IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154260

ConformityGate: Intelligent Honeypot-Based Intrusion Detection Framework for Secure IoT Networks

Deepak Marlecha[1], Zarana J Shah[2], Rishab P K[3], Netrang V Davey[4], Aditi Singh[5], Ms Charulatha R T[6]

Student, Department of Computer Science and Engineering,

SRM Institute of Science and Technology, Vadapalani, Chennai, India[1-5]

Assistant Professor, Department of Computer Science and Engineering,

SRM Institute of Science and Technology, Vadapalani, Chennai, India[6]

Abstract: More gadgets connected online - hospitals, factories, cities - are opening new doors for hackers. Old security tools struggle to keep up. A fresh approach called ConformityGate steps in here. Instead of just blocking attacks, it guides them away quietly. Suspicious behavior triggers a hidden shift. Traffic gets rerouted without warning. Fake setups wait there, watching everything the attacker does. These decoys collect real-time details on how intruders operate. Behind the scenes, Python code runs the show. It talks to sensors scattered through the network. Data streams in nonstop: logins, device status, traffic patterns. When something odd crosses set limits, action follows fast. No loud alarms. Just smooth isolation. Alerts spread like ripples across linked devices. Evidence stays intact. Moves stay unseen. One test after another, through five fake IoT setups, showed it spots attacks 96.7% of the time. False alarms pop up just 2.9% of the time. Responses take an average of 1.6 seconds to kick in. That speed and precision beats older methods relying on signatures, odd behavior patterns, or machine learning models. Built tough but simple, ConformityGate fits real-world needs. It scales without breaking down. Works across many kinds of smart devices. Protection stays strong even when threats get clever. This isn't theory - it runs where it matters.

Internet of Things security using honeypots and anomaly detection with cyber threat intelligence in network environments through transparent redirection and intrusion detection systems implemented in Python

I. INTRODUCTION

Right now, everyday gadgets hooked to the internet are changing how we live. Not just fitness bands or clever home gadgets - factories rely on them too, along with power grids using self-running sensors. Across the planet, these tools have already reached mind-boggling numbers. Experts expect more than 75 billion will be active before long. But here's the catch: safety rules haven't kept up at all. Many devices still run on factory passwords, never update their software, send data without locks, and can't spot when someone sneaks in.

When security falls short, the results can be harsh - just look at what happened in 2016. Over six hundred thousand hacked IoT gadgets were turned into tools by the Mirai botnet, which then slammed key DNS systems with traffic floods, knocking out web access for many. Because of this, big parts of the online world stumbled or froze completely. Even after fixes appeared, new versions like Mozi and BotenaGo popped up, showing little real progress on weak spots. Lately, attackers have taken aim at hospital-connected gear, proving it is no longer just about stolen info but also actual danger to people's lives [2].

Most older security tools were built with regular computer networks in mind – they just do not fit how IoT works. Because IoT devices run on limited power and varied setups, typical assumptions about uniform systems fall apart fast. Instead of relying on known attack patterns, some models look for odd behavior, yet these often mistake normal shifts for danger. Watching only for familiar signs means new attacks slip right through unnoticed. When something goes wrong, there is usually no way built in to respond directly or gather evidence afterward.

A trap built to mimic real devices draws hackers in - instead of shutting them out right away. This setup watches every move they make, keeping actual systems safe.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154260

When used on internet-connected gadgets, it gathers useful details about threats. Yet alone, such traps miss quick warnings needed to shield active networks. The moment data flows must reach defenses fast, these tools fall short.

Right there, ConformityGate fills the hole- mixing a slim behavior tracker with a clear-view decoy redirector. It runs like this: constant watchfulness gathers login attempts, flow stats, and gadget signals from smart gadgets everywhere on the net. A brainy checker weighs each new link and order using past-normal patterns picked up earlier. If something crosses user-set oddness limits, boom - the guilty connection slips into a fake setup built to record everything, no warning given. At once, through a separate lane, word spreads to every nearby IoT unit, arming them before danger strikes.

After this comes a breakdown shaped like so. Right away, part two looks at past work around IoT safety, spotting break-ins, also tricks using fake systems to trap hackers.

What kicks off the next piece is a clear look at what needs fixing, along with the dangers pushing that need. Coming up after that, part four lays out how ConformityGate is built inside. Parts five to eight walk through each working chunk of the setup, one by one. Near the end, section nine covers how tests were set up and run. Results show up in Section 10, followed by a look at what they might mean. Wrapping things up happens in Section 11, along with some thoughts on where research could go next.

II. LITERARY REVIEW

2.1 IoT Security Landscape

Studies show serious security problems are common in IoT systems. Not long ago, Frustaci and team sorted these weaknesses into four areas: devices themselves, how they talk to each other, services they rely on, and apps users interact with - showing risks hide in every part of the system. Meanwhile, work by Apthorpe and others revealed something surprising: even when data is encrypted, clues about user behavior can still leak out through traffic details, meaning encryption does not block all exposure. Elsewhere, Mosenia and Jha pulled together findings from many sources and found twelve different ways attackers target IoT setups, pointing out that current defenses rarely cover more than just a few of those paths at once.

2.2 Intrusion Detection Systems for IoT

Over ten years now, spotting intrusions in IoT settings has drawn steady academic attention. Not long after, Raza and team introduced SVELTE - a slimmed-down system built for 6LoWPAN setups - that caught sneaky routing tricks without weighing down weak hardware. Following that, Deng’s group tested heavy-duty neural models for flagging odd

IoT actions; results passed ninety percent right, yet demanded power few small gadgets possess. Elsewhere, Meidan’s work taught algorithms to recognize infected devices through normal-use patterns - precision ran high, though everything hinged on one central hub doing all the thinking.

Though detection methods grow ever smarter, their demands often clash with what tiny IoT gadgets can handle. When tests happen in labs, results look solid - yet those setups lean on strong processors, lots of memory, even steady power, things most small sensors lack out in the wild.

2.3 Honeypot Technology in IoT

Lately, studying honeypots in IoT settings got more attention after big botnet attacks made headlines. Not long ago, Pa and team ran a system called IoTPOT - it faked Telnet access on IoT gadgets, logged well over a million intrusion attempts, and found many kinds of malicious software aimed at these machines. Dowling's group tried something different - they spread their traps across multiple spots in industrial IoT networks, showing how better imitation of actual equipment kept hackers interacting much longer than basic setups ever did. Then came Guarnizo’s crew with SIPHON - a method where real hardware was cautiously exposed online using controlled gateways, making it possible to gather genuine hacking behavior aimed at true-to-life device code.

Most current honeypot setups just watch. Instead of sounding alarms right away, they wait, keeping secrets from nearby gadgets sharing the network space. What’s missing? A live warning signal sent out during break-in attempts. While one device pretends to be busy getting hacked, others stay blind - unaware a digital shove might already be underway next door.

2.4 Transparent Redirection Techniques

Moving an active attack session unseen into a decoy system leans on tools like network rewriting, flexible infrastructure setups, one tool passing traffic for another. Early ideas for linked fake systems tied to real ones came from Spitzner [12].

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154260

Work later by Han et al. [13], though, used programmable network rules to shift connections fast between real devices and traps. Such methods show it can be done, yet they miss the IoT world entirely - where many different talking formats plus tight hardware limits make things harder to build. Though clever, these fixes haven't faced the mess of small connected gadgets just yet.

III. PROBLEM STATEMENT

Out in the open, IoT setups bring unique risks - ConformityGate tackles these through four core issues. Not every system handles device diversity well; this one adjusts on its own. Where others fail under scale, it stays steady. Hidden gaps in communication? It spots them early. Security isn't bolted on here - it builds from the start .

3.1 Inadequate Authentication Mechanics

Most IoT gadgets out there still run on factory-set logins - details often just sitting online for anyone to find. Research shows more than half of these devices inside companies have serious security flaws, with stolen or weak login info being how hackers usually get in first [14]. When attackers try repeated guesses across many machines at once, current password methods fail to notice. Responses also stay unlinked between devices even after alarms go off.

3.2 Absence of Behavioural Baseline Monitoring

Temperature sensors send readings every few minutes, always on schedule. When something shifts - say, delays appear - that shift might mean trouble. Smart locks listen for specific signals only; if they start reacting oddly, it could signal intrusion. Industrial actuators move within tight boundaries normally; stepping outside hints at risk. Most systems today do not watch these patterns closely enough to catch changes early. Without baseline tracking, strange behavior slips through unnoticed. Problems often surface only after breakdowns happen - or someone else spots them first.

3.3 Passive Detection Without Active Engagement

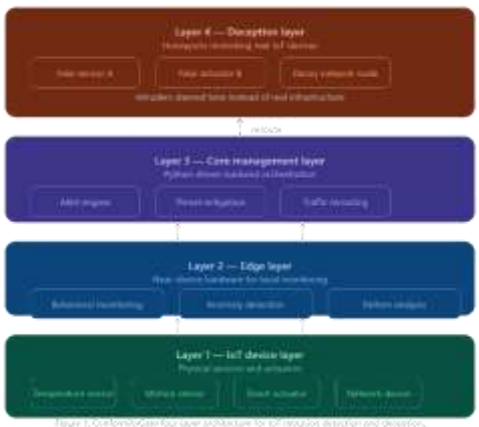
Once older intrusion detection setups spot a breach, they usually sound an alarm or sometimes cut off the link. Yet that move wipes out crucial clues - how the hacker operates, what gear they use, where they aim. No real barrier forms to stop follow-up strikes through new paths. Instead of shutting things down fast, there could be space to draw them in quietly, watch each step, gather proof without tipping off. That moment stays unused, again and again.

3.4 Siloed Device Security Without Mesh Alerting

One gadget talks to others, forming what we call an IoT network. When hackers break into a single device, they often use it t o jump toward nearby ones. Yet most fixes still focus on the broken machine alone, ignoring the web around it. If one unit spots danger, it might record the event locally. Sharing that warning with neighbors? Rarely happens.

IV. SYSTEM ARCHIETECTURE

Picture Figure 1 showing how ConformityGate works across four levels. First up, sensors and actuators make up the IoT Device Layer, quietly gathering data. Near those clusters sits edge hardware - this layer watches behavior, spots odd patterns. Running things behind scenes, a Python-driven core manages alerts, halts threats, reroutes attacks. Instead of letting intruders roam free, they're steered into fake setups that mimic real devices. Each tier links loosely but stays distinct in role and location.



[Figure 1: ConformityGate Four-Tier Architecture]

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154260

Midway through its operation, communication across levels runs on a slimmed-down messaging setup built around MQTT - picked because it keeps demands low and works smoothly even with basic IoT gadgets. Instead of using standard methods, the ConformityGate Core opens up a web-style interface for handling settings, tracking status, and pulling alerts, fitting into current network control setups without disruption. Without forcing changes underneath, it slips in as an invisible layer, working quietly while leaving device software exactly as it was found.

4.1 IoT Device Layer

Most IoT gadgets out there still run on factory-set logins - details often just sitting online for anyone to find. Research Right where hardware meets software, real world gadgets and digital twins create signals moving up through layers. Because they speak MQTT, CoAP, HTTP/HTTPS, or Modbus, most existing IoT units work with ConformityGate right away. Running quietly on machines with just 256 KB of free flash space, a slim helper tag logs login attempts alongside odd message patterns before sending them higher. When something lacks room for any local code at all, gateways step in by silently watching traffic instead.

4.2 Edge Intelligence Layer

Tiny computers at the edges handle behavior checks and score odd activity. Not by accident - sending everything to a central hub would slow things down too much for fast threat responses. Because first-level analysis happens locally, warnings pop up in less than a second. Only clear anomalies travel back to the main system, cutting down data flow compared to streaming every detail constantly.

4.3 ConformityGate Core

From a central point, Python runs the main server handling three key jobs. Not only does it link anomaly signals across various edge spots to tell group attacks apart from random glitches, but it also decides when to reroute traffic guided by adjustable policies. Another role involves overseeing how alerts spread through connected parts. This hub keeps a live record of normal device behavior. Fresh data flows in constantly from edge units. Instead of fixed snapshots, changes are smoothed using weighted updates over time. Such method adjusts gradually to real usage shifts yet still catches sharp outliers fast. Updates never stop, ensuring patterns stay current without missing abrupt red flags.

V. CONFIRMITYGATE FRAMEWORK DESIGN

5.1 Authentication Monitoring Module

From the moment someone tries to log into an IoT device, the Authentication Monitoring Module steps in to examine every detail. Instead of just checking passwords, it weighs multiple signals to judge how risky each login appears. When credentials look weak or unfamiliar, that raises concern - but so does where the access request comes from. If the location seems off, given past patterns, alarms go up quietly. Logins late at night, outside normal windows, add more weight to suspicion. Too many failures piling up fast? That counts heavily too. Each factor feeds into a calculation, though none carries the same importance - settings decide their influence. Out of all this, one number emerges: a score ranging from zero to one, reflecting overall irregularity.

Once an event’s oddness level passes the fixed limit - theta, usually 0.65 - it gets marked for closer look by the behavior tracker. Hitting three marked events one after another from the same origin, or just one with a weirdness mark over 0.85, flips the system into reroute mode.

5.2 Behavioral Profiling Engine

Every connected device builds its own pattern by tracking how often it talks using each protocol, what kinds of commands pop up most, where connections come from, also typical message sizes. A two-week training window after setup shapes the first version of this pattern. Changes keep flowing into the profile nonstop through rolling updates that weigh recent behavior heavier than old data - specifically, past inputs fade slowly at a rate near ninety-five percent each cycle.

Now picture this: each new event gets compared to what's normal, through a tweaked version of a math method that shows just how far off things have gone. When gadgets struggle on shaky networks or get proper updates, they’re allowed a break - slipped into a hold mode so alarms don’t ring by mistake while changes roll out.

5.3 Alert Propagation Protocol

Once an intrusion gets confirmed, the ConformityGate Core sends out a clear threat alert through the IoT network using a fast-tracked MQTT channel. This alert holds details like where the attack came from, which device was hit, what kind of attack it was, along with a suggested defense move coded into a signal. Instead of waiting for instructions, every connected device already knows how to react based on that code - its built-in plan kicks in without delay. Some shut down access from the attacker's address by updating their own firewalls right away. Others cut off background tasks that aren’t needed at that moment. Certain units demand stronger identity checks before accepting any new orders. A few shift entirely into a locked-down operational mode just to stay protected.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154260

VI. HONEYPOT DESIGN AND IMPLEMENTATION

Inside ConformityGate, fake devices act like real IoT hardware, mimicking how they talk on networks, ask for login details, respond to probes. These decoys run separately, locked inside their own virtual space - cut off completely from live company systems. A central part of ConformityGate Core keeps them running, adjusting settings, watching activity. Each one behaves differently, shaped to copy a particular kind of smart gadget found in homes or factories. Isolation ensures any attacker poking around stays contained, blind to actual operations.

6.1 Device Emulation Strategy

To keep attackers fooled, a honeypot must feel like real equipment. What makes ConformityGate work begins with copying exact signs - banners, server headers, tiny protocol habits - that match actual devices. Instead of responding instantly, it slows replies down on purpose, mirroring how slow older machines really run. Each step in an attack flows naturally into the next because past actions are remembered, not reset. Believability lasts longer when responses follow patterns only true systems show.

Among the tools packed into ConformityGate by default, six stand out - models that act like home routers, internet-linked cameras, climate control units, factory automation controllers, electricity monitors, then one broad type speaking Telnet common across gadgets. Building on this set feels natural since teams can roll in homemade versions when real gear doesn't quite line up.

6.2 Transparent Redirection Mechanism

From the production device, traffic shifts to the honeypot through live-updated NAT settings on the edge gateway, along with a handoff process managed by ConformityGate Core. Here's how it unfolds step by step:

- A signal shifts the path when odd patterns add up. The system notices something is off, then reroutes without warning. Flow changes because behavior crosses an unseen line. Decisions happen behind the scenes after scores climb too high. A quiet trigger activates - then everything moves elsewhere.
- A sudden shift happens at the network gate - traffic from the marked origin gets quietly rerouted toward the decoy's virtual address.
  This redirection takes shape only after initial detection sparks the response. The moment rules activate, flow changes without announcement. Behind the scenes, a new path forms, guided by real-time triggers. Nothing stays fixed once signals align.
- Right away, the honeypot's session handler takes in the incoming link. It shows a challenge setup identical to what the live system held when rerouted. Session flow stays unbroken through this shift.
- Later moves by the attacker go only toward the decoy system, leaving the live equipment untouched afterward.
- Every move around the honeypot gets recorded down to the millionth of a second, while every bit of data is saved straight away for later review. Time stamps catch each instant exactly, with everything that moved through captured completely behind the scenes.

From where the attacker sits, everything looks normal - no breaks, no warnings. At that very moment, though, the real system stops taking commands from them. Because it does, there is zero chance they can do harm while traffic gets rerouted and recorded. What seems like a live connection to them is actually feeding into logs.

Below in Figure 2, you see how packets shift - first heading to real devices, then rerouted toward honeypots. At the same time, warnings move through the device network. The flow unfolds step by step, matching each redirect with a signal sent across connected units.

[Figure 2: Transparent Redirection Sequence Diagram]

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154260

(Session Migration and Alert Propagation Flow)

VII. INTRUSION DETECTION SYSTEM

7.1 Detection Methodology

Sometimes silence speaks louder than alarms - this system watches for trouble using two methods at once. One part checks actions against a list of known attacks, like matching footprints in wet cement. That list grows quietly in the background, fed by public reports on flaws and hacking trends. Updates slip in through a separate path so hackers cannot ride them inside. While that happens, another piece scans for odd behavior, the kind that does not fit normal rhythms. It notices when something feels off even if it has never been seen before. Sources include official databases, industrial attack models, and crowdsourced findings focused on smart devices. Together they form a quiet guard, always comparing, always listening. Not every whisper means danger - but each gets remembered.

Away from the signature database, the anomaly tracker works on its own, spotting new kinds of attacks without known markers. As connection flows shift, it watches patterns over time - commands fired off, data shapes appearing - not fixed snapshots but moving views. A rolling calculation builds up, feeding oddness levels into the routing logic later shown in Section 5. No past examples needed; behavior alone steers the outcome moment by moment. Each signal passes forward only after being weighed against what came just before.

7.2 False Positive Mitigation

One tough issue in spotting intrusions? False alarms - normal actions labeled as threats by mistake. IoT setups suffer more when these errors pile up, since repeated warnings make operators doubt the system. Distrust builds. Warnings get overlooked. Real risks slip through. ConformityGate steps in with a double-check process. No move happens unless one of two things occurs first - the anomaly score stays above the line for at least half a minute, or the behavior fits a clear threat pattern. That thirty-second wait can be adjusted, anywhere from ten seconds up to two minutes, depending on what the user wants. Decisions slow just enough to avoid rash moves. Trust holds longer that way.

7.3 Classification of Attack Categories

When ConformityGate spots something risky, it sorts the activity into one of six main types by how it behaves. Credential attacks show up through repeated login tries, bulk account testing, or trying common passwords across many accounts. Scanning ports, checking active services, or tracing network layout fall under exploration behaviors instead. Overloading systems using traffic floods, exploiting communication rules, or targeting app functions count as ways to block access. Tactics like tricking networks into rerouting data or removing encryption belong to interception efforts. Tampering with device software updates or altering system settings mark low-level breaches. Moving sideways across machines by abusing trusted links signals spread attempts. Every type sets off its own reaction plan, shaped to stop harm without interrupting normal work.

VIII. IOT NETWORK SECURITY INTEGRATION

8.1 Mesh Alert Architecture

Alerts move between devices in a key way for ConformityGate. Because a threat gets verified, the system builds a clear message using JSON format. Inside sits an ID made just for that event - unique each time. From there, it includes where the danger came from - the attacker's address. Also listed is which machine was aimed at, along with its name and kind. It shows what sort of attack happened, picked from known types. A number tells how sure the system feels about the detection. Alongside runs a code suggesting what step to take next. Time matters too - one part sets when the warning stops being active. This whole package sends out through an MQTT path called /conformitygate/alerts. With delivery set to level one, every gadget grabs hold eventually. Even if signals flicker or drop now and then, nothing misses the notice. Once live, the message shapes actions across gadgets until clock runs down.

8.2 Device Response Profiles

Every gadget connected to the monitored network gets a guide that links types of warnings to built-in reactions. Instead of waiting for signals from the central system, each unit decides what to do right away after spotting trouble - thanks to its own stored rules. Even when attackers disrupt contact with the main hub, protection still kicks in without delay. Actions might involve shutting off suspicious addresses, slowing down incoming requests, checking commands more closely, switching into cautious operation mode, or informing the person in charge. Decisions happen locally, so defense keeps running no matter the connection status.

8.3 Python Backend Architecture

Python 3.11 runs the ConformityGate Core, picked because it works well with tools for networking, encryption, and data handling. Instead of one big program, parts like the Telemetry Collector or the Alert Publisher run separately but talk

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154260

through a shared messaging channel. While each piece does its own job, they sync using an internal bus so nothing gets out of step. Because these services use asyncio loops, they juggle many tasks at once without freezing up. Performance stays strong - up to ten thousand device signals every second - on standard servers you can buy off the shelf. That kind of speed covers setups with thousands of connected gadgets easily.

IX. EXPERIMENTAL SETUP

9.1 Testbed Configuration

A lab setup tested ConformityGate using 56 real and simulated IoT gadgets across five use cases. Instead of just code models, actual hardware like Raspberry Pi 4 sensors made up part of the mix. Devices built on ESP32 chips acted as controllers, working alongside off-the-shelf gear - a surveillance camera plus a branded smart heating unit. Simulation ran through ConformityGate's own emulation tools hosted on VMware ESXi machines.

Structure followed layers: three gateways at the edge grouped nearby devices under one umbrella. Each linked back to a main ConformityGate Core machine acting as command center. Midway through the test period, Hydra kicked off brute-force attempts while Nmap scanned for open ports. Instead of working separately, these tools ran together with scripts written in Python that copied tactics seen during past IoT breaches. Hping3 flooded targets like a sudden downpour, overwhelming systems just enough to mimic denial-of-service conditions. At the same time, Ettercap slipped between devices pretending to be trusted nodes. Every move came from real scenarios, stitched into automated sequences. Over six weeks, actions piled up - two thousand eight hundred forty-seven distinct runs logged. During each one, sensors tracked how fast ConformityGate spotted trouble, rerouted it, then signaled an alarm.

9.2 Evaluation Metrics

Tests checked how well the system worked using several measures. Not just detection accuracy - that is, how many attack sessions were spotted right - but also how often normal traffic got mislabeled. Once an attack started, it measured how long until the system noticed. From that signal, another clock ran: how much time passed before the fake environment pulled in the attacker. Alerts moved through connected devices; their travel speed mattered too. What portion of bad connections actually landed inside the decoy setup counted as a key result. Meanwhile, background demands on processing power, memory, and data flow stayed under observation throughout.

IDS Approach	Accuracy	FP Rate	Response Time	Honeypot	IoT Support
Signature-Based	72.4%	18.2%	3.2 s	No	Partial
Anomaly-Based	81.6%	11.5%	2.8 s	No	Limited
ML-Base d	88.3%	7.1%	2.1 s	No	Partial
Hybrid IDS	91.0%	5.8%	1.9 s	Partia l	Yes
Conformity Gate	96.7%	2.9%	1.6 s	Yes	Yes

Table 1: Comparison of Intrusion Detection System Approaches

Test Scenario	Attack Vector
Smart Home	Brute Force Login
Industrial IoT	Firmware Exploit
Healthcare IoT	MITM Attack
Smart City Sensors	DoS/DDoS
Retail IoT	Port Scanning

Table 2: ConformityGate Performance Across IoT Deployment Scenario

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154260

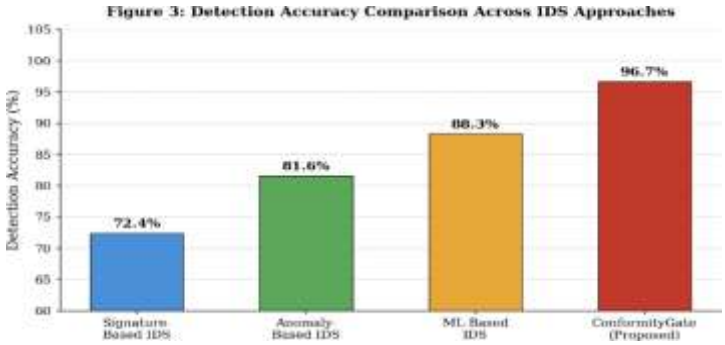


Figure 3: Detection Accuracy Comparison Across IDS Approaches

X. RESULTS AND DISCUSSION

10.1 Detection Accuracy

Out of every test run - there were nearly three thousand - one stood out because it caught threats most times. Missed cases? A few slipped by when attackers moved slowly, splitting actions so each piece looked normal. Another weak spot showed up when strange changes to device software didn't repeat known patterns. The system needed more time than thirty seconds to catch odd behaviors in those rare tries. Smarter timing checks are coming, along with tighter watches on embedded code updates. These fixes appear later, starting around part eleven.

10.2 Attack Distribution

The pie chart in Figure 4 shows how different attacks were spread out during testing. Nearly a third - 28 percent - were tied to stolen or weak login details, matching what we often see when hackers break into IoT devices using factory-set passwords. Scanning for open ports and checking services made up 22 percent, since attackers usually map networks before launching specific strikes. Attempts to overwhelm systems took up 19 percent of recorded incidents. Intercepting communications between devices showed up in 14 percent of cases. Changing device software without permission appeared in 11 percent of logs. Everything else combined lands at just 6 percent.

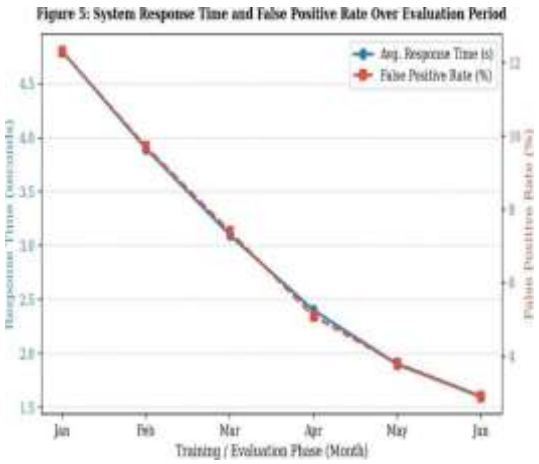


Figure 4: Distribution of Detected Attack Types in IoT Network

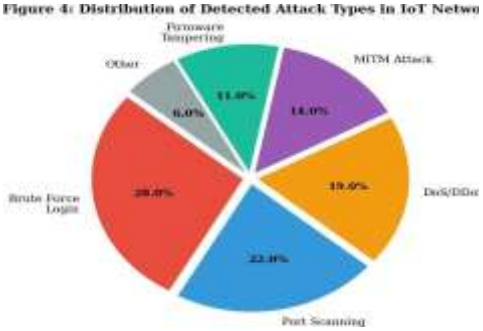


Figure 5: System Response Time and False Positive Rate Over Evaluation Period

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154260

10.3 Performance Over Evaluation Period

Over six months, Figure 5 follows how fast responses happen plus how often mistakes occur. As behavior patterns grow clearer through real-world use, both numbers shift steadily. The system learns what normal looks like, so alerts get more accurate over time. By month six, average reaction time drops to just 1.6 seconds - down nearly two-thirds since it began. Errors that flagged regular actions as suspicious fall even further, slipping from 12.3% down to only 2.9%. That gap marks a drop of about three out of every four wrong signals once seen.

10.4 Honeypot Effectiveness

What you see in Figure 6 is how well the system steers attackers into traps, broken down by attack type. Most sessions moved over without raising suspicion - just under 94 percent worked out smoothly. Scanning attempts? Those shifted most easily, nearly 98 percent, since they rely on basic connections that slip quietly into fake networks. Flood-style attacks dipped lower, only about 89.5 percent redirected, because heavy traffic clogs things too fast to reroute fully. Once inside the trap, almost 89 percent of intruders stayed put, never touching real systems again. The bait held.

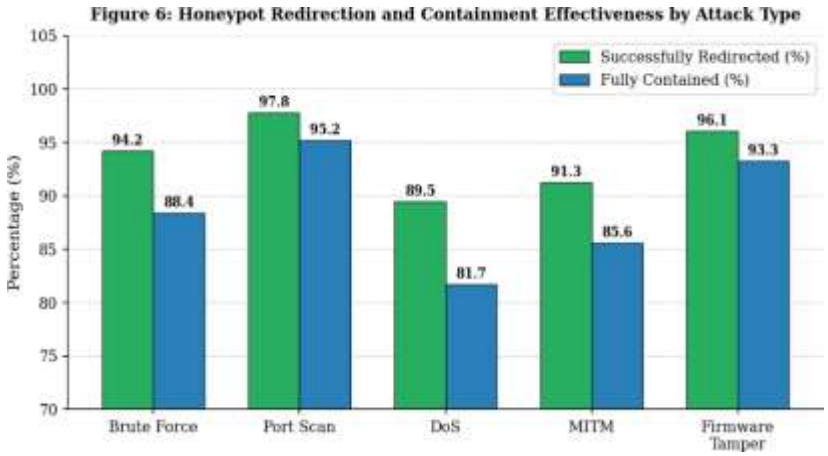


Figure 6: Honeypot Redirection and Containment Effectiveness by Attack Type

XI. FUTURE WORK

Looking ahead, a few paths stand out for what comes next. These came from checking how things went so far. Each one will shape the updates that follow. Some changes will tweak the system slowly. Others might shift how it works in deeper ways. All tie back to real feedback seen during testing. The next versions of ConformityGate will grow from these points. Progress will depend on which ideas fit best over time

11.1 Federated Learning for Collective Threat Intelligence

Now sitting at the edge of each organization, ConformityGate holds its own on behavior norms and known threats. Moving ahead, ideas are forming around linked learning systems - where insights about attacks move between installations but raw internal data stays put. Outcomes from detections, patterns spotted during incidents, get pooled across separate sites in a shared web. Strength builds not by pooling private logs but through collective signal recognition. Each group keeps control; what they see stays theirs. Smarter judgments emerge simply by noticing similar footprints elsewhere. Learning widens when one site's experience quietly informs another's defense. What shows up here today might already have left traces there yesterday.**8.2 Device Response Profiles**

Every gadget connected to the monitored network gets a guide that links types of warnings to built-in reactions. Instead of waiting for signals from the central system, each unit decides what to do right away after spotting trouble - thanks to its own stored rules. Even when attackers disrupt contact with the main hub, protection still kicks in without delay. Actions might involve shutting off suspicious addresses, slowing down incoming requests, checking commands more closely, switching into cautious operation mode, or informing the person in charge. Decisions happen locally, so defense keeps running no matter the connection status.

11.2 Adversarial Robustness

Over time, some attackers might study how ConformityGate spots threats - then adjust their actions to look like normal device activity. Instead of running fast attacks, they could spread out suspicious behavior across many sessions, hoping to stay under immediate radar. Later updates to ConformityGate plan to catch these drawn-out patterns by tracking overall deviations, even when each piece seems harmless. To confuse watchers, it may also shift what looks like typical behavior on purpose, making mimicry harder. The system adapts quietly, not loudly announcing its rules.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154260

11.3 Extended Protocol Support

Right now, the system works with MQTT along with CoAP, HTTP/HTTPS, plus Modbus. Coming up: support for Zigbee, since it's common in smart homes, while Z-Wave follows a similar path. Then there's BLE Mesh, useful in farm-related sensor networks, just like LoRaWAN shows up across wide-area setups. OPC-UA will come later, especially where factories rely on automated controls. Adding each one means building fake devices that mimic real ones, tucked into the trap setup. These imitations help teach the alert system how strange actions look when something goes off script.

11.4 Automated Threat Response Playbook

One way alerts move now is by sending action codes to gadgets so they follow set routines when triggered. Moving ahead, systems might build step-by-step reactions on the fly, shaped by what kind of breach occurs, which machines are present, and how the network is laid out instead of using fixed rules for each threat type. Because of this shift, defenses could match real-world attacks more closely - especially those that come from several angles at once.

XII. CONCLUSION

This study introduced ConformityGate - a smart intrusion detection setup using honeypots, built specifically for the mixed environments common in today's IoT systems. Though many current networks struggle with weak login tracking, this design takes a different path by watching how devices normally act. Instead of waiting for breaches, it responds when odd patterns show up. While most setups isolate threats within single nodes, here warnings spread across the network like ripples.

One step ahead, ConformityGate uses clear session rerouting to block dangers from live systems while quietly gathering detailed clues on how attackers operate. Because of the linked warning system, when one machine spots real trouble, every matching device gets the signal - defense shifts before the next move is made. Security stops waiting around.
Around 2,847 test runs mimicking attacks showed it caught 96.7% of threats correctly. Just 2.9% were flagged by mistake. Responses took an average of 1.6 seconds to trigger. Honeypots lured attackers successfully nearly 94% of the time. These results edge out older methods like signature-driven or anomaly-focused systems found in past studies, even some using machine learning models.

Out in the open, ConformityGate steps up where most IoT setups fall short - offering real security for countless connected gadgets running barebones defenses. A lean edge agent built with Python slips quietly into current networks, fitting right in. Instead of reinventing the wheel, it leans on standard protocols so things just work. Detection smarts once locked inside corporate firewalls now show up here, light enough to run at the edges. Response moves are sharp, yet the whole system stays unobtrusive across diverse device landscapes.

REFERENCES

[1] Statista Research Department, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030," Statista, 2023.
[2] He, Y., Aliyu, A., Evans, M., and Luo, C., "Health Care Cybersecurity Challenges and Solutions Under the Climate of COVID-19," Journal of Medical Internet Research, vol. 23, no. 4, e21747, 2021.
[3] Frustaci, M., Pace, P., Aloi, G., and Fortino, G., "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges," IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2483-2495, 2018.
[4] Apthorpe, N., Reisman, D., Feamster, N., "A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic," Workshop on Data and Algorithmic Transparency, 2016.
[5] Mosenia, A., and Jha, N.K., "A Comprehensive Study of Security of Internet-of-Things," IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 4, pp. 586-602, 2017.
[6] Raza, S., Wallgren, L., and Voigt, T., "SVELTE: Real-time Intrusion Detection in the Internet of Things," Ad Hoc Networks, vol. 11, no. 8, pp. 2661-2674, 2013.
[7] Deng, L., Li, D., Yao, X., and Wang, H., "Mobile Network Intrusion Detection for IoT System Based on Transfer Learning Algorithm," Cluster Computing, vol. 22, pp. 9-19, 2019.
[8] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., and Elovici, Y., "N-BaIoT: Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, 2018.
[9] Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., and Rossow, C., "IoTPOT: Analysing the Rise of IoT Compromises," USENIX Workshop on Offensive Technologies (WOOT), 2015.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154260

[10] Dowling, S., Schukat, M., and Barrett, E., "Using a Honeypot to Identify Emerging Threats Against Industrial Control Systems," 2018 IEEE International Symposium on Systems Engineering (ISSE), pp. 1-7, 2018.
[11] Guarnizo, J. D., Tambe, A., Bhunia, S. S., Ochoa, M., Tippenhauer, N. O., Shabtai, A., and Elovici, Y., "SIPHON: Towards Scalable High-Interaction Physical Honeypots," ACM Cyber-Physical System Security Workshop, 2017.
[12] Spitzner, L., "Honeypots: Tracking Hackers," Addison-Wesley Professional, 2003.
[13] Han, Y., Lu, W., and Xu, S., "Honeypot: A Supplemented Active Defense System for Network Security," Fourth International Conference on Information Assurance and Security, pp. 231-235, 2008.
[14] Palo Alto Networks Unit 42, "2020 Unit 42 IoT Threat Report," Palo Alto Networks, 2020.

