Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 5, 2026

IMPROVING IOT INTRUSION DECEPTION THROUGH NAF-ENHANCED Q-LEARNING-BASED ADAPTIVE HONEYPOTS

[1]Munazzah Munwer, [2]Shaina Laraib, [3]Rubab Haider, [4]Binish William, [5]Uzair Saeed, [*6]Abdul Rauf

[1-6]Department of Computer Science The University of Faisalabad Faisalabad, Pakistan
[1]munazzahaineelala@gmail.com, [2]shainalaraib.cs@tuf.edu.pk, [3]rubabhaiderawan@gmail.com, [4]komalbenishwilliam@gmail.com, [5]uzairsaeed.cs@tuf.edu.pk, [*6]abdulrauf.cs@tuf.edu.pk

DOI:-

Keywords
Internet of Things (IoT), Q-Learning, honeypot, Normalized Advantage Function (NAF), SSH Honeypo

Article History
Received: 17 April 2026
Accepted: 22 May 2026
Published: 23 May2026

Corresponding Author: *

Abdul Rauf

Abstract
The rapid growth of Internet of Things (IoT) en vironments has increased the vulnerability of connected devices to malware, botnets, credential abuse, and distributed attacks. Honeypots are widely used to monitor attacker behavior and collect threat intelligence; however, many existing solutions re main static, are vulnerable to fingerprinting, or provide limited interaction realism. This paper presents an adaptive SSH-based honeypot framework for IoT-oriented deception environments using Q-learning with a Normalized Advantage Function (NAF) enhanced action-value formulation. The proposed system models attacker behavior at the command level and dynamically selects deception actions such as allowing, blocking, delaying, or gen erating fake outputs. To support efficient operation and forensic analysis, the framework employs a dual-layer storage design consisting of lightweight XML event logs and extended MySQL session records. The prototype was implemented in Python and evaluated in a controlled environment using simulated adversarial SSH interactions. Experimental results show that the system sup ports scalable deployment and sustained up to 28 simultaneous honeypot instances on the tested hardware configuration, while also enabling adaptive command handling and reverse-Turing test-oriented interaction logic. The main contribution of this work is a practical command-level adaptive deception mechanism for SSH honeypots that moves beyond static or rule-based decoy models. The findings provide a basis for future work on real world validation, anti-fingerprinting strategies, and coordinated multi-honeypot deployment for IoT security.

I. Introduction

Internet of Things (IoT) systems are increasingly used in smart homes, healthcare, industry, agriculture, and urban infrastructure, creating a tremendous surface of cyber-attacks. IoT ecosystems involve massive numbers of interdependent devices, which can often be built on different types of firmware, with widely varying communication stacks and varying security settings. Due to their access to the internet, substandard credential practices, and software updates, they are tempting targets for botnets, malware propagation, creden- tial abuse, and distributed denial-of-service (DDoS) attacks. [1]–[3].

While essential cybersecurity tools like firewalls and intru-ion detection systems (IDSs) continue to play a critical role in cyber defense, they currently lack sufficient visibility into the intent of attackers and their post-compromise activities. In many cases, these defenses are geared to prevent or de-tect malicious traffic rather than investigating the adversary interactions with the system when access attempts start. In consequence, they provide only minimal information about attacker strategies, command execution patterns, malware de- ployment and system probing behaviour. This is where honey- pots come in: they are controlled decoy systems that can draw in malicious traffic, allowing for in-depth analysis of attacker behavior in a controlled environment [4]–[6].

A honeypot is a security resource whose value lies in being probed, attacked, or compromised under controlled conditions [3]. By emulating legitimate systems, services, or devices, hon- eypots provide valuable intelligence regarding attacker tools, techniques, and objectives. They are commonly categorized according to their interaction level, such as low-interaction and high-interaction honeypots, and according to their deployment purpose, such as production and research honeypots [4]. As shown in Fig. 1, a honeypot can be positioned within a network infrastructure as a concealed server that appears to be a normal node. In such a setting, suspicious traffic may be diverted toward the honeypot, allowing malicious behavior to be observed while reducing direct exposure of production resources.

Despite their practical relevance, many existing honeypot systems remain static in nature. They expose fixed con-figurations, predefined responses, and predictable interaction patterns regardless of the attacker's behavior. Such limitations reduce realism and increase the likelihood of fingerprinting by skilled adversaries. Resource-constrained networked en- vironments beyond IoT, such as wireless sensor networks, have also motivated deception-oriented protection strategies, including camouflage and concealment mechanisms for de- fending critical nodes against targeted attacks [7].Over time, honeypot research has evolved from basic trap-based sys- tems toward virtualized, high-interaction, and more recently adaptive deception environments capable of modifying their behavior according to attack context [5], [8]. This transition is particularly important in IoT environments, where large-scale automated probing and bot-driven interactions are common.

To improve realism and engagement, recent research has explored self-adaptive honeypots that employ machine learn- ing and reinforcement learning techniques to select responses dynamically. Instead of relying solely on handcrafted rules, adaptive honeypots evaluate attacker actions, infer behavioral context, and choose suitable responses to prolong interaction and collect more informative attack traces. This capability is especially useful in SSH-based IoT attack scenarios, where adversaries commonly perform reconnaissance, issue shell commands, attempt malware downloads, and probe system behavior to determine whether the target is real or deceptive. This paper proposes an enhanced adaptive honeypot frame-work that integrates continuous Q-learning with Normalized Advantage Functions (NAF) to improve the learning capabil- ity, scalability, and deception effectiveness of IoT honeypots. The proposed framework consists of three principal modules: a Honeypot Module, an Action Module, and a Q-learning with NAF Module. The Honeypot Module manages attacker inter- action and command handling, the Action Module executes deception responses such as allow, block, delay, and fake out- put, and the Q-learning with NAF Module performs intelligent action selection based on observed system state. In addition, the framework incorporates dual-level attack logging through XML and MySQL storage to support efficient monitoring and
forensic analysis.The key contributions include:

- integration of NAF-enhanced Q-learning for optimized response strategies
- a modular architecture with honeypot, actions, and NAF modules
- dual-level attack logging using XML and MySQL storage
- improved reverse Turing test capabilities for distinguish-ing human attackers from automated bots.

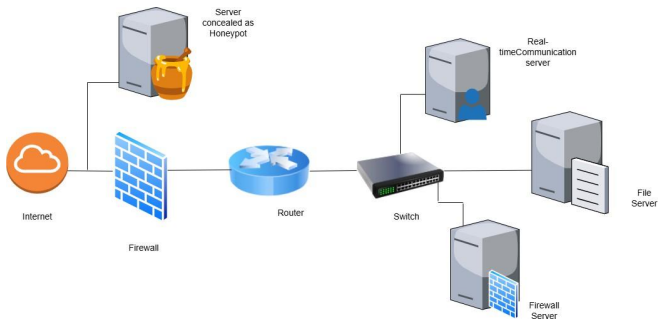


Fig. 1. Honeypot deployment within a network infrastructure: the server concealed as honeypot attracts and diverts malicious traffic away from production servers

II. Related Work

Honeypots have evolved from static decoy services into adaptive deception platforms. Foundational work established the role of deception in information protection and demon- strated how monitored decoys can be used to study adversarial behavior in controlled settings [4]–[6]. Over time, research shifted from simple low-interaction traps toward virtualization, service emulation, and high-interaction designs that expose richer attacker behavior while attempting to limit risk to production systems [8].

In the IoT domain, several notable systems have shaped the literature. SIPHON proposed a scalable high-interaction physical honeypot architecture that exposes a small num- ber of real IoT devices through globally distributed worm- holes [9]. ThingPot demonstrated an interactive IoT honeypot that emulates not only application protocols but an entire IoT platform using XMPP and REST interfaces [10]. U- PoT addressed UPnP-based IoT devices through protocol- aware emulation [11], while IoTCandyJar explored intelligent- interaction honeypots that learn behavioral responses for het- erogeneous IoT devices [12]. These systems increased realism, but many still rely on relatively fixed interaction logic or target only specific device classes.

A parallel line of work has focused on adaptive honey- pots driven by reinforcement learning. RASSH introduced an adaptive SSH honeypot that uses reinforcement learning to interact with attackers [13]. QRASSH extended this direction using Deep Q-Learning for response selection during live SSH sessions [14]. Pauna et al. later studied reward design in self- adaptive IoT honeypots and showed that reward functions have a strong influence on engagement quality and malware collec- tion [15]. Heliza explored adaptive command blocking, error return, and attacker provocation in high-interaction environ- ments, highlighting the promise of behavior-driven deception but also its limitations in realism and linguistic scope [16]. Additional work has examined hiding honeypot functionality through reinforcement learning so as to reduce early attacker abandonment [17]. Related adaptive-system research in wire- less sensor networks has also explored memory- and context- enriched models that improve future system responses by reusing prior observations, which conceptually supports the learning-oriented direction taken in this work [18].

More recent literature reinforces the relevance of learning-based IoT deception. Javadpour and Conti surveyed cyber-deception techniques and identified realism, stealth, adapt-ability, and integration with broader detection pipelines as persistent challenges [1]. Ilg et al. surveyed contemporary open-source honeypots and discussed detection and evasion techniques that affect real deployments [8]. Morozov et al. proposed an adaptive framework for design and evaluation in IoT environments [2], while Lanz et al. reviewed machine- learning optimization strategies for IoT honeypots and high- lighted the need for efficient adaptive behavior under re- source constraints [19]. Guan et al. presented learning-based IoT honeypots for cyber deception in IEEE Security & Pri- vacy [20], and Guan and Cao further proposed coordinated cyber-physical deception through IoT honeypots at USENIX Security 2025 [21].

Related work also connects honeypot telemetry with ma-chine learning for downstream detection and profiling. Lee et al. combined honeypots with machine learning for classifying IoT botnet attacks in a smart-factory setting [3]. Ahmed et al. similarly used honeypot-driven data for attack detection in IoT ecosystems [22]. Kristyanto et al. compared reinforcement learning algorithms for adaptive SSH honeypots and reported that Q-learning variants remain a useful design space for response optimization [23]. Together, these studies suggest that adaptive honeypot research now sits at the intersection of cyber deception, protocol realism, machine learning, and scalable deployment.

Despite these advances, existing honeypots either remain too static, depend heavily on handcrafted policies or reward tuning, or lack a practical command-level adaptive deception mechanism that is lightweight enough for operational deploy- ment while still supporting session-aware analysis. This paper addresses these limitations by combining attacker behavior in- terpretation, NAF-enhanced Q-learning, and dual-layer storage in a medium/high-interaction SSH honeypot.

III. Proposed Method

A. System Architecture

The proposed system is an adaptive SSH honeypot designed for IoT-oriented threat observation in a controlled environment. It is implemented in Python and organized into five major layers: command interpretation, attacker behavior modeling, NAF-enhanced Q-learning decision making, action execution, and dual-layer logging. The overall objective is to keep intrud- ers engaged long enough to reveal useful behavioral patterns while reducing risk to the underlying system. The proposed architecture integerates three core modules: Honeypot Module, Q-Learning with NAF Module and Action Module as shown in Fig 2.

Honeypot Module: This module extends existing SSH hon-eypot capabilities with additional commands (e.g., useradd, userdel, ifconfig, wget, ping, etc.) and operates in a proxy mode between the Linux OS shell and the emulated shell accessed by attackers.In this proxy mode, the honeypot module functions as an intermediary—before executing any command at the operating system level, it waits for action decisions from the NAF module. This ensures that every attacker interaction is governed by the learned policy rather than default system behavior.

Action Module: This intermediary module bridges the NAF decision engine and the honeypot's command execution layer. It implements four distinct response types determined by the NAF module, each designed to serve different deception objectives:

- Allow—permits full command execution, enabling the attacker to proceed and reveal further intentions.
- Block—returns a contextual error message from the database without executing the command, frustrating the attacker while maintaining the illusion of a real system;
- Fake Output—prints a fabricated response stored in the XML database, deceiving the attacker into believing the command succeeded.
- Delay—introduces a configurable timed pause before execution, simulating system load while buying time for analysis.

Q-Learning with NAF Module: This module implements continuous Q-learning with Normalized Advantage Functions to optimize the honeypot's response strategy. NAF is a com- plementary approach to model-based algorithms that reduces sample complexity for continuous control tasks. Unlike tra- ditional DQN approaches that require discretization of the action space, NAF uses a single neural network to handle continuous state-action spaces, providing better performance with fewer training samples. The module maps honeypot states to the general reinforcement learning components: Agent (hon- eypot), Environment (attackers), States (current interaction context), Reward function (payoff from responses), and Q-values (learned action-value estimates)

B.Intrder Behavior Modeling

Effective deception requires accurate modeling of attacker behavior. The system identifies intruder actions through three sequential stages that progressively reveal attack intent:

- CheckHWConf: The intruder examines hardware configu- ration files of the system. This initial reconnaissance helps the attacker determine whether to continue or abort the attack based on the target's hardware capabilities. The honeypot monitors and logs all hardware-related queries to build an attacker profile.
- CheckSWConf: The intruder inspects software settings to understand the compromised system. Current software configurations may reveal password storage locations, ease information theft, or enable system lockout. Attackers at this stage may also attempt to block system administrators or other competing hackers. The honeypot captures all software-related queries for behavioral analysis.
- ChangeSysConf: The intruder attempts to modify system configurations. This represents the most aggressive phase where attackers actively alter system settings. The honeypot tracks all modification attempts to understand the attacker's ultimate objectives. Fig. 3 illustrates the complete intruder behavior identification model.

C.Attack Logging & Data Storage

The framework employs a dual-level storage architecture de-signed to balance processing speed with comprehensive data retention. This split architecture is critical for maintaining hon- eypot efficiency during active engagements while preserving complete attack records for post-incident analysis.

Primary Storage (XML): An initial XML file with .ews extension serves as a lightweight data store recording timestamps, IP addresses, port IDs, and executed commands. This file functions as an enhanced log with rapid lookup capability.

When a new IP address connects, this lightweight repository is searched first for quick determination of whether the IP has been involved in previous attacks, enabling immediate behavioral classification without the overhead of querying the full database.

Secondary Storage (MySQL): A comprehensive relational database stores complete session-level data including full interaction sequences between the honeypot and intruders. Each session captures the complete attack trace—every command issued, every response given, timing data, and behavioral classifications. This repository supports complex queries for pattern analysis across multiple attack sessions and serves as the training data source for the NAF learning module. Table III details the data elements captured at each storage level.

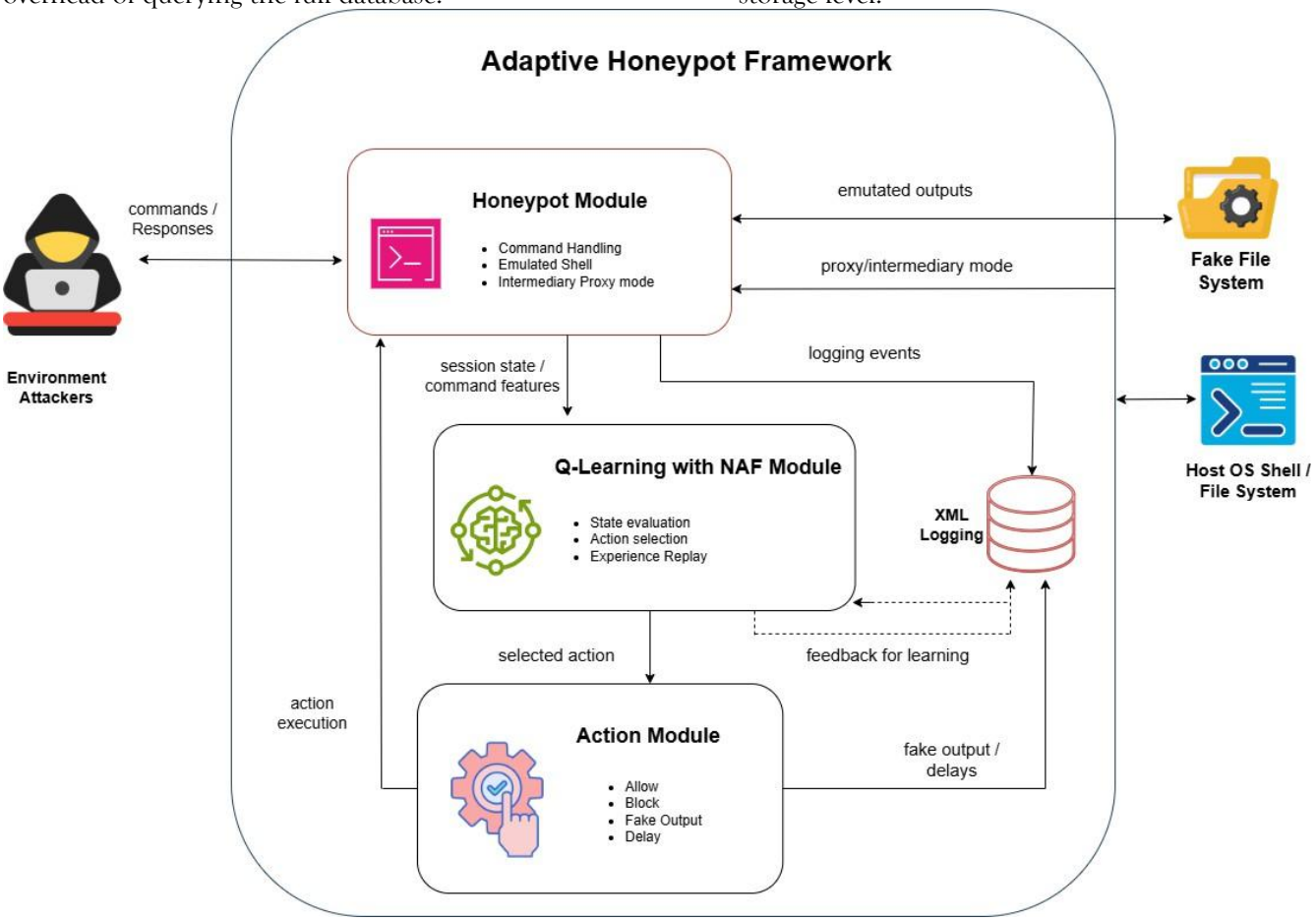


Fig. 2. Proposed Architecture: Q-Learning with NAF

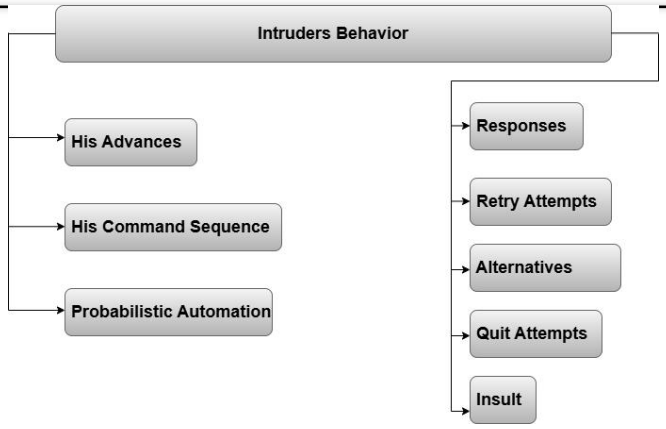


Fig. 3. Intruder Behavior Identification

D. Q-Learning with Normalized Advantage Functions (NAF)

To support adaptive response selection in continuous-control settings, the proposed framework employs Q-learning with Normalized Advantage Functions (NAF). Unlike conventional Deep Q-Networks (DQN), which are mainly designed for discrete action spaces, NAF provides a continuous-action formulation of Q-learning using a single neural network. This makes it suitable for adaptive honeypot environments, where the system must select response actions dynamically according to the observed attacker state.

In NAF, the action-value function is decomposed into a value term and an advantage term as

$$Q(x, u) = V(x) + A(x, u), \quad (1)$$

where x denotes the current state and u denotes the selected action. This formulation enables efficient learning in continu- ous state-action spaces while avoiding the need for a separate actor network.

During training, the model maintains a normalized Q-network $Q(x, u \mid \theta Q)$ and a target network $Q'(x, u \mid \theta Q)$. Transitions are stored in a replay buffer and sampled in minibatches to stabilize learning. The Q-network is updated by minimizing the loss

$$L = \frac{1}{N} \sum_{i=1}^{N} y_i - Q(x_i, u_i \mid \theta^Q)^2, \quad (2)$$

where yi is the target value computed from the observed reward and the estimated next-state value. The target network is updated using a soft-update rule to improve convergence stability. The overall NAF-based learning procedure used in the proposed framework is summarized in Algorithm 1.

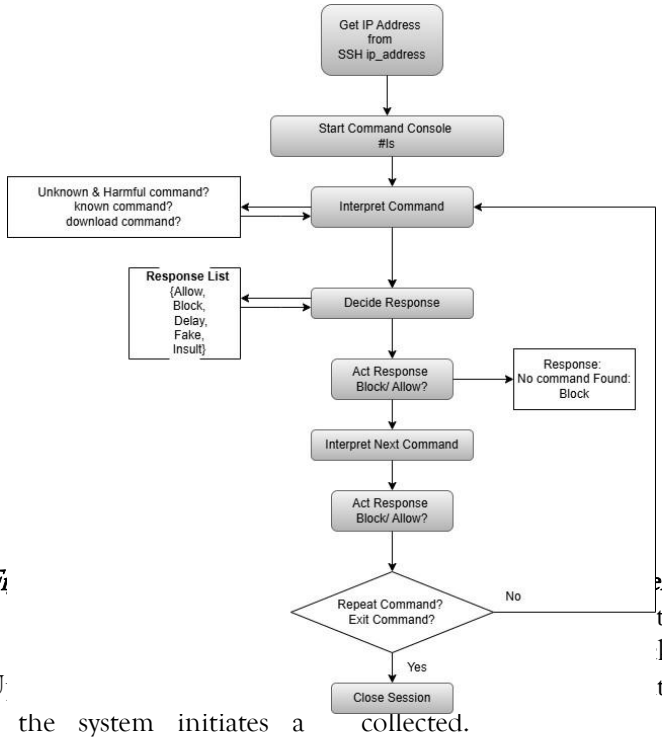
In the proposed honeypot, this module is responsible for evalu- ating attacker interaction states and selecting suitable response actions that improve deception quality, sustain engagement, and support adaptive command-level behavior.

Table I: Data Elements Captured At Each Storage Level

Data Element	XML (Primary)	MySQL (Secondary)
Timestamp	✓	✓
IP Address	✓	✓
Port ID	✓	✓
Executed Command	✓	✓
Command Sequence	-	✓
Session Data	-	✓
Honeypot Response	-	✓
Intruder Response	-	✓
Repeat Commands	-	✓
Behavioral Classification	-	✓

Algorithm 1 Q-Learning with Normalized Advantage Functions (NAF)

Randomly initialize normalized Q-network $Q(x, u \mid \vartheta^Q)$
Initialize target network Q' with weights $\vartheta^{Q'} \leftarrow \vartheta^Q$
Initialize replay buffer $R \leftarrow \emptyset$
for episode = 1 to M **do**
  Initialize a random process N for action exploration
  Receive initial observation state $x_1 \sim p(x_1)$
  for $t = 1$ to T **do**
    Select action $\mu_t = \mu(x_t \mid \vartheta^\mu) + N_t$
    Execute μ_t and observe reward r_t and next state x_{t+1}
    Store transition $(x_t, \mu_t, r_t, x_{t+1})$ in R
    for iteration = 1 to I **do**
      Sample a random minibatch of m transitions from R
      Compute target value $y_i = r_i + \gamma V'(x_{i+1} \mid \vartheta^{Q'})$
      Update ϑ^Q by minimizing $L = \frac{1}{N} \sum_i (y_i - Q(x_i, u_i \mid \vartheta^Q))^2$
      Update target network: $\vartheta^{Q'} \leftarrow \tau\vartheta^Q + (1-\tau)\vartheta^{Q'}$
    end for
  end for
end for



Fi *em*

E. Operational Workflow

Fig. 4 illustrates the complete proposed honeypot system. U connection from an attacker, the system initiates a command console and begins interpreting each received command. Commands are classified as known, unknown/harmful, or download instructions. For each command, the system consults a response list comprising Allow, Block, Delay, Fake, and Insult options.

The response decision is driven by the NAF module's learned policy based on the current state (attacker behavior clas- sification) and historical payoff data. After responding, the system interprets the next command. If a repeated command is detected, indicating potential DDoS or flooding behavior, the system may terminate the session to tion. Otherwise, the interaction ker issues an exit command or the t maximum intelligence has been collected.

IV: Experimental Setup

The prototype was implemented in Python and evaluated in a controlled environment using simulated adversarial SSH interactions. The implementation environment includes Python libraries, XML-based logging, MySQL-backed extended stor- age, and isolated deployment mechanisms. Harmful commands such as download, installation, and system-query actions were permitted only under controlled conditions so that realistic in- teraction could be stimulated without exposing the underlying host to unacceptable risk. This experimental framing is aligned with common

evaluation practices in prior adaptive and IoT- oriented honeypot research [2], [10], [11].

The tested hardware configuration consisted of an Intel Core i7 processor with 8 GB of RAM. This hardware context is important because the observed scalability of the system is bounded by the available local resources rather than by a high- capacity server environment.

The evaluation was organized around three aspects:

1)Scalability: concurrent honeypot instantiation under the tested hardware constraints.

2)Reverse-Turing-Test-Oriented Interaction: the ability to support interaction patterns that help distinguish repetitive automated behavior from more deliberate attacker activity.

3)Adaptive Learning Capability: the system's ability to choose responses according to attacker behavior and main- tain useful engagement.

V. Results and Discussion

The experimental results indicate that the proposed adaptive SSH honeypot is operationally feasible and capable of support- ing behavior-aware deception in a controlled IoT-oriented en- vironment. In contrast to a purely static decoy, the framework supports scalable deployment, command-level interaction anal- ysis, adaptive response handling, and dual-layer logging. The findings are discussed with respect to scalability, incoming instruction patterns, interaction statistics, reverse-Turing-test- oriented behavior, and adaptive learning capability.

A.Scalability Analysis

Scalability is an important requirement for any honeypot system intended to operate under varying request loads. A honeypot can be considered scalable if it is capable of evolving according to incoming requests without being limited to a fixed number of instances. In the proposed system, honeypot logging is automatic and does not require additional configuration after the first startup. The prototype includes a logging driver that interacts directly with the monitoring environment, which enables multiple honeypot instances to be executed without manual monitoring reconfiguration.

To evaluate this capability, a bash-script-driven docker run cycle was used to launch repeated honeypot instantiations. Un- der the tested hardware configuration, the prototype sustained up to 28 simultaneous honeypot instances. This limit was not predefined by the architecture itself; rather, it depended on the computational resources available at the host side. The experiment was conducted on an Intel Core i7 laptop with 8 GB RAM, and therefore the observed limit should be interpreted as a hardware-bounded scalability result rather than a hard system constraint.

This result shows that the proposed honeypot can support multi-instance deployment in a resource-constrained local en- vironment. Although this does not yet establish large-scale cloud readiness, it provides clear evidence that the system is capable of evolving according to incoming request load and available host resources.

Table II: *Observed Scalability Result of the Proposed Prototype*

Hardware	Deployment Style	Max. Concurrent Instances
Intel Core i7, 8 GB RAM	Bash-script-driven container deployment	28

Table III: *Incoming Instruction Frequency*

Sr. No.	Instruction	Frequency (%)
1	id	18.7
2	exit	16.9
3	wget	12.9
4	cd	9.6
5	last	7.5
6	w	8.2
7	ps	6.6
8	uname	5.1
9	Other	14.3

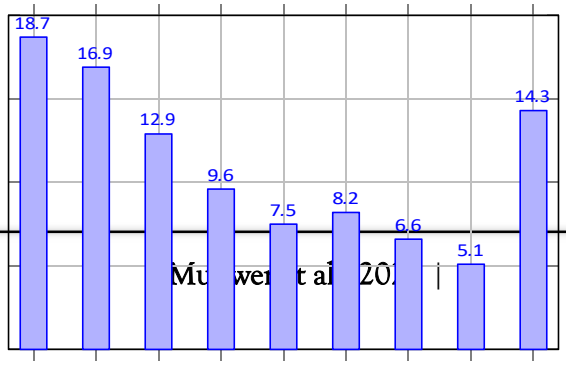


Fig. 5. Frequency analysis of incoming SSH instructions observed by the honeypot.

B.Incoming Instruction Frequency

The developed honeypot was also evaluated in terms of the frequency of incoming attacker instructions. The observed instruction distribution provides insight into the types of ac- tions most commonly attempted during SSH-based attacks. As shown in Table III, the most frequent incoming instruction was id (18.7%), followed by exit (16.9%) and wget (12.9%). These results suggest that attackers commonly begin with system identification and reconnaissance commands, followed by attempts to download external content or tools.

These findings are useful because they confirm that the hon- eypot is capturing realistic attacker interaction patterns rather than only artificial or isolated events. The prominence of reconnaissance-oriented commands also supports the need for command-level adaptive response handling in the proposed framework.

Table IV: *General Evaluation Statistics*

Sr. No.	Parameter	Value
1	Total attacks performed	250
2	Minimum duration of attack	3 s
3	Maximum duration of attack	25 s
4	Average duration of interaction	12 s
5	Actions allowed	37%
6	Actions blocked	17%
7	Actions altered	29%
8	Response messages to attacker	19%

Fig. 6. Distribution of honeypot responses during evaluation.

C.General Evaluation Statistics

The evaluation further reports general statistics obtained from
250 SSH attack attempts observed by the honeypot. The minimum recorded interaction duration was 3 s, the maximum duration was 25 s, and the average duration of interaction was 12 s. These values indicate that the system was able to engage attackers for measurable interaction periods while preserving session-level monitoring.

The response distribution is also informative. Across all ob- served interactions, 37% of actions were allowed, 17% were blocked, 29% were altered, and 19% involved a response message to the attacker. This shows that the proposed honeypot does not rely predominantly on blocking. Instead, it favors a more interactive strategy in which attackers are engaged, redirected, or provided with controlled deceptive responses. These results strengthen the argument that the proposed system behaves as an adaptive honeypot rather than a purely restrictive filter. The balance between allowed, altered, and blocked actions suggests that the system attempts to preserve attacker engagement while still controlling operational risk.

D.Reverse-Turing-Test-Oriented Interaction

A particularly important design goal of the proposed frame- work is its support for reverse-Turing-test-oriented interaction. In this context, the purpose is not merely to insult or frustrate the attacker, but to provoke interaction that helps distinguish human-guided attackers from automated bots. Instead of rely- ing on aggressive or random insulting behavior, the proposed honeypot uses interactive prompts such as “is it everything that you want to do” to elicit a response from the attacker. This interaction strategy offers two advantages. First, if the attacker responds meaningfully, the response itself can provide insight into the human operator’s language, possible location, or behavioral style. Second, if the attacker

abandons the session after receiving such a message, this can also serve as a useful behavioral indicator. Therefore, the honeypot uses interaction as a behavioral inference mechanism rather than only a blocking mechanism. This makes the reverse-Turing- test capability more practical and less dependent on simple insult-based heuristics.

The current paper does not yet provide a dedicated benchmark for human-versus-bot classification; therefore, this capability should be framed as an architectural and behavioral strength rather than a fully benchmarked detection result. Even so, it clearly increases the practical intelligence value of the prototype.

E.Adaptive Learning Capability

Adaptive honeypots derive much of their value from their learning capability. In the proposed system, attacker actions, responses, and behavioral traces are stored in the repository until explicitly removed by an administrator. This design enables the framework to identify repeated attacks, compare current and historical attacker behavior, and select more suit- able response options when a similar attacker or attack vector reappears.

The learning mechanism operates in two complementary ways. First, it uses interaction-based inference to identify character- istics of the intruder. For example, if the source IP address and the attacker's language response do not align, the system may infer the use of a rebound machine or connection laundering. Second, once an attack is completed, the honeypot uses the stored interaction history and payoff outcomes to estimate the most effective response for future attacks of a similar type. The decision process is based on payoff-driven action selec- tion. If a particular interaction causes the intruder to deviate from the attack path or abandon the session, that action receives a higher payoff. In this way, the honeypot learns from previous engagements and updates response probabilities dynamically. The framework therefore operationalizes adaptive deception by linking historical interaction outcomes with fu- ture response selection. This behavior is also consistent with broader adaptive-system research, where previously observed contexts and stored responses are reused to reduce repeated processing and improve future decision quality [18].

F.Discussion

Taken together, the results support a moderate but credible claim for the proposed system. The framework demonstrates multi-instance scalability, captures realistic SSH attacker com- mand patterns, supports adaptive interaction beyond sim- ple blocking, and incorporates a learning-oriented response- selection strategy. The most important quantitative strengths are the observed support for 28 simultaneous instances, the analysis of 250 attacks, and the balanced response distribution showing that the honeypot does not rely solely on restrictive responses.

Compared with purely static honeypots, the proposed sys- tem offers a more interactive and behavior-aware design. Its strongest contribution lies in combining adaptive response logic with practical deployment and logging mechanisms. At the same time, the results should still be interpreted within the limits of a controlled environment. The present evaluation does not yet include extensive baseline comparison, convergence analysis, or large-scale production deployment. Nevertheless, the results provide meaningful evidence that the proposed adaptive SSH honeypot is both feasible and practically useful as a foundation for more advanced IoT-oriented deception systems.

VI. Conclusion and Future Work

This paper presented an adaptive SSH honeypot framework for IoT-oriented deception environments that combines attacker behavior modeling, Q-learning with Normalized Advantage Functions (NAF), and dual-layer operational logging. The framework is designed to support command-level adaptive deception rather than a purely static or rule-based decoy strategy. The implemented prototype demonstrates that the proposed architecture is feasible in practice and can sustain up to 28 simultaneous honeypot instances under the tested hardware configuration.

The study makes three main contributions. First, it introduces an adaptive honeypot architecture in which response selection is guided by a NAF-enhanced Q-learning module. Second, it integrates a modular design comprising the Honeypot Module, Action Module, and Q-learning with NAF Module for con- trolled and behavior-aware interaction. Third, it employs dual- level XML and MySQL logging to support both lightweight operational monitoring and richer post-session forensic analy- sis.

Although the results are encouraging, they should be inter- preted within the scope of the current evaluation. The proto- type was assessed in a simulated attack environment, which may not fully capture the complexity and unpredictability of real-world adversaries. In addition, the present framework does not yet provide complete anti-fingerprinting protection against advanced honeypot-detection strategies. The XML/MySQL storage design, while practical and effective in the current setup, may also introduce bottlenecks under higher attack volumes or larger-scale deployments. Furthermore, the current study

does not include a comprehensive quantitative comparison against established SSH honeypot baselines. These limitations are aligned with concerns raised in recent surveys on cyber deception, open-source honeypot evaluation, and machine-learning-based honeypot optimization.

Future work should therefore focus on extending the evalua- tion in four directions. First, direct baseline comparison should be performed against static and established SSH honeypots to quantify the benefits of adaptive response selection. Second,

richer quantitative metrics should be introduced, including at- tacker session duration, command diversity, response realism, and interaction retention. Third, stronger anti-fingerprinting mechanisms should be incorporated to reduce the risk of early honeypot detection. Finally, the framework should be evaluated in larger-scale and more realistic distributed IoT environments in order to assess its robustness, scalability, and practical deployment value.

References

[1]A. Javadpour and M. Conti, "A comprehensive survey on cyber- deception techniques for improving intrusion detection and honeypots," Computers & Security, vol. 140, p. 103285, 2024.

[2]D. S. Morozov, A. A. Yefimenko, T. M. Nikitchuk, R. O. Kolomiiets, and S. O. Semerikov, "The sweet taste of iot deception: An adaptive honeypot framework for design and evaluation," Journal of Edge Com- puting, vol. 3, no. 2, pp. 207–223, 2024.

[3]S. Lee, A. Abdullah, N. Z. Jhanjhi, S. C. Kok, R. Chandran, and

R. Gururajan, "Classification of botnet attacks in iot smart factory using honeypot combined with machine learning," PeerJ Computer Science, vol. 7, p. e350, 2021.

[4]B. Cheswick, "An evening with berferd in which a cracker is lured, endured, and studied," in Proceedings of the Winter USENIX Conference, 1992, pp. 163–174.

[5]L. Spitzner, Honeypots: Tracking Hackers. Addison-Wesley, 2003.

[6]F. Cohen, "A note on the role of deception in information protection," Computers & Security, vol. 17, no. 6, pp. 483–506, 1998.

[7]S. Ubaid, M. F. Shafeeq, M. Hussain, A. H. Akbar, A. Abuarqoub, M. S. Zia, and B. Abbas, "Scout: A sink camouflage and concealed data delivery paradigm for circumvention of sink-targeted cyber threats in wireless sensor networks," The Journal of Supercomputing, 2018.

[8]N. Ilg, P. Duplys, D. Germek, and M. Menth, "A survey of contemporary open-source honeypots, frameworks, and tools," Journal of Network and Computer Applications, vol. 220, p. 103737, 2023.

[9]J. D. Guarnizo, A. Tambe, S. S. Bhunia, M. Ochoa, N. O. Tippenhauer, A. Shabtai, and Y. Elovici, "Siphon: Towards scalable high-interaction physical honeypots," in Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security, 2017, pp. 57–68.

[10]M. Wang, J. Santillan, and F. Kuipers, "Thingpot: An interactive internet- of-things honeypot," arXiv preprint arXiv:1807.04114, 2018.

[11]M. A. Hakim, H. Aksu, A. S. Uluagac, and K. Akkaya, "U-pot: A honeypot framework for upnp-based iot devices," in 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC), 2018, pp. 1–8.

[12]T. Luo, X. Xu, F. Wang, P. Peng, K. Xu, and Q. Li, "Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices," Black Hat USA, 2017.

[13]A. Pauna and I. Bica, "Rassh: Reinforced adaptive ssh honeypot," in RoEduNet International Conference — Networking in Education and Research, 2014, conference metadata should be verified against the final publisher record.

[14]A. Pauna, N. Iacob, and I. Bica, "Qrassh: A self-adaptive ssh honeypot driven by q-learning," in 2018 International Conference on Communi- cations (COMM), 2018.

[15]A. Pauna, I. Bica, F. Pop, and A. Castiglione, "On the rewards of self- adaptive iot honeypots," Annals of Telecommunications, vol. 74, no. 7–8,

pp. 501–515, 2019.

[16]G. Wagener, R. State, A. Dulaunoy, and T. Engel, "Heliza: Talking dirty to the attackers," Journal in Computer Virology, vol. 7, no. 3, pp. 221– 232, 2011.

[17]S. Dowling, H. Janicke, J. Organ, and P. A. H. Williams, "Using reinforcement learning to conceal honeypot functionality," in Machine Learning and Knowledge Discovery in Databases Workshops (ECML PKDD Workshops), 2018.

[18]M. Hussain, M. F. Shafeeq, S. Jabbar, A. H. Akbar, and S. Khalid, "Cram: A conditioned reflex action inspired adaptive model for context addition in wireless sensor networks," Journal of Sensors, vol. 2016, p. 6319830, 2016.

[19]S. Lanz, S. L. Pignol, P. Schmitt, H. Wang, M. Papaioannou, G. Choud- hary, and N. Dragoni, "Optimizing internet of things honeypots with machine learning: A review," Applied Sciences, vol. 15, no. 10, p. 5251, 2025.

[20]C. Guan, J. Zhang, G. Cao, T. F. L. Porta, and J. C. Acosta, "Learning- based internet of things honeypots for cyber deception," IEEE Security & Privacy, vol. 23, no. 6, pp. 58–65, 2025.

[21]C. Guan and G. Cao, "Cyber-physical deception through coordinated iot honeypots," in 34th USENIX Security Symposium (USENIX Security 25), 2025, pp. 529–546.

[22]Y. Ahmed and coauthors, "Securing smart cities through machine learning: A honeypot-driven approach to attack detection in internet of things ecosystems," IET Smart Cities, 2024, article metadata should be checked against the final publisher record.

[23]M. A. Kristyanto and coauthors, "Evaluation and comparison of the use of reinforcement learning algorithms on ssh honeypot," TEKNIKA, 2024, final volume, issue, and page metadata should be verified.