals, unauthorized data transfers, login breaches, malicious software operations, irregularities in encrypted streams, threats targeting cloud environments, alongside industrial system risks all mapped to categories outlined in the Rudras framework.

Though benchmark datasets guide part of the evaluation, replayed network flows also shape how anomaly scores respond across shifting conditions. Instead of relying solely on static inputs, testing includes scenarios where concept drift emerges alongside blended patterns of normal and hostile activity.

Threshold mechanisms adapt accordingly, while federated aggregation undergoes scrutiny through varied edge-node interactions. Separately, synthetic policies populate the verification pipeline each crafted with deliberate overlaps, hidden rules, or contradictions. These test cases expose whether flaws in access logic get detected early. Before any rollout occurs, the system must flag such inconsistencies reliably.

Performance and Stress Testing:

Despite heavy loads, Rudras handles packets efficiently measured through delay times, data volume moved, and how much computing power it uses. Mixed scenarios test its stability: sudden surges of SYN and UDP requests appear alongside fast HTTPS calls and massive streams of DNS plus mail activity. When faced with intense TLS attacks requiring complex math, the system adapts. Inspection depth drops only where needed, switching targeted flows to lightweight

hash checks instead of full analysis. This shift prevents overload, keeping core functions alive even during assault-like conditions.

Under high load, competing threads and distributed gossip processes reveal how locks create delays, backed by metrics tracking wait times, task queues, spinning operations, plus bottlenecks tied to frequent reading under shared access controls. The reliability of memory protection, rooted in Rust's compile-time checks, emerges from systematic code probing and structure reviews that expose hidden flaws like out-of-bounds writes or race conditions during message processing. Check Fig. 4: even with mixed legit + attack traffic, adaptive load shedding keeps latency way down during stress tests.

Scenario-Based Evaluation:

Starting with real-world setups, Rudras undergoes testing through complex situations where several components run together. One case involves shielding an external API gateway from automated login attacks alongside SQL injection attempts. Another scenario strengthens development interfaces by guarding against tainted software libraries and unauthorized remote access tools. Detection of internal users secretly transferring sensitive data out of restricted database areas also forms part of the evaluation process.

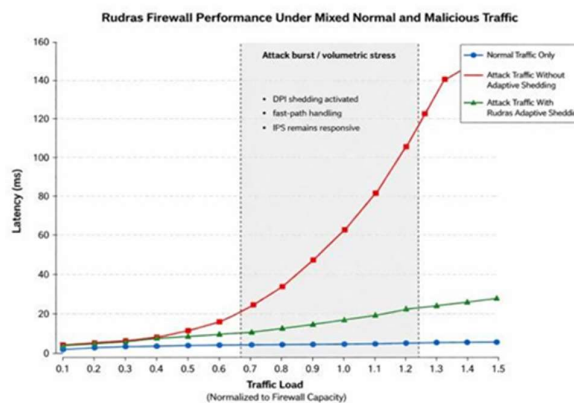


Fig. 4: Rudras performance under mixed normal and malicious traffic, showing the effect of adaptive load shedding on latency during stress testing.

Although the API gateway setup uses DPI/WAF together with adaptive load shedding, it also brings in threat intelligence alongside SOAR to spot heavy login rejections and code injections then quietly isolates harmful origins while alerting investigators. When developers interact at endpoints, protection emerges not just from DNS safeguards but also through behavioral analytics spotting links to command-and-control hubs or strange outward traffic; at the same time, running processes reveal forbidden inspection or interference utilities.

During internal data leaks, control shifts toward tight network segmentation working with user behavior tracking plus data loss prevention, halting odd database queries across zones and stopping confidential file movements all recorded thoroughly for later review. Fig. 5 sketches our testbed setup: attackers hitting clients, firewall in the middle, protected services behind it, SOC watching everything

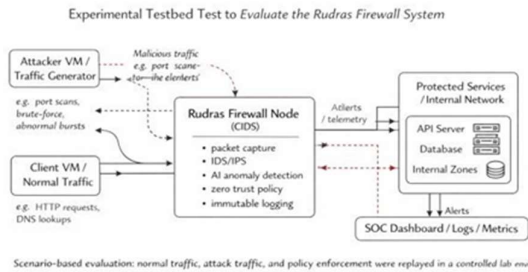


Fig. 5: Rudras experimental testbed used for scenario-based evaluation, showing attacker and client traffic, the firewall node, protected services, and SOC telemetry.

VI Results and Discussion

Qualitative Security Outcomes:

What makes Rudras stand out isn't just volume, it's how defenses interact. Instead of stacking tools, it weaves them so gaps in one layer get covered by another. When attackers slip past static rules, behavior analysis picks up subtle shifts in activity patterns. Rather than depend on known attack fingerprints, the system uses focused protocol inspectors that spot odd formatting or timing mismatches. Deception elements add noise from the attacker's view, making reconnaissance risky.

Even if models miss something, checks based on mathematical logic catch flawed policies before they go live. Logs written once stay unchanged, removing tampering chances after incidents. During tests, suspicious login bursts got flagged even when IPs looked clean. Command-and-control chatter was cut off mid-beacon because traffic rhythms broke norms. Movement across systems triggered alerts not through single clues but pattern clusters. Data dumps slowed automatically, regardless of whether thresholds were crossed yet. Each safeguard alone might fail; together, they create resilience without relying on perfect prediction.

Starting from the ground up, packet-level zero-trust controls paired with user behavior analytics and fine-grained segmentation cut down lateral spread when devices or accounts are breached, fitting current security frameworks [2][18]. Instead of fixed rules, live assessments of device health, ongoing trust checks, and shifting policy triggers keep permissions aligned with real-time threats. Built on SHA3-256 hashed logs and organized data streams, the system leaves behind tamper-resistant records valuable during investigations, regulatory reviews, or shared studies across institutions. Fig. 6 sums up four real attack scenarios triggers, what caught them, responses fired, and final outcomes.

Scenario-Based Evaluation Results for Rudras

Trigger	Detected by	Response	Outcome
Port Scan Port scan, brute force, abnormal burst, C2-like	→ IDS signature + IP block	→ Scan stopped	→ Scan stopped
	→ Rate-limit + alert	→ Login attempts slowed	→ Login attempts slowed
Brute-Force Login Detection	→ Zero trust policy + micro-segmentation	→ Movement contained	→ Movement contained
Lateral Movement Containment	→ UEBA + DLP alert	→ Suspicious outbound traffic blocked	→ Suspicious outbound traffic blocked
Data Exfiltration Prevention	Each scenario was replayed in a controlled lab environment using the experimental testbed.		

Fig. 6: Scenario-based evaluation results summarizing attack triggers, Rudras detection modules, responses, and outcomes across four representative security events.

6.2- Engineering Benefits of Rust and Unified Architecture:

Security gains emerge when building Rudras fully in Rust, contrasting older C or C++ firewall systems through early detection of buffer overflow issues during compilation. Memory demands behave predictably since no runtime garbage collector intervenes, a shift that sidesteps disruptive lags common in managed environments. Concurrency hazards like data races drop sharply within packet-handling routines thanks to strict compile-time checks on shared state access. Under traffic floods, such traits matter unmanaged memory flaws or GC-induced stalls might otherwise trigger crashes or open exploitation paths. Evidence supports these claims, anchoring them in measurable system behavior rather than theoretical advantage alone [27].

One system tie together intrusion detection, prevention, artificial intelligence, zero trust, data protection, threat intel, and logs through a single shared framework cutting setup complexity when contrasted against patchworks of third-party tools requiring isolated hardware. Fewer mismatched settings appear over time because components update in sync instead of relying on shaky connections between external platforms. Policies stay consistent across locations where security decisions take effect, especially vital as environments mix cloud, on-premise, and edge setups. This alignment matters more now that networks span unpredictable configurations.

Limitations and Future Work:

Even with wide coverage, Rudras misses some areas where new study and development could take hold. Though features like kernel enforcement exist today, others such as eBPF/XDP offload, P4 switch links, BGP hijacking alerts, deep QUIC/HTTP3 checks, protection at Wi-Fi's second layer, and scrutiny of hidden traffic remain planned or under construction. What's more, homomorphic threat data exchange and shared machine learning have entry points built in; yet these tools demand stronger shields against smart algorithmic threats. Testing them across vast networks involving many groups has not happened fully either.

So far, assessments have leaned heavily on simulated scenarios alongside checks of individual components. Moving ahead, studies ought to bring in

structured testing using openly available datasets instead. Controlled trials meant to expose weaknesses might follow, paired with rigorous proof of security across full systems when layers interact. Another angle: methods that clarify AI decisions could help justify unusual findings. These same tools may guide policy choices too. Operators in monitoring hubs would likely feel more confident relying on such outputs. Mental strain during active threat tracking could ease as a result.

VII Conclusion

Because Rudras builds a single firewall platform rooted in artificial intelligence and zero-trust principles, it manages memory safety without slowing down data flow. Not only does it run at full wire speed, but also supports deep customization for experimental needs. Through integration of a biology-inspired protection scheme, more than forty layered detection methods, and AI tied to observable actions, gaps in classic firewalls fade. Instead of relying on isolated intrusion systems, this design uses fine grained segmentation alongside mathematically checked rules. After every event, unchangeable logs preserve evidence exactly. From its structure to real code and planned studies, progress toward shared encrypted threats becomes tangible. Even distributed detection across organizations gains clarity. Shifting defense positions adapt fast and security no longer sits still. Verified stacks ready for quantum shifts now have working paths forward [26].

What stands out about Rudras is how it shapes security systems research during the late 2020s through clear framing. A precisely outlined threat model sets the stage early on. Its design follows established principles while aligning closely with recognized standards. The system runs in live environments yet allows flexibility for experimental adjustments. Evaluation unfolds around realistic scenarios, focusing on how actual operators would interact. Open challenges are laid bare at the end, encouraging others to build further. This approach gives readers a path forward without dictating next steps.

REFERENCES

- [1]. A, Deepak P S, Rudras: The Firewall in Practice – v4.0.0 Architecture and Module Documentation, technical manual, 2026. [Rudras_ The Firewall in Practice.pdf](#)
- [2]. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, Zero Trust Architecture (NIST SP 800-207), Gaithersburg, MD, USA: NIST, Aug. 2020, doi: 10.6028/NIST.SP.800-207. <https://csrc.nist.gov/pubs/sp/800/207/final>
- [3]. A. A. Laghari, G. Mustafa, A. S. M. Alghamdi, and S. M. K. Quadri, “A novel and secure artificial intelligence enabled zero trust intrusion detection in industrial internet of things architecture,” *Sci. Rep.*, vol. 15, art. 26843, Jul. 2025. <https://pmc.ncbi.nlm.nih.gov/articles/PMC12287373/>

- [4]. Y. Tong, "The Application of Artificial Intelligence Technology in Big Data Network Security Defense," Int. J. Advance Appl. Sci. Res., vol. 5, no. 1, pp. 12–16, 2026. Link: <https://www.h-tsp.com/index.php/ijaasr/article/download/226/226>
- [5]. M. B. Al-Doori and K. M. A. Alheeti, "AI-Driven Features for Intrusion Detection and Prevention Using Random Forest," J. Cybersecurity Inf. Manag., vol. 16, no. 1, pp. 1–14, 2025. Link: <https://www.americaspg.com/article/pdf/3586>
- [6]. S. Subi, K. Kalaiselvi, P. K. Devi, and S. K. Karthika, "AI-Based Cyber Threat Detection Using Isolation Forest Int. J. Innov. Res. Technol. (IJIRT), vol. 12, no. 10, pp. 7397–7401, Mar. 2026. Link: https://ijirt.org/publishedpaper/IJIRT195146_PAPER.pdf
- [7]. O. Mykhaylova, A. Shtypka, and T. Fedynyshyn, "An Isolation Forest-based approach for brute force attack detection," in Proc. 1st Int. Workshop on Bioinformatics and Applied Information Technologies (BAIT 2024), CEUR Workshop Proc., vol. 3842, 2024. Link: <https://ceur-ws.org/Vol-3842/paper3.pdf>
- [8]. "Performance and efficacy of Snort versus Suricata in intrusion detection: A benchmark analysis," AIP Conf. Proc., vol. 3232, no. 1, art. 020024, 2024. https://pubs.aip.org/aip/acp/article/3232/1/020024/331661_1
- [9]. A. Imran and R. Chitchyan, "A Comparative Analysis of Snort 3 and Suricata," Univ. of Portsmouth, Tech. Rep., 2023. Link: https://pure.port.ac.uk/ws/portalfiles/portal/79753845/A_Comparative_Analysis_of_Snort_3_and_Suricata.pdf
- [10]. M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in Proc. 13th USENIX Syst. Admin. Conf. (LISA), 1999, pp. 229–238. Accessible via USENIX/Snort archives; commonly cited without direct open URL.
- [11]. V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," Comput. Netw., vol. 31, nos. 23– 24, pp. 2435–2463, 1999.
- [12]. (Publisher access via Elsevier/ACM; widely known Bro/Zeek paper.)
- [13]. E. Bou-Harb and A. Mangino, "Zeek (Bro) Intrusion Detection System (IDS)," Research Note, Univ. of South Carolina, 2019.
- [14]. Link: [https://research.cec.sc.edu/files/cyberinfra/files/Bro%20Intrusion%20Detection%20System%20\(IDS\).pdf](https://research.cec.sc.edu/files/cyberinfra/files/Bro%20Intrusion%20Detection%20System%20(IDS).pdf)
- [15]. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in Proc. 8th IEEE Int. Conf. Data Mining (ICDM), Pisa, Italy, Dec. 2008, pp. 413–422.
- [16]. Link: <https://ieeexplore.ieee.org/document/4781136>
- [17]. H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proc. AISTATS 2017, pp. 1273–1282.
- [18]. Link: <https://arxiv.org/abs/1602.05629>
- [19]. D. Nikolić, "Federated learning-based intrusion detection for 6G-enabled IoT networks," Arch. Tech. Sci., vol. 74, no. 4, pp. 33–49, 2025.
- [20]. Link: <https://www.arhivzatehnickenuke.com/article/792>
- [21]. Girish H., Srinivas S., Kavya M., ...Kabbur S. "Swift Autonomous Fire Suppression Robot", 16th International Conference on Advances in Computing Control and Telecommunication Technologies Act 2025

- [22]. Girish H., Soujanya S., "A Comprehensive Cadence-based Design Flow for RISC-V UART SOC IN SCL 180NM", 16th International Conference on Advances in Computing Control and Telecommunication Technologies Act 2025
- [23]. Girish H., Rakesh S., Prasanna Raj C.P., Singh A.S., Yathisha L, "Deep Learning for Diabetic Retinopathy: Automated Detection and Classification from Retinal Images", 16th International Conference on Advances in Computing Control and Telecommunication Technologies Act 2025
- [24]. Girish H., Shyam Sundar V., Priyanka C., Renuka Hebballi R.S, "RISC V Steel Processor Implementation for Communication between Two Chips using AXI Lite", 16th International Conference on Advances in Computing Control and Telecommunication Technologies Act 2025
- [25]. M. H N, G. H and N. K N, "Simulation and Verification of SPI Master and Slave Communication," *2025 6th International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, 2025, pp. 1531-1534, doi: 10.1109/ICOSEC67334.2025.11459684.
<https://ieeexplore.ieee.org/document/11459684>
- [26]. S. S. V, P. C and G. H, "Design and Analysis of a D-Band LC Bandpass Filter and Low-Noise Amplifier for 6G Front-End Applications," *2026 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2026, pp. 1-6, doi: 10.1109/IATMSI68868.2026.11466038.
<https://ieeexplore.ieee.org/document/11466038>
- [27]. "A Comprehensive Cadence-based Design Flow for RISC-V UART SOC IN SCL 180NM" , Girish H, Soujanya, Grenze International Journal of Engineering and Technology, 2025
- [28]. "Swift Autonomous Fire Suppression Robot" Girish H et. Al., Grenze International Journal of Engineering and Technology, 2025
- [29]. "Deep Learning for Diabetic Retinopathy: Automated Detection and Classification from Retinal Images" , Girish H, Cyril Prasanna Raj P , Grenze International Journal of Engineering and Technology, 2025
- [30]. "RISC V Steel Processor Implementation for Communication between Two Chips using AXI Lite" Shyam Sundar V, Dr Girish H, Priyanka C, Renuka S Hebballi Grenze International Journal of Engineering and Technology, 2025
- [31]. Sujatha.S, Mahendra BM, & Dr. Girish H. (2024). Smart Irrigation System for Optimal Water Conservation. *Revista Electronica De Veterinaria*, 25(1), 2613 - 2619. <https://doi.org/10.69980/redvet.v25i1.1311>
- [32]. Mamatha K R, Thejaswini S, Rashmi N, & Dr. Girish H. (2024). Laser Fencing Surveillance System With Email Alert. *Educational Administration: Theory and Practice*, 30(2), 1539–1544. <https://doi.org/10.53555/kuey.v30i2.7919>
- [33]. Thejaswini S, Mamatha K R, Rashmi N, & Dr. Girish H. (2024). Performance Evaluation of AI Models in Early Heart Disease Diagnosis. *Educational Administration: Theory and Practice*, 30(2), 1531–1538. <https://doi.org/10.53555/kuey.v30i2.7918>

- [34]. Sushma K Sattigeri, Seema Srinivas, Shridhar Kabbur, Girish H, "Revolutionizing Urban Waste Management: An AI-Powered Autonomous Solution for Smart Cities", Tuijin Jishu/Journal of Propulsion Technology ISSN: 1001-4055, Vol. 45 No. 02 (2024)
- [35]. Girish H, Vajjala Sai Sree Hari,Ponde Lakshmi Narasimha,P N Shreya,Dadigala Bhargavi, "THE DESIGN AND VERIFICATION OF A 32-BIT RISC V PROCESSOR USING VEDIC MATHEMATICS", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.12, Issue 5, pp.1130-1135, May 2024, Available at :<http://www.ijcrt.org/papers/IJCRTAB02172.pdf>
- [36]. IoT based home automation using Blynk cloud, J. Chetan Naik, B. S. Deeksha, H. Girish, Gurusharan Kashetty, T. G. Shivapanchakshari, Manohar Sai Polakam, Journal: AIP Conference Proceedings, AIP Conf. Proc. 2965, 020006 (2024), <https://doi.org/10.1063/5.0215427>, Published: July 2024
- [37]. IoT based fire alarm system using Blynk app, T. Bhargava, H. Kishor Kumar, Pramod Mahalingappa Godi, H. Girish, J. Chetan Naik, Journal: AIP Conference Proceedings, AIP Conf. Proc. 2965, 020007 (2024), <https://doi.org/10.1063/5.0212325>, Published: July 2024