International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

# Real-Time DDoS Attack Detection and Mitigation Techniques

Mohammad Rizwan Ahmed
Bachelor of Engineering
Student
Cyber Security Department
ACS College of Engineering
mdrizwan24824@gmail.com

Sweta Pathak
Bachelor of Engineering
Student
Cyber Security Department
ACS College of Engineering
sweta.pathak20268@gmail.com

Syed Hammad Yaseen
Bachelor of Engineering
Student
Cyber Security Department
ACS College of Engineering
syedhamyaseenmay6th@gmail.com

Ms Sanjana R
Orcid ID :0009-0003-3988-8326 Guide
Cyber Security Department ACS
College of Engineering
Bengaluru-560074
sanjanachavan98@gmail.com

***Abstract- When a flood of data hits fast, sites go down, turning away everyday users. Hackers cause these jams by overwhelming systems with excessive traffic. Spotting them used to rely on methods that move slower than the threat itself. By the time warnings sound, attackers have already pushed further. Responses happening right then still miss the moment damage begins.***

***A tiny setup thrives when rules are sharp - it catches glitches quick, moves right after, skips hesitation, trims wait times, guards the flow. Simple guts mean swift steps, choices line up one by one, safety holds firm thanks to clever pauses between actions***

***Keywords— DDoS Attack, Real-time Detection, Network Traffic Monitoring, Packet Flow Analysis, Wireshark, Tshark, Python Script, Firewall Mitigation, Rule-based System.***

## I. INTRODUCTION

Crashing into servers, a flood of data blocks everyday users from getting through. When systems drown under overload, attackers slip in unnoticed. Speed becomes useless if machines choke on too much at once. By the time warnings pop up, the breach has already moved deeper. Response may be fast, yet still trails behind the opening move.

A tiny setup runs smooth when rules click right - catching glitches quick, adapting on the fly, dropping doubts, cutting dead time, holding the beat steady. Simple guts mean swift steps, choices pile clean, defense holds firm since clever pauses nest between every step. From the outer layer, firewalls along with tools like iptables create a first line of defense, giving administrators quick control to block dangerous traffic. Since they restrict how deep breaches go, these setups reduce exposure by isolating affected areas using address-based policies. Moving one level deeper, apps such as Fail2Ban jump into action - reading log files to detect repeated attempts and blocking those sources automatically. With responses triggered by behavior, protection becomes faster, leaving less time for external risks to succeed.

Halfway into today's network designs, Zeek - formerly known as Bro IDS - works alongside Grafana and Prometheus to monitor hidden activity. Rather than sit back until issues arise, security staff catch strange patterns in traffic using straightforward visuals from these systems. A sudden change triggers notifications, pushing teams to act before failures occur. Since they link straight into incident handling steps, protections evolve gradually, building up without requiring round-the-clock control.

Something becomes clear from studying the data – counting on a single approach fails when DDoS threats grow more complex. Layering multiple defenses brings better results. Take Wireshark, it inspects individual packets.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Traffic flow gets shaped by iptables. Repeat attackers? They're shut down automatically thanks to Fail2Ban. System activity stays visible through dashboards built in Grafana. Then there's Zeek, Suricata, and Snort spotting odd patterns inside protocols. Together they respond fast. Efficiency remains high during operation. Scaling isn't a problem either. Networks hold steady under strain. Even during active assaults service continues without interruption.

## II. PROBLEM STATEMENT

Without warning, sudden surges crash into digital networks. Not smooth streams but waves - rushing, relentless - overload machines until they freeze. When capacity drowns, users find doors shut tight. A few bits might slip past; most simply vanish mid-air. The chaos isn't random - it's shaped, aimed, forced into motion by design. Suddenly, the flow stops when overload hits without warning. Real needs vanish beneath tides built for nobody.
Faster changes catch rigid systems off guard - human monitoring can't scale when traffic spikes without warning. Lags appear right away once pressure builds, giving problems time to spread well ahead of any response. Sudden jumps overwhelm fixed rules, leaving gaps until someone steps in.

Now, catching DDoS dangers quickly depends on constant flow checks. Odd signs trigger instant moves by the machine. Rather than pause, rules built into firewalls cut off bad streams. That way, responses come faster than they used to. Breaks in service get shorter, occasionally disappearing completely. Floods of fake requests come crashing in, yet the entire network holds firm. Staying online through it all, automated checks slowly toughen the system over days.

## III. RESEARCH OBJECTIVES

- The main objective of this project is to develop an automated, rule-based system capable of detecting and mitigating Distributed Denial-of-Service (DDoS) attacks in real time. The system is designed to continuously monitor incoming network traffic, analysing data flow patterns to quickly identify sudden surges or suspicious request behaviours that typically indicate distributed attack activity.

- By implementing customizable rules and adaptive thresholds, the proposed system can distinguish between normal user traffic and potentially harmful flows, allowing it to adjust dynamically as network conditions and attack strategies evolve. When a potential DDoS attempt is detected, the system initiates immediate countermeasures—such as blocking malicious IP addresses, limiting excessive connection rates, or modifying firewall configurations—to preserve network performance and availability.

- The broader aim is to ensure that legitimate users experience minimal disruption while strengthening overall network resilience and security with little to no manual intervention. This approach not only addresses traditional DDoS attack patterns but also establishes a scalable and adaptable framework capable of responding to emerging threats in the ever-changing digital landscape.

## IV. LITERATURE REVIEW

### *A. Overview of Existing Solutions*

Brief look at what's already out there.Yesterday's ways of catching DDoS bursts have shifted - yet echoes remain in today's shields. Spotting flows by comparing them to stored attack shapes forms the base, tagging danger once similarity clicks. Tools such as Snort or Suricata run hard on this method, snapping alerts at familiar rhythms. Yet gaps open when unseen moves slip through - delays let unknowns pass while defenses wait for new rules [1, 8, 14].

Most times, folks pick threshold-based detection because it shows up everywhere. Once numbers break past fixed limits - say, packet volume, bandwidth consumption, or open links - the alarms go off. A spike in network activity? That kind of surge usually trips the signal quick. What pulls people in is how straightforward it feels; being light on resources helps too. Funny how regular intense work sometimes gets flagged as risky. That mix-up reveals the need for quick tweaks, guidelines that stay loose as days go by [25, 16, 17].

Something feels off? That's often what alerts detection software scanning network behavior. Over time, certain platforms pick up on regular communication rhythms across systems. Tools like Zeek - once called Bro IDS - focus heavily on protocol-level movement, logging interactions in a distinct way. When activity drifts far from


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

the norm, it draws notice. Even brand-new oddities get flagged without prior examples. Finding fresh threats could happen sooner now. Behind the scenes, good performance needs precise adjustments paired with powerful machines. Tiny teams often hit walls trying to run things without hiccups. As networks expand, so does hunger for processing power. Keeping steady watch on activity patterns feels out of reach for some groups [15, 5].

*B. Machine Learning Approaches (Brief Mention)*

Lately, research has shifted toward using machines that learn, simply because these systems notice odd changes during attacks more easily. As time passes, they grow sharper at finding new dangers since they pull clues straight from live traffic patterns. Yet they often demand powerful hardware, oceans of past examples, along with careful choices about which signals matter most. Clarity takes priority - instead of stacking layers of complexity, this approach follows straightforward logic that reacts quickly while staying light on power. [4, 10, 21].

*C. Tools and Techniques*

Watch closely how data moves across networks, tools like Wireshark or Tcpdump help uncover odd patterns. Live monitoring gives instant views, while past logs reveal slower unfolding events just as well. When traffic splits into small parts, strange actions become harder to miss. The starting points of breaches usually appear once someone digs deep into those fragments [25].

Right after delivery, programs such as Snort and Suricata begin checking data constantly - comparing every fragment to known danger profiles. Once matches appear, warnings fire off instantly. Speedy identification means responses happen just as fast. Running on traces from earlier breaches, these setups catch duplicates early. Warnings show up not due to strangeness, rather because the behavior was logged long ago. Recognition comes from memory, not guesswork. Through these channels, vehicles move without stopping, one after another. Spotting what matters means checking every piece against the real thing [18].

Logs shaped by protocol habits make network events easier to see when using Zeek (Bro IDS). Not limited to familiar signatures, it notices strange actions through thoughtful analysis that catch new risks slipping past most systems [1, 17, 15].

Staying safe online sometimes means blocking trouble before it gets close - tools such as iptables do this by sorting incoming data through fixed rules. Watchful software like Fail2Ban steps in once break-in attempts pile up, shutting down access on its own after spotting suspicious patterns in log files [23].

Right away, live updates pop up using software like Grafana and Prometheus. Since everything displays in real time, strange behavior shows itself earlier. Odd trends? They trigger warnings that let teams understand faster - no more waiting around. Watching traffic move second by second changes the whole approach to fixing problems [24, 22].

*D. Identified Research Gap*

Today’s methods for blocking DDoS attacks have gotten better, still they lack accuracy and quick response since devices end up doing more work than required. Fixed-pattern systems spot known dangers only but rely heavily on updates just to remain effective. Threshold-based rules often mistake regular traffic surges for threats instead. Behavior-focused detection consumes excessive computing strength meanwhile demands fine adjustments tough to maintain under limited capacity. Tracking handled in one place adds delay plus raises risk when any component breaks down [15].

When battling DDoS attacks head-on, a rule-based setup works clearly. Fast movement with low energy need stands out, meaning Snort pairs smoothly with iptables and Zeek. Because choices come from fixed logic, spotting threats takes almost no time. The instant risk appears, reactions fire off automatically - no delays. Simplicity wins speed, leaving behind heavy systems that drag performance down. Here, quick response walks hand in hand with lean design. Out of tangled webs, it chooses clear routes shaped by familiar designs. Because every section sticks to strict limits, safety grows without extra effort. With little drag between parts, speed holds steady. It steps into a gap left open - going beyond seeing trouble toward moving ahead of harm. Past work backs its roots in practice, not guesswork.[19,20]


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

## V. PROPOSED SYSTEM

### *A. System Architecture*

A shift follows - into the Analysis Module, where Python-powered tools such as Scipy take hold of gathered details. Rather than staring at bare figures, the system pulls apart individual packets, tracking exactly how quickly data flows through lines. When traffic suddenly surges, those moments jump into view here. Since trends are already mapped, finding problems down the road turns quicker, sharper. What results is a clearer path through clutter. Tightly connected pieces form a loop, so help travels either way, spotting danger faster. From one piece to the next it rolls, almost like linked rings, but together they shift at once when bad data slips in. Speed builds where gaps shrink, with messages leaping sharply across zones. Alerts fire off rapidly just because of shape - strength takes a back seat here. How things click together counts far more than brute force when time tightens.

Flying past, every chunk of data gets spotted instantly by the Traffic Capture Module. Live flows never stop pouring in, caught steadily using tools such as Wireshark or Tshark. Performance holds strong under watch - sluggishness simply doesn't show up. The moment packets roll in or head out, they land directly into stored files. Review work down the line depends completely on this unbroken flow. Midway through tracking activity, patterns begin to emerge based on preset guidelines that examine incoming details. Rather than assuming issues, the process compares live conduct with established red flags - like jumps in volume, irregular gaps in interactions, or unusual structure in messages. When present behavior echoes those signals, potential problems come into view. Control stays with operators, who can tweak or reset limits at any point without restriction. Fences hold steady, even as winds shift unpredictably. When pressure comes from new angles, they bend without breaking - adaptation happens silently, built into their bones.

One moment passes after a warning sounds - that is when the Mitigation Module wakes up. Moving without delay, it reshapes defenses on the fly: under Linux, iptable rules rewrite themselves, whereas Windows uses native firewall controls. Connections from harmful IP addresses disappear right away, often faster than they can react. Damage stops in its tracks, frozen halfway through. Operations continue smoothly, undisturbed. Silent corrections hold stability together, out of sight.

### *B. Workflow Explanation*

1. Right off, Wireshark or Tshark pulls live data - flow capture kicks in immediately. Each stream appears fast, seen across several interfaces while still moving. From the first packet onward, monitoring goes nonstop.

2. A chunk waits, just sitting there, then suddenly Python pries it open - attention shifts to how fast things move through, patterns of returning locations begin to surface, strange bits hidden inside parts rise up, looking for jumps without reason or surges that arrive out of nowhere in the flow.

3. Out of nowhere, odd data moves across the network. If that happens, the system looks for patterns tied to flood-style attacks. A single spot sending repeated signals stands out fast. Mistakes in how messages travel trigger warnings just as quick. 3. Something odd happens, someone spots it right away. When things repeat, they stand out quick. What shows up again tells the real story. Slowly, small signs add up into clear danger.

4. Out of nowhere, odd things start happening - posts shift on their own, flawed zones stop crawling halfway. Right when danger pokes through, blocks flicker awake, powered by steady watchfulness that stamps out trouble in seconds. Then silence.

5. Logs catch every moment. Right there, alerts sit beside blocked attempts, recorded the instant they occur. Since each entry mirrors real actions, reviewing them reveals precisely how things unfolded. If a single setting functions - or fails - it leaves hints about where tweaks might shift outcomes. Over time, choices build on past ones, gently shaping what follows. Each move forward leans on what came before. A trace remains when something comes through - not proof, but a sign of paths chosen. Tools shaped what arrived.The system integrates several widely adopted and reliable tools for performance, flexibility, and compatibility:

- Packet Capture: *Wireshark* and *Tshark* are used for real-time traffic monitoring and packet collection, offering deep protocol inspection and detailed packet-level insights.

- Traffic Analysis: *Python* forms the core of the analytical engine, utilizing libraries such as *Scapy* for packet parsing, *subprocess* for command execution, and *os* for system integration.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

- Mitigation and Défense: Automated firewall control is achieved through *iptables* on Linux systems and the *Windows Firewall API* for Windows-based deployments.

- Deployment Environment: The system can operate seamlessly on both Linux and Windows, providing adaptability for diverse organizational infrastructures.

## VI. IMPLEMENTATION ANALYSIS

- Phase 1: Environment Setup

  Off we go - sorting out the groundwork comes before anything else. Depending on the environment, you might see Linux here, a bit of Windows there. Packets need watching, so tools like Wireshark slide into place. Tshark joins the mix, making flow checks quick without losing depth.
  Packed with Python 3.x, scripts lean on Scapy to juggle network traffic - subprocess steps in for running commands, while os handles the groundwork behind the scenes.

  Halfway through installation, rules inside tools such as iptables or Windows Firewall update themselves if danger appears. When something triggers a change, adjustments happen instantly - no delays. Permissions slide into position just as connections between programs form. Quietly, pieces fit together where you do not see them. Inside quiet connections, strength grows. How things join matters more than loud changes. Little parts of surroundings settle slowly. They hold work steady, keeping it smooth.

- Phase 2: Traffic Capture

  Traffic moves through the network while tools like Wireshark stay silent, watching from key spots. Real data gets captured on the fly - no changes made. These captures happen constantly, though regular operations keep going without a hitch.

  Right after capture, packets become pcap files or flow directly into real-time streams. Feeding active monitoring right away, they keep systems alert. Each shift across the network leaves a mark like this. Important events rarely pass by unlogged. Downstream analysis needs that constant feed to catch repeating shapes.

- Phase 3: Script Development and Traffic Parsing

  Breaking down packets happens using special Python tools made for tracking network movement. With Scapy leading, every script probes saved records, spotting origin and destination - those two IP points. What comes next reveals the path taken - the protocol tag pops into view right after. Then arrives port specifics, slotting neatly behind that last clue. Close to the finish, the measurement of what got delivered sticks around. Every detail fits together, never missing step.

  Inside the script's table, flow specifics pile up - packet counts, data amounts, timed snapshots for each IP or protocol tagged along. Information splits like that let normal rhythms emerge after a while. A drift from what is typical? The setup catches it quick. Unusual surges, timing quirks - they whisper of problems warming up. Seeing whispers early changes everything later on

- Phase 4: Detection Logic Implementation

  Out of nowhere, heavy traffic from one spot sets off warnings. If packets shoot beyond set levels, the system reacts fast. Data flooding in unusually quick? That kind of rush gets flagged without delay. Once limits break, signals fire - guided by how much arrives and when it hits. When numbers jump out of line, alarms sound thanks to preset limits tracking flow patterns. Outliers trigger alerts the moment behavior drifts beyond normal bounds.

  Finding too many attempts often stands out because deeper scans notice quirks in how machines exchange data. Some ports react more when sharp analysis tracks every step they take. When communication acts strangely, hidden settings compare it to past habits. Shifts away from steady rhythms draw eyes once small hints add up.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Finding bad traffic pops up fast, yet everything keeps moving smoothly. When quickness lines up with accuracy, results show without fanfare. The approach stays effective while using only what it needs. Less weight lets responses hit harder right when needed.

- Phase 5: Automated Mitigation

Instantly, when a potential threat appears, the system activates its internal defense mechanism. Next up, using Python's subprocess module, firewall settings are put into place right away - reading lines of code shifts quickly into stopping network flow. On Linux systems, that means working with iptables; on Windows, it talks straight to the Firewall API. Changes apply in real time, no waiting around. Cutting off certain IP addresses is common, though at times it only limits their connection attempts.

First thing, changing rules shut down bad data so it can't move inside. Quick threat blocking means trouble gets stopped right at the door.

Systems keep working, even when hit hard, because safeguards change as needed - slowing things down or cutting ties before chaos spreads. Response speed matches attack strength; bigger waves bring quicker shifts. Changes roll out right away, well ahead of any fallout. Operations continue smoothly since rules reshape themselves in real time. As pressure builds, limits tighten - all by design, none left to chance.

- Phase 6: Logging and Reporting

Fresh records appear the moment something triggers, responses tagged right after. Everything flows into storage exactly as it happens - empty spots? Never.

A moment in time shows up in each record, beside flagged IPs and rules that fired, followed by what happened when defenses responded. Entries shaped this way don't just help look back - they let administrators tweak warnings a little at a time, gradually sharpening how the setup performs with ongoing operation.

- Phase 7: Testing and Validation

Last stands happen during trials - when operations glide without effort, yet also when strained under pressure. Out in the open, systems face trials with tools that act like actual online behavior. Not limited to standard scans, pressure comes from simulated break-in attempts measuring reaction pace. Normal operations moving through reveal if settings wrongly catch safe moves. If waves of harmful signals arrive, how quickly defenses activate tells the story. Checking both angles exposes if protection keeps step with precision.
Precision in identifying problems kicks off the evaluation, followed by response times shaped by system requirements on hardware resources. Real-world performance reveals whether daily operations are manageable beyond lab conditions.

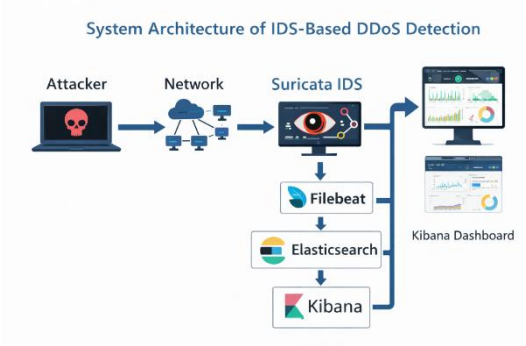


Figure 1. System Architecture



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

A glance at Figure 1 reveals the full layout of the suggested Intrusion Detection System, built to spot DDoS attacks as they happen. This setup forms an unbroken chain - tracking data from initial network flow collection right through to how alerts show up on display panels.

Into the monitored network segment pours network traffic at step one, carrying normal user data alongside simulated attack attempts. The Monitor VM receives a copy of this flow, since that is where the intrusion detection system runs.

Packets flow in. Suricata watches them right away, every single one. Right there, it checks what they're doing using set rules that spot known threats, also digging into how protocols behave. Odd signs? It notices those fast - patterns like floods of junk traffic meant to overwhelm systems show up clearly. Once something sketchy appears, alerts pop up instantly. Each record holds clear facts: who sent it, where it went, what protocol moved it, when it happened, plus which rule flagged it. Logs build quickly, full of raw details, nothing left out.

Out in the open, these logs move toward the Log Collector - known as Filebeat - a slim tool built just to carry data. Not waiting long, it keeps an eye on what Suricata writes down, every bit of it. Right away, each file gets picked up, handled carefully. From there, details travel safely, step by steady step, into central storage. Nothing slips through. Every piece arrives.

Logs get indexed here - speedy searches happen later because storage works smart. When traffic spikes, or attacks hit, this part keeps up without slowing down. Filtering through alerts feels smooth since connections between events show clearly. Handling tons of network noise? That is where it matters most.

Only after sorting comes Kibana showing results through dashboards built for spotting threats fast. These views light up live warnings alongside shifts in network flow, how protocols stack up, where connections begin and end, plus when attacks unfold across time. With visuals like these, staff grasp what each incident looks like without getting lost - measuring scale becomes clearer almost at a glance.

## VII FUNCTIONAL OVERVIEW

*Proposed System Overview*

Out of sight, the device runs on its own, stopping DDoS threats as they start to build. Through every hour, it watches shifts in data flow, acting quickly before harm spreads. Instead of waiting, it moves first, shielding networks from sudden crashes. Components work in sync - bit by bit - maintaining operation when dangers show up.Traffic Capture Module

The Traffic Capture Module acts as the system's first line of observation. It continuously monitors incoming network traffic using reliable packet-sniffing tools such as Wireshark and Tshark. This module interfaces directly with network adapters to record detailed packet-level information, including source and destination IP addresses, protocols, payload sizes, and timestamps. By capturing this granular data in real time, the module ensures no suspicious activity goes unnoticed, laying a strong foundation for accurate traffic analysis and anomaly detection in later stages.

*Analysis Module*

Pulled in by the system, packets head straight into the Analysis Module. Once settled there, processing kicks off using Python scripts. Inside this phase, raw data undergoes changes bit by bit.

Out of each chunk of data, pieces such as IP labels, port numbers, or how much moves at once are drawn out by programs like Scapy. After a stretch of time, counting steps reveal shapes - sudden surges appear when volume leaps unexpectedly. When actions change quicker than usual, rare waves become clear.

From tangled streams of data, shapes slowly take form - sharp jumps show up once noise turns quiet. A lone jump connects to a second, then keeps going, building pulse after pulse where nothing made sense before. Faint hints of strain rise into view well ahead of breakdowns. Clues arrive without fanfare, still loud enough to notice.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

*Detection Module*

At its core lies the Detection Module, operating through fixed guidelines that shape each of its actions. Though quiet, it follows only what it has been told. Every motion comes from earlier decisions built into its frame. Not guessing, just responding as trained. Built on structure, not instinct. Each step tied directly to a prior signal.

A jump out of nowhere from a single location could spark it, if traffic pushes beyond fixed thresholds. It might also wake up due to odd conversational timing among machines, rather than simply tallying pings per tick.

Signals that matter rise above clutter, once you strip away unnecessary layers. Odd patterns catch attention faster than waiting for clues to add up. This approach makes normal behavior easier to tell apart from risky shifts. Steps stay simple because nothing crowds the view.

A step like that values accuracy but asks little from the machine, so it works well when speed or energy are t tight. Even better, staying sharp without bulky code lets devices react quickly even with limited resources. Still, good structure often brings quick outputs right when needed - more so when gear is basic.

*Mitigation Module*

Instantly spotting odd or risky behavior, the system shifts to block mode. Running tools like iptables on Linux, or drawing from Windows Firewall's inner safeguards, it tweaks rules automatically. Any threat - be it a lone IP address or an entire subnet - finds itself locked out quickly. These updates roll through quietly, hidden from view, stopping entry points before harm moves further.

When bad traffic gets shut down quickly, regular users deal with less hassle. Threats get stopped early because defenses adapt in real time.

Mid crisis, the module carries on - no human required. Because it reacts on its own, operations continue despite ongoing dangers.

*Logging and Administrative Reporting*

Every action within the system gets recorded - alerts, reactions, steps taken - forming a visible path forward. Because tracking risks leads straight into blocking them, gaps stay closed without exception.

Midway through each entry sits a precise moment noted down. Right behind it shows where the signal came from. After that piece fits the specific trigger pulled by the incident. Then without delay the machine acts on a preset move to lower danger.

When problems pop up, a clean record makes sorting through events much easier. Since logs show each move the system takes, those managing it stay fully aware. Little by little, adjustments add up - each change nudging performance just slightly better. Catching risks becomes more precise the longer tweaks go on.System Workflow

The system operates as a continuous pipeline, where each module feeds into the next for smooth, real-time operation:

1. The Traffic Capture Module continuously gathers live network data.
2. The Analysis Module processes and summarizes traffic statistics.
3. The Detection Module evaluates traffic behaviour and flags potential threats.
4. The Mitigation Module automatically applies firewall rules to block malicious IPs.
5. The Logging Module records all actions for administrative review and fine-tuning.

This seamless workflow ensures that every stage—from detection to defence—occurs rapidly and automatically, resulting in minimal downtime and maximum network resilience.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

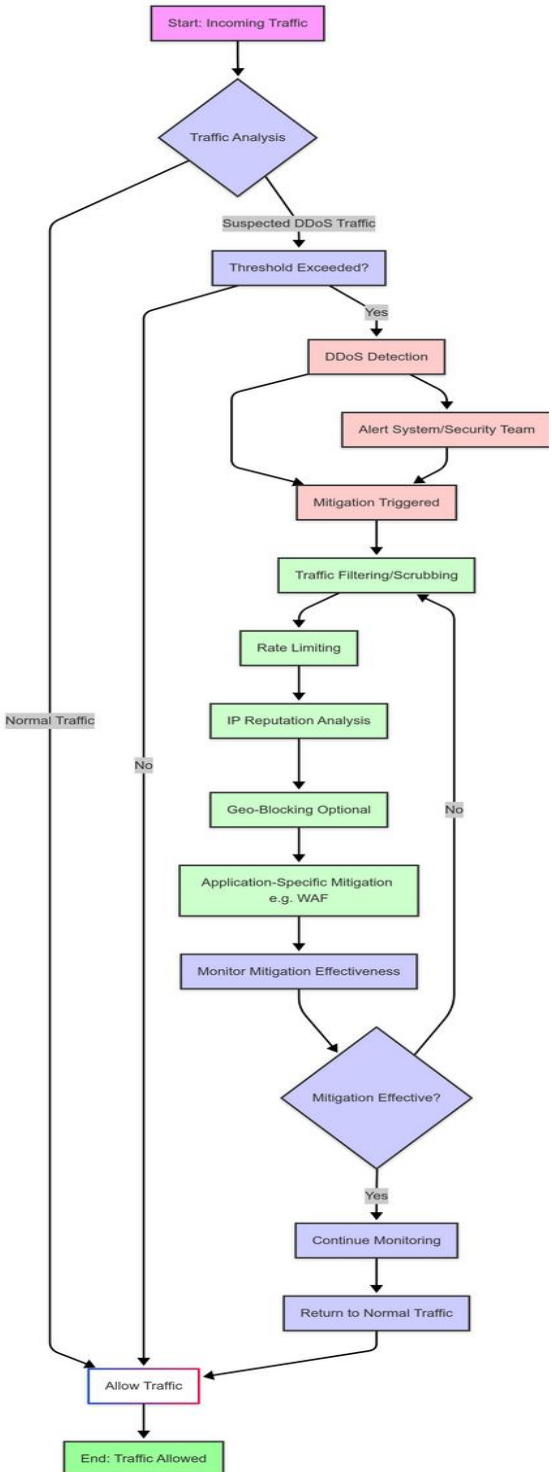


Figure 2 DFD

From figure 2 the incoming network activity comes a mix of real user demands alongside possible hostile signals. Into the Traffic Analysis block goes this stream, there each piece gets checked by its shape, speed, pattern, besides how it talks across protocols.

A green light shows when traffic acts usual - passage happens right away, so real users keep moving without a hitch.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Once odd patterns show up, it checks if set limits - like how fast requests come in or how many connections pile up - are crossed. Only when those numbers go too high does anything change; until then, everything flows as usual.

When the limit gets crossed, alerts pop up right away to tell the security crew or monitoring setup that a DDoS attack has been spotted. Right after spotting it, the system kicks into gear and starts blocking the threat without waiting.

- Moving forward, protection happens through several barriers stacked together
- Traffic filtering and scrubbingto remove malicious packets
- Slowing down too many requests if they come from odd places
- IP reputation analysisto identify known malicious IP addresses
- Geo-blocking (optional)to block traffic from high-risk regions
- Sometimes fixes depend on what the software actually does,
- web application firewall rules

Once fixes take effect, watching how well they work becomes the main task. When those steps actually stop the problem, oversight keeps going without pause. Over time, if trouble fades, operations slowly shift back toward usual patterns.

When mitigation fails, another round begins - this time with tougher measures tucked in. Stability waits. Efforts shift. Adjustments pile on until things settle again.

## VIII SECURITY PERFORMANCE EVALUATION

*Security and Performance Review*

A close look at the suggested DDoS detection and response setup reveals clear advantages, though some aspects still need attention down the road. Despite its strong points, certain elements could benefit from refinement over time.

*1. Detection Accuracy and Reliability*

Most of the time, wrong signals stay rare because the setup watches data flow without pause. Detection works well since rules inside the core spot flood-style and handshake-breaking attacks alike. Traffic shapes get studied nonstop, which helps catch harmful actions early. What stands out is how steady the method stays when picking odd packets from normal streams.
Finding the right balance matters most when traffic spikes happen on live systems - spotting true threats without mistaking normal surges keeps services running smoothly while protecting access for real people who depend on them.

*2. Live Reaction and Damage Control*

What stands out about the system is how fast it reacts. As soon as a threat shows up, actions kick in without delay. Firewall settings shift almost instantly, shutting down risky connections. In moments, harmful IPs get cut off - no waiting. Speed here isn’t just useful - it’s built into every step. Responses happen while threats are still forming, not after.

Using tools like iptables on Linux or the built-in firewall in Windows, the setup quickly shuts down threats before they can drag on and disrupt operations. When danger shows up, it reacts fast without waiting, much like high-end anti-DDoS solutions do by default. Service stays online, interruptions stay short, simply because delays are cut out from the start. Protection kicks in the moment risk appears - no lag, no gaps, just steady uptime maintained through immediate automated steps.

*3. Efficient Resources and Room to Grow*

Light on its feet, the setup runs smoothly thanks to freely available tools such as Tshark and code written in Python. With little strain on resources, it still manages strong protection, skipping expensive gear altogether.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

When tested against fake attacks, the system keeps running smoothly despite high data flow. Though stress increases, operation stays consistent across different setups. Whether used by a tiny team or a large corporation, it adjusts without slowing down. Its ability to grow fits real-world needs while keeping expenses low. Widespread

*4. Complete Logs and Evidence Handling*

What stands out next is how thoroughly the system logs each activity. Because every time a threat gets spotted, blocked, or sets off a rule, it leaves behind a clear trace. That trail helps admins understand exactly what happened after an incident occurs.

A trail of activity hides inside these records, useful later when digging into breaches, meeting audit rules, or fine-tuning protection methods. Clear visibility like this fits how top teams handle security work while making systems answer for their actions.

*5. Adjusts to Different Kinds of Attacks*

Even if it runs mostly on fixed rules, the setup stays loose enough to shift when needed. When fresh threats show up, spotting them gets adjusted without hassle.

Still, outside threat data flows into the system, boosting how it spots complex DDoS strikes. These attacks pile flood-style traffic on top of broken protocols plus sneaky app-level tricks. The setup adapts, pulling in signals from beyond its core. Detection sharpens when layers of assault come together - fast surges, rule-breaking handshakes, hidden payload hits. Outside smarts feed in, quietly raising the guard.

When attack methods change, the system adjusts just the same - its flexibility keeps defenses strong over time. Shifts in threats don’t weaken it; built-in responsiveness holds the line. As tactics shift, so does its approach, quietly matching each new challenge. Longevity comes not from rigidity but from steady, quiet adaptation behind the scenes.

## IX. SYSTEM BENCHMARKING

A thorough evaluation of the proposed DDoS detection and mitigation system shows that it delivers strong security protection while remaining lightweight and practical. The results highlight several key strengths that support effective real-world deployment.

*1. Accurate Reliable Attack Detection*

Now and again, sharp spikes in data flow become obvious only after staring hard at real-time traffic lines. When packets reroute without warning, strange bursts catch the eye - false triggers vanish just as quick. At first glance, regular usage seems fine until pace and size disrupt old habits. Rather than assuming, the system flags issues based on how signals move between links second by second.

Even during sudden surges in activity, genuine visitors move freely when false alerts stay low. The flow never breaks since ordinary actions aren’t confused with danger. Real people pass through quietly, simply because smart checks know the difference.

*2. Fast Response with Auto Processing*

Speed jumps out first thing you notice. Right after spotting sketchy traffic, defenses activate - rules land inside iptables or Windows Firewall instantly. Nasty IP addresses get shut down automatically, zero lag, nothing left open. The second threats appear, response follows like shadow.

Faster responses slow the damage, shrinking both time and impact of assaults. Since machines take charge right away, delays shrink along with human errors during fixes. Networks stay online through peak attack pressure because help arrives before collapse can start.

*3. Lightweight and Scalable Performance*

A machine running everyday tasks handles it just fine when built with tools like Tshark and Python. Even under heavy network load, performance holds up - processor use stays low, memory barely shifts.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Here, growing bigger fits smoothly - be it a small workspace or vast company setup - skipping expensive gear or subscriptions that pile up fast. It simply runs clean, needing less while doing more, tucked into place without fuss.

*4. Detailed Logging for Security Insight*

Midnight alerts flash across screens - timestamps stack up, one after another. Patterns emerge slowly, then all at once, revealing hidden trails. Blocked locations appear in logs, each entry sharp and unmissable. What unfolded sits recorded, moment by moment. Sightings get logged, links severed, every move pinned down without guesswork.

These logs help administrators:
· Review past incidents
· Improve detection rules
· Meet compliance and audit requirements
Each move understood speeds up solving issues, yet stronger safeguards grow through practice. Step by step clarity cuts confusion later, although steady learning shapes better barriers ahead.

*5. Adaptable Security for New Risks*

Today's checks follow set paths, yet evolving threats keep progress moving because the framework bends instead of breaks.
It can adapt by:
· Adjusting threshold values
· Adding new firewall response rules
· Integrating external threat intelligence sources
Fleet moves like these shut down a wide range of DDoS attacks on the spot, yet quietly set up defenses for threats not even seen yet. A single shift does double duty - stopping now, preparing later.

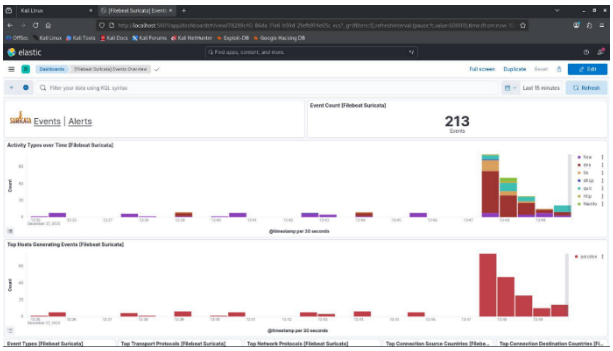


Figure 3. Suricata events and alert analysis

Look at Figure 3- it shows a dashboard built from log data sent by Filebeat, focused on Suricata security alerts. Bars track event activity across time, while another chart highlights which machines trigger the most warnings. This view helps spot patterns in how traffic flows and where alarms pop up across the network.

Over time, the graph tracks how often Suricata spots various kinds of events during set moments. Peak times stand out where odd behaviors pile up - like probes creeping through systems or sudden surges in network noise. When these happen, they might signal early moves toward an attack, whether that's flooding servers or sneaking past defenses. Seeing patterns unfold this way makes it easier to spot trouble before full breakdowns occur. Moments of rising alerts can link back to actions meant to test or overwhelm weak points. Each spike tells a story about what the system faced when under quiet pressure. Instead of waiting for failure, watching shifts gives room to respond while things tilt slowly sideways.

Behind most alerts, a few machines often stand out. When these show up repeatedly, they tend to signal trouble. Spotting them early helps teams respond faster. Some IPs just keep appearing at the top - like clocks ticking wrong. Focus shifts naturally toward those outliers. Patterns emerge when you watch who sends what. Not every host



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

behaves the same way. A small handful drives much of the noise. Watching this flow reveals where stress builds. Action follows attention, especially under pressure.

This setup helps match hacker actions to actual security warnings seen on screen. Thanks to fast log delivery using Filebeat alongside live threat spotting through Suricata, it shows exactly where incidents come from, how they unfold over time, plus common attack styles spotted across systems. As things shift moment by moment, teams gain sharper insight, leading to quicker and more accurate replies when threats appear.

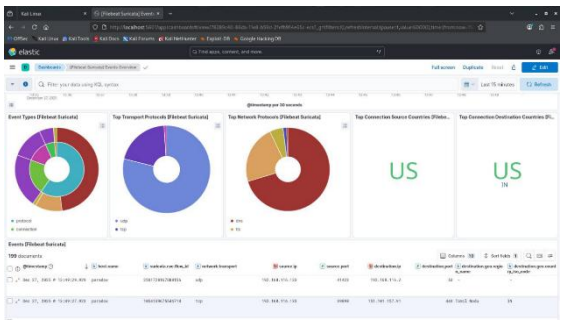


Figure 4 Security events and overview dashboard

Look at Figure 4 - it shows the Security Events Overview Dashboard, giving one clear snapshot of all flagged security issues. Pie charts appear alongside summary boxes, each sorting incidents by kind, how serious they are, or where they came from. Because everything is laid out simply, those managing the system can spot major threats fast. Seeing high-risk alerts compared to minor ones helps balance response efforts. Where the odd behaviors start also becomes visible right away. With this big picture ready to see immediately, choices about what to do next happen more smoothly when attacks like DDoS or break-in attempts show up.

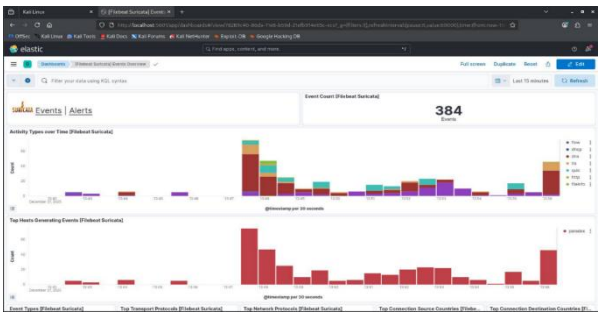


Figure 5. Suricata event alerts by attack type

Look at Figure 5- it shows bars standing for different kinds of network threats caught by Suricata, passed along through Filebeat. Each bar stands apart, sorted not just by what kind of data flow it came from but also by behavior patterns. Flow appears first, then shifts toward services like DHCP handing out addresses. Next comes DNS answering name requests, followed by encrypted talks under TLS. QUIC pops up too, handling fast exchanges. Then HTTP moves in, tracking web interactions. Lastly, File Info holds details about transferred documents.

Spotting the main kinds of network actions becomes easier with this view, along with signs of possible threats seen while watching. A jump in flow-based activity might point to scans or floods, whereas more DNS or HTTP records could mean someone is testing app-level weaknesses. When TLS or QUIC logs rise, it shows efforts to examine hidden data streams, meanwhile File Info entries reveal details about transfers involving files across the system.

Sorting warnings by how they happen and what kind of actions they involve lets this view show clearer patterns in attacks and data flow. Because it shows more than just numbers, those watching security can see what kinds of events are occurring, which makes labeling dangers correctly easier and shapes better responses.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

## X. FUTURE SCOPE

1. Adaptive thresholds shift with changing rules

Later moves could come from code that learns better over time, adjusting boundaries as traffic shifts. Where things go hinges on how real-time data steers those guidelines, left or right.

Slowly, patterns begin to settle into place as the software learns how traffic moves day after day. When changes happen, it shifts along - no need for constant checking by people. With each adjustment, its guesses get sharper. As trust builds, performance holds steady, needing fewer outside checks.

2. Connects to threat intelligence feeds

Midway through its cycle, new data arrives from live threat streams - rules adapt instantly, folding in the latest global breach trends. Alerts spark filter changes right away, no delays built in.

When it sees odd signs first, the system catches new DDoS moves earlier, letting protection shift while traffic flows normally. Since spotting starts fast, changes happen on their own long before major crashes appear.

3. Advanced Anomaly Detection

Finding fresh paths around fixed boundaries, maybe it leans on noticing basic patterns to steer choices. Rather than hard lines, tiny hints in actions can influence the system's reactions.

Watch for strange network patterns without slowing things down. Light methods work well because they skip heavy AI demands. New DDoS attacks show up even when tools are small and fast. Big systems aren’t always better at finding hidden issues. Speed wins when every second makes a difference. Faster results often come from watching how things move, not from piling on extra steps. Simplicity gets there just as quick - sometimes quicker - if you let it.

4. Distributed and Cloud Based Deployment

Out in the open, operations stay steady even under pressure once things spread out. Clouds reshape themselves without hiccups when demand climbs or drops. Pieces traveling through links cut down on big breakdowns. Work split wide means no one spot holds everything at risk. Weight balances easier when power comes from many spots, not just one.

When multiple units share the job of catching and stopping threats, things tend to break less often. Big networks benefit because one machine alone could drown under wide attacks. Spreading the work helps keep everything steady.

5. Encrypted Traffic Analysis Supported

Even with encrypted traffic, strange actions can be caught by tracking packet movement rather than contents. Timing shifts and data sizes hint at activity, despite HTTPS hiding the details. The approach focuses on flow - how often and how much - not decoding messages themselves. Meaning shows up in pauses between bursts, not within the information sent. Arrival moments hold weight equal to the amount delivered. Shape speaks even when stored tight. Under pressure, movement shifts - protocol aside.

Faults slip through less often if oversight works unseen, because private data hides within setup limits built to fit law boundaries.

6. automated incident response linked

Once connected through APIs, the system replies on its own while fitting right into existing setups. Older tools stay aligned because the link removes the need for hands-on tasks. Speed picks up when security layers talk directly thanks to open access points. New features slide into routine operations without causing ripples when interfaces are shared freely. Updates move through every platform when links are active. With interfaces, doing things together works even if the main code stays untouched.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

A signal hits - suddenly firewalls shift by themselves, alerts light up across control panels, each station reacts before you can blink. Breach one point, response explodes everywhere, not step by step but in unison, as if struck by the same spark.

7. Smarter Layouts Brighter Screens

Numbers update quickly on a bright display, revealing real-time movement across the network. The instant anything changes, alerts show up without delay, so staff stay aware. Visuals turn raw figures into simple images, helping track activity clearly. Patterns emerge step by step, replacing uncertainty with sight. Spikes show up fast when you see them next to flat lines. A clean setup lets eyes move freely across shifts. When nothing hides, decisions follow close behind what appears. Growth paths stand out during routine glances through shifting patterns. Small changes add up quietly while attention stays steady.

With visuals that shift as you interact, attack patterns pop out quicker. Motion reveals system actions better than static images ever could. Changes click into place easier when feedback happens instantly.

## XI. CONCLUSION

From tests run, sudden floods of data plus hidden tricks in protocols popped out quickly. Live packets plucked from traffic flows went straight into scans guided by strict rules. Right when strange jumps appeared, their origins marked without delay. Just the ones blowing past thresholds lit up alarms - everything else just sat there quiet. Most of the time, actual people logged in just fine. False alarms on regular activity were nearly nonexistent.

Suddenly, the firewall adjusted on its own, blocking strange IPs before they could get near. Thanks to those changes, actions followed quickly - no pauses, no drawn-out gaps - and everything kept working without a break, despite constant attempts. Under strain, it still performed well, moving quietly through each demand. Even during heavy load, operation remained calm.

Barely any CPU or memory got used during testing, even when lots of data flowed through. Performance held up fine because the Python code stays lean. Without needing powerful hardware, things kept moving fast. Packets were gathered easily by small tools, which helped it work across many network types. When demand spiked, speed never dropped. Finding better ways matters more than having strong equipment. Where supplies run short, it still gets done.

Fresh traces mark each alert, then show steps taken and how tech reacted - paper trails form naturally. Happenings get easier to follow since those notes let crews trace moments step by step. Over weeks, choices grow sharper because patterns emerge, tweaks arrive through live feedback on triggers. Each line brings sharpness, weaves a sequence sturdy enough for tough review.

Even so, much like systems built on rigid rules, it relies on set boundaries that often require adjustments as networks evolve. When shifts happen, detecting sneaky or jumbled attack patterns may grow harder. As conditions shift, smarter threshold controls combined with live updates from external threat sources might boost detection. Over time, weaving in these elements could sharpen response to layered assaults. Though unchanged in scope, adaptation becomes key under pressure.

When stress comes, this system keeps working, blocking real-time DDoS strikes without slowing down. Built basic, yet it outperforms cluttered designs by spotting danger early and moving fluidly. Since it uses power wisely, connections hold strong when chaos begins. Precision doesn’t drop, no matter how wild things get, because hidden logic adapts on its own. Each unseen block boosts reliability a little more, stopping harm before it grows.

## REFERENCES

[1] N. S. Bansal et al., “End-to-End DDoS Detection and Mitigation Using Federated Learning and SDN, ” *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 1450–1463, May 2025.

[2] L. Singh and P. Jain, “Design and Implementation of Real-Time DDoS Mitigation System Using SDN,” *Future Internet*, vol. 17, no. 2, p. 113, Feb. 2025.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

[3] H. Rahman and M. R. Islam, "Zero-Day DDoS Detection Using Transformer-Based Network Models," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 1, pp. 21–30, Jan. 2025.

[4] Julien, V., Baerts, E., Deleu, J., and Open Information Security Foundation (OISF), "Suricata: An Open Source IDS/IPS Engine," 2024.

[5] A. Sharma and N. Singh, "Mitigating Layer 7 DDoS Attacks Using Reinforcement Learning-Based Traffic Shaping," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 13, no. 2, pp. 91–101, 2024.

[6] D. R. Kumar et al., "AI-Based Mitigation Techniques for Real-Time DDoS Detection in Smart Cities," *IEEE Smart Cities*, vol. 7, no. 2, pp. 77–86, Mar. 2024.

[7] S. Mehta and V. Jain, "Performance Analysis of DDoS Attack Detection Models Based on Dataset Feature Optimization," *Information Security Journal: A Global Perspective*, vol. 33, no. 1, pp. 15–27, Jan. 2024.

[8] Wang, K., Zhang, Y., Sun, Q., & Zhang, H., "MDDCC: A Multi-level DDoS Detection and Control Collaboration Scheme in SDN," *IEEE Transactions on Network and Service Management*, Early Access, 2024.

[9] R. Kumar and A. Sharma, "An Efficient Machine Learning-Based Framework for DDoS Attack Detection in SDN Environment," *Journal of Network and Computer Applications*, vol. 213, p. 103553, 2023.

[10] A. Sinha and R. Dubey, "Lightweight ML Model for DDoS Detection on IoT Devices," *Sensors*, vol. 23, no. 2, pp. 765–779, Jan. 2023.

[11] S. R. Jadhav et al., "Enhanced DDoS Attack Detection in Cloud Using Hybrid RNN-GRU Model," in *Proc. ICAC3*, 2023, pp. 55–60.

[12] K. Chen and F. Liu, "Software-Defined Network-Based DDoS Detection Using Flow Entropy and Graph Features," *Journal of Systems Architecture*, vol. 139, p. 102775, Jun. 2023.

[13] M. Patel et al., "Anomaly Detection in Encrypted Traffic for DDoS Mitigation Using Autoencoders," *Computer Networks*, vol. 227, p. 109792, Mar. 2023.

[14] T. Yadav and M. Mishra, "Real-Time Detection of DDoS Attacks Using Ensemble Learning," *International Journal of Information Security Science*, vol. 12, no. 1, pp. 11–20, Jan. 2023.

[15] M. Alazab, K. R. Choo, Z. Xu, and S. Zeadally, "A Survey of DDoS Attack Detection and Mitigation Using AI Techniques," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1254–1283, 2023.

[16] Kumar, A., & Pandey, M., "Hybrid Deep Learning Framework for DDoS Detection Using ResNet-50 and Optimized AlexNet," *Procedia Computer Science*, vol. 219, pp. 14–23, 2023.

[17] Wang, J., Tang, M., Liu, X., & Wang, K., "CC-Guard: A Deep Learning-Driven SDN Architecture for DDoS Attack Detection and Mitigation," *IEEE Access*, vol. 11, pp. 16422–16434, 2023.

[18] Y. Zhang, X. Liu, and W. Chen, "Real-Time Detection of Distributed Denial of Service Attacks Using Hybrid Deep Learning Model," *IEEE Access*, vol. 10, pp. 37654–37667, 2022.

[19] R. Vinayakumar, S. Soman, and P. Poornachandran, "Real-Time DDoS Detection Using Ensemble Learning," *Future Generation Computer Systems*, vol. 125, pp. 45–57, 2022.

[20] Mehta, R., & Tiwari, A., "Comparative Study of Zeek and Suricata IDS Tools for Network Traffic Analysis in Cybersecurity," *Proceedings of ICCIDS*, 2022, pp. 105–112.

[21] Kaur, R., & Singh, M., "An Overview of Open-Source Tools for Network Traffic Monitoring and Intrusion Detection Systems," *International Journal of Computer Applications*, vol. 184, no. 25, pp. 1–6, 2022.

[22] Golduzian, A., Azgomi, M. A., & Bitaraf, F., "DDoS Attack Detection Using Machine Learning Techniques with CICDDoS2019 Dataset," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, pp. 5



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

[23] S. M. Fatani and S. M. Altowaijri, "A Review of DDoS Attack Detection Using Machine Learning and Deep Learning Techniques," *Sensors*, vol. 21, no. 11, p. 3704, 2021.

[24] Chatterjee, M., & Saha, B., "Real-Time Detection of DDoS Attacks Using Lightweight Flow Statistics and Rule-Based Filtering," *Computer Networks*, vol. 194, p. 108164, 2021.

[25] Noor, A., & Khan, M. S., "Anomaly-Based Detection of Distributed Denial of Service Attacks Using Flow Statistics," *Journal of Network and Computer Applications*, vol. 173, p. 102869, 2021.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)