International Journal of Safety and Security Engineering
Vol. 15, No. 5, May, 2025, pp. 873-882
Journal homepage: http://iieta.org/journals/ijsse

Next-Gen Cloud Security: IRDS4C's Deception Strategy for Early Intrusion and Ransomware Detection

Ahmed El-Kosairy*, Nashwa AbdelBaki

Centre of Informatics Science, School of Information Technology and Computer Science, Nile University, Giza 12588, Egypt

Corresponding Author Email: ah.elkosairy@nu.edu.eg

https://doi.org/10.18280/ijsse.150502

Received: 7 March 2025
Revised: 15 April 2025
Accepted: 19 May 2025
Available online: 31 May 2025

Keywords:
deception systems, fake files, decoy, honeypots, intrusion detection, ransomware detection

ABSTRACT

Cloud computing is rapidly expanding, offering users efficient and cost effective data storage and management. However, increased adoption has also amplified security vulnerabilities, particularly from sophisticated threats such as ransomware, which encrypts user data and demands payment for its release. Traditional cybersecurity methods are insufficient against these evolving threats, particularly zero-day attacks and advanced persistent threats (APTs). This paper introduces IRDS4C, an Intrusion and Ransomware Detection System specifically designed for cloud infrastructures, leveraging innovative deception strategies. Unlike conventional systems that rely on signature based or behavioral detections, IRDS4C strategically deploys decoy mechanism including carefully positioned fake files, high and low interaction honeypots, and decoy resources to proactively mislead and capture malicious actors. By positioning these decoys in commonly targeted locations and utilizing naming strategy optimize for early interaction, IRDS4C rapidly identifies ransomware activity and intrusions. Experimental validation demonstrates that IRDS4C significantly outperforms traditional methods such as file hashing and entropy analysis, achieving faster detection with higher accuracy. Consequently, IRDS4C effectively prevents attackers from accessing critical production data, marking a substantial advancement in proactive cloud security.

1. INTRODUCTION

Cybersecurity is an ongoing battle between security professionals and cybercriminals. As organizations trying to change their info and on-prem to cloud technologies, hackers continue to develop more advanced attack methods to exploit vulnerabilities. Cyber threats have evolved beyond simple malware infections and phishing attempts attackers now leverage sophisticated techniques, including zero-day exploits, ransomware, and APTs to compromise cloud infrastructures. Offering scalability, flexibility, without cost consumption. However, this widespread adoption has also introduced security challenges. Web applications and cloud based systems are prime targets for cyberattacks due to the shared responsibility model, where security obligations are divided between cloud providers and users. Misconfigurations, weak authentication, and insider threats further increase the risk of data breaches. To protect cloud environments, organizations use IDS, IPS, and Web Application Firewalls (WAF). While these tools help monitor and prevent known threats, they rely heavily on signature based detection [1]. This approach is ineffective against zero-day attacks [2], where hackers exploit unknown vulnerabilities before security teams can respond. Moreover, behavior analysis and AI security systems, while promising, often generate false positives and require frequent updates to stay effective [3, 4]. A more proactive approach is needed one that goes beyond traditional security measures. Additionally, using advanced techniques to detect unknown attacks is beneficial but insufficient by itself. Approaches relying on Artificial, Machine Learning (AI&ML), and Behavioral Analysis require significant time and extensive tuning to minimize false positives. Moreover, these methods can be costly and often necessitate additional infrastructure. Since time is critical, most organizations aim to reduce the Time to Detect (MTTD) and the Time to Respond (MTTR). Therefore, relying on sophisticated detection methods only is not adequate, particularly across various cloud service layers. This paper introduces IRDS4C, an intrusion and ransomware detection system designed to combine traditional security with decoy techniques. IRDS4C provides multi layered detection across all cloud layers, from the network border gateway down to the hardware and server level. The core idea is to use sophisticated deception techniques to mislead attackers and detect abnormal behavior at every cloud layer, regardless of the cloud service type. However, in this paper, our primary focus is on the OS layer, as most targeted attacks such as ransomware and data breaches predominantly aim to compromise or corrupt this specific layer [5]. IRDS4C was tested on Google Cloud and provides an innovative way to detect and stop cyber threats before they compromise cloud environments. This paper explores how IRDS4C enhances security across these models by detecting and neutralizing intrusions and ransomware attacks before they cause significant damage.

2. BACKGROUND

The cloud introduces many benefits, such as scalability, cost efficiency, and flexibility, but it also introduces new attack surfaces that cybercriminals actively target. Unlike traditional on-premises systems, where organizations have full control over their security infrastructure, cloud environments operate under a shared security responsibility model. Unfortunately, many organizations fail to properly configure their cloud environments, leaving them vulnerable to cyberattacks. One of the most dangerous threats to cloud environment today is ransomware, which has become a serious problem for businesses and government institutions alike. As cloud computing keeps expanding, so do the security problems that come with it. The cloud brings many advantages like scalability, cost savings, and flexibility, but at the same time, it creates new opportunities for cybercriminals. Unlike traditional IT systems where companies have full control over their security, cloud environments follow a shared responsibility model. The issue is, many organizations fail to configure their cloud environment properly, which makes them easy targets for hackers.

2.1 Ransomware history

Ransomware has evolved significantly since its inception, becoming one of the most pressing cybersecurity threats today [6]. The first known ransomware attack, the PC Cyborg Trojan (also known as AIDS), emerged in 1989. Created by Dr. Joseph L. Popp, this malware was distributed via 20K floppy disks with name "AIDS Information – Introductory Diskettes" to attendees of the World Health Organization's AIDS conference. Once installed, it encrypted file in Panama to restore [7]. In the early 2010s, ransomware attacks became more sophisticated and widespread. CryptoLocker, which appeared in 2013, utilized strong encryption algorithms and demanded payments in Bitcoin, making it nearly impossible to recover files without paying the ransom [8]. Similarly, TeslaCrypt, discovered in 2015, targeted specific file types, particularly those associated with video games, and was distributed through exploit kits like Angler [9]. The ransomware landscape further transformed with the advent of Ransomware as aService (RaaS). This approach allows hackers to use ransomware tools to affiliates, lowering the barrier to entry for launching attacks. Notable RaaS groups include REvil, Ryuk, Conti, and DarkSide, which have been responsible for high-profile attacks on various sectors, including healthcare, education, and critical infrastructure. Understanding the historical evolution of ransomware, from the AIDS Trojan to modern RaaS operations, underscores the increasing complexity and threat posed by these attacks. This context highlights the necessity for advanced detection and prevention systems like IRDS4C, which are designed to address the sophisticated nature of contemporary ransomware threats.

2.2 Ransomware types and families

There are different kinds of ransomware, each with its own way of attacking victims [10]:

- Encrypting Ransomware: The most common type, where files are encrypted and attackers demand payment for the decryption key. Examples include WannaCry, REvil, and LockBit.
- Locker Ransomware: Instead of encrypting files, this type locks victims out of their own computers or systems, showing a ransom message. Examples include WinLocker and Petya.
- Scareware: This tricks victims into thinking their system is infected with malware, pushing them to buy a fake antivirus tool. It does not lock or encrypt files but causes panic.
- Doxware (Leakware): Hackers steal critical info and informed the client that it will be released it unless the client/victim pays a money as a (ransom). Maze and Babuk are examples of this method.
- Ransomware Worms: These spread across networks without user interaction. NotPetya and WannaCry spread automatically by exploiting system vulnerabilities.

2.3 Traditional ways to detect ransomware

Most ransomware detection methods rely on signature-based detection, which is used by traditional antivirus and security software. This method scans files and compares them to a database of known ransomware signatures [11]. While this is effective against older threats, it has major weaknesses:

- Fails Against Zero-Day Attacks: Since this method relies on existing malware databases, it cannot detect new ransomware variants that have not been documented yet.
- Evasion Tactics: Modern ransomware uses techniques like code packing, polymorphism, and encryption to change its appearance and avoid detection.
- Constant Updates Needed: Security tools must update frequently to recognize new ransomware versions, but attackers create new threats faster than updates can keep up.

Some other traditional detection methods include heuristic analysis, which looks for suspicious behavior, and sandboxing, where files are tested in a controlled environment before running on a real system. While these methods provide better protection than signature-based detection, they still struggle against advanced ransomware that can bypass security controls.

2.4 A smarter approach with deception based security

Because of these limitations, researchers have explored deception-based techniques such as honeypots and decoy systems to catch hackers and study their methods. Deception security has been around since 1991 and works by misleading attackers into revealing themselves [12]. By placing fake files, decoy tokens, and monitored honeypot environments, security teams can detect ransomware early before real data is compromised. The IRDS4C framework is built on these decoy techniques. It is designed to detect both ransomware and intruders in cloud environments before they can cause damage. In this section, we will discuss real-world ransomware incidents, how cloud storage synchronization can help ransomware spread, and why deception-based security is an effective way to improve cloud security. We will also look at different detection methods, such as file hashing, entropy analysis, API hooking, and decoy resources, and compare how effective they are against cloud-based attacks.

3. LITERATURE REVIEW

DDS Safe, a cloud service provider backup and storage system for dentists, got infected with the ransomware called ReVil in 2019. In the study [13], about 400 clinics couldn't access their data and were out of service. Another big attack was NotPetya. NotPetya takes advantage of system weaknesses to break into networks and operating systems. Unlike many other ransomware types, it doesn't need to trick a user into clicking anything [14]. The objective is to encrypt the files and data before getting noticed. This type of attack is known as "Ransomware-as-a-Service" (RaaS). Examples include REvil, WannaCry, BadRabbit, and NotPetya. Understanding how ransomware infects cloud storage is very important. One of the biggest risks is how cloud storage syncs with local storage. Services like Google Drive let users work on the local folders, and any changes automatically update in the cloud [15]. Ransomware takes advantage of this to spread.To protect cloud servers with sensitive data, researchers focus on detecting intruders. Since 1991, decoy systems have been used to confuse hackers [16]. In the study [17], author showed how honey files and tokens can trick attackers. Similarly, Whitham [18] explained how honeypots can detect unauthorized access by feeding intruders fake data. Decoy files and tokens help track unauthorized activity in cloud servers, and decoy resources can also be used to detect ransomware or malware trying to steal or encrypt sensitive files [19, 20]. The IRDS4C model is built on decoy techniques [21]. These techniques are very important for keeping the system stable and secure. It is based on four main ideas to fool hackers and make them interact with the decoy resources. The first one is making sure the fake resources look real enough to convince the attacker. The second one is reachability, which means placing decoys in common folders where ransomware usually looks for files. The third one is diversity—using different types of decoys at the same endpoint so intrusions and ransomware can be caught quickly. The fourth is making sure that real users don't mistakenly interact with decoy files. To catch intrusions within the cloud system, decoy endpoints, honey folders, and honey tokens are used.

Honey files come in two types [21]: "high" and "low" interaction files. High-interaction files confuse hackers and keep them engaged with fake data. Low-interaction decoy files, on the other hand, quickly detect scripts or ransomware trying to steal or encrypt files. Different kinds of decoy servers [22] and fake tokens (such as decoy emails, fake directories, or fake pictures). The more honey servers are used [23, 24], the better the detection rate. Many techniques exist to detect ransomware, both in cloud and on-premises setups.

This paper focuses on detecting ransomware in Windows systems and protecting files from being encrypted or corrupted. Microsoft Windows is the commone used OS worldwide, with over 70% shares. Linux and Unix will be explored in future work. One common technique is Comparing Files Hashing, which calculates the hash value of files and checks for changes over time [25]. Another method is Comparing File Entropy Value, which measures randomness in file data to detect suspicious activity [26]. A third method uses hooking API functions, which allows the system to monitor file system activity for unusual changes [27]. Finally, decoy assets use a positioning technique to trick ransomware into interacting with them first as listed in Table 1.

Our paper uses file *event-handler* watching with fake methods and reallocating ways. If ransomware or any code/script tries to access IRDS4C's low-interaction files filled with random data, it gets detected instantly. The high-interaction decoy files mislead hackers and push them deeper into fake traps. The decoy resources include fake folders, partitions, and storage, helping detect intrusions, unauthorized access, and ransomware attacks. Using both high and low-interaction files ensures that both simple and advanced attacks can be caught effectively.

Table 1. Differences among detection methods for ransomware [28]

Methods	Strengths	Downsides
Comparing files hashing	Works on all windows platforms with accurate results	Uses a lot of RAM and system resources
Comparing file entropy value	Works on all windows platforms	Can have false positives, also consumes RAM and resources
Hooking file system API functions	Accurate results, no extra RAM usage	Not compatible with all Windows versions, ransomware can sometimes detect it
Decoy resources using positioning technique	Accurate detection, no extra RAM usage	Needs proper file placement, only works with NTFS

4. IRDS4C SYSTEM CHARACTERISTICS

The IRDS4C system is developed around the approach/idea of using fake files and tokens to catch intruders and ransomware in cloud servers. These decoy resources could be anything such as a student's report card, a harmless looking image, or even a fake word document. The content inside might not be important, but it's enough to lure attackers. If an intruder tries to access these deceptions [29], an alert is triggered, making it clear that unauthorized activity, possibly ransomware, is happening.

4.1 IRDS4C and reallocating method to detect ransomware

IRDS4C detects ransomware by placing decoy files in a way that makes them the first target of an attack. In Windows, ransomware searches for files using *Find_First_File* (FFF) & *Find_Next_File* (FNF) [30]. These methods sort files based on their ANSI/ASCII values in ascending order. Ransomware usually goes after important files like documents and spreadsheets stored in common directories. For example, ransomware prioritizes files with extensions like (.pdf, .docx, .txt) inside folders such as My Documents, Desktop, and the C:\ drive. To take advantage of this behavior, IRDS4C places decoy files at the top of the search results, ensuring they are attacked first. If ransomware encrypts or modifies these fake files, our system detects it immediately and stops the attack. To make sure these decoy files appear first, we use special file names that start with characters like "#" or "!" since ANSI and ASCII sorting prioritize these symbols [31]. For instance, naming a file #00000.txt or!.jnt ensures it is listed at the top. This trick makes ransomware interact with the decoy files before any real data is affected, allowing IRDS4C to detect and respond in time as illustrated in Table 2.

Table 2. ANSI code characters table

Dec.	Hex	Unicode	Char.
32	20	U+0020	
33	21	U+0021	!
34	22	U+0022	”
35	23	U+0023	#
36	24	U+0024	$
37	25	U+0025	%
38	26	U+0026	&
39	27	U+0027	’
40	28	U+0028	(
41	29	U+0029	)
42	2A	U+002A	*
43	2B	U+002B	+
44	2C	U+002C	,
45	2D	U+002D	-
46	2E	U+002E	.
47	2F	U+002F	/
48	30	U+0030	0

4.2 Misleading contents and IRDS4C

IRDS4C doesn’t just place fake files—it also uses deceptive content to trick attackers into revealing their presence. This method, called High Interaction Mode, misleads hackers by providing them with fake but realistic-looking information. For example, a decoy file might appear to contain login credentials for another system, but it directs the attacker to another fake server where their actions can be monitored.

A similar trick is used with honey tokens, where fake registry entries suggest remote desktop connections leading to a non-existent system inside the cloud. These honey files, in different formats like .pdf, .txt, and .docx, are placed in well-known locations such as My Desktop and My Documents. Hackers are naturally drawn to these files because they believe they contain something valuable [32].

To further tempt attackers, these files are given attention-grabbing names like “Password_List.docx” or “Confidential_Financial_Report.pdf”. The moment any of these fake files are opened, IRDS4C detects it and raises an alert [33].

Besides decoy files, different decoy methods are used based on the cloud model (IaaS, PaaS, or SaaS). Table 3 shows how different types of decoys are applied [34, 35].

Table 3. Deception systems methods for cloud system

Cloud Model	IaaS	PaaS	SaaS
Honey Net (Decoy Routers & Switches)	✓	✓	✓
Decoy Hypervisor	✓	✓	✓
Decoy Endpoints	✓	✓	✓
Decoy Folder&Files for OS	✓	✓	
Face Tokens for OS	✓	✓	
Face Tokens for Web Apps		✓	✓

The IRDS4C system architecture is shown in Figure 1, demonstrating how decoy system is applied to detect ransomware and intrusions. It works by creating decoy resources at multiple levels inside cloud environments.

- For cloud servers: Decoy files, folders, website directories, and email accounts are used to detect breaches.
- For cloud networks: Decoy servers and IP addresses are used. These include Honey endpoints, and Honey IPs.

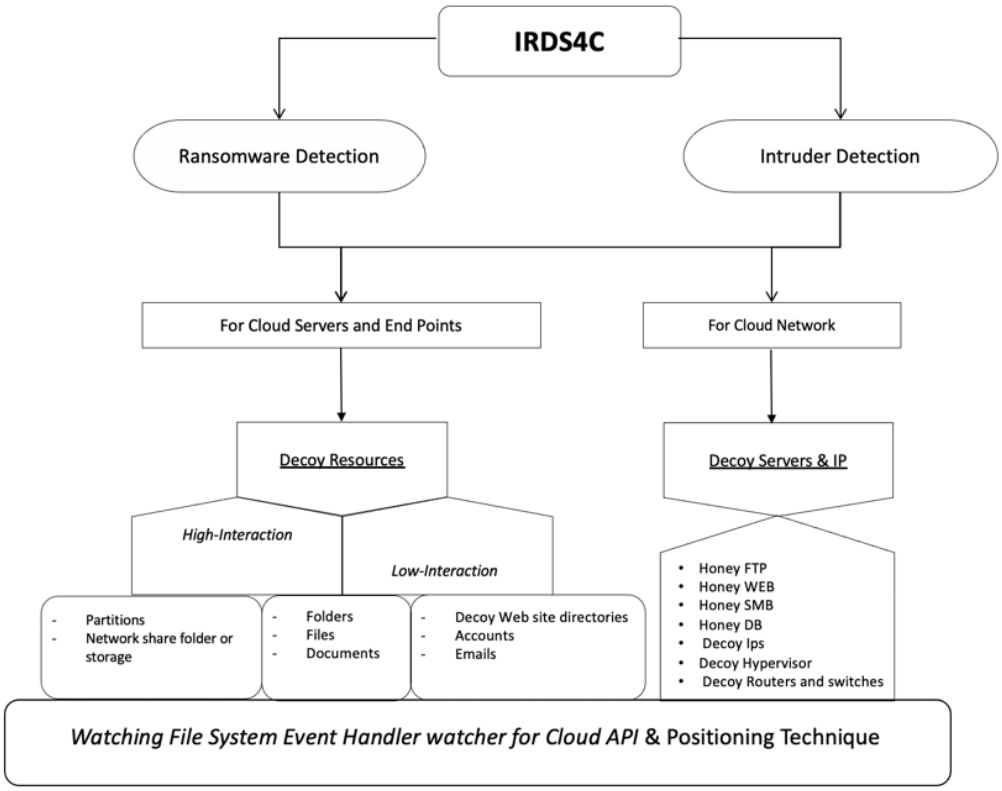

Figure 1. Proposed IRDS4C framework structure [36]

All activities around these decoys are closely monitored using the *event_handler* watcher for cloud APIs & hypervisors, ensuring that any suspicious activity is detected and responded to in real-time.

By combining all these deception strategies, IRDS4C significantly improves cloud environment security by catching threats before they can cause real harm [36].

5. IRDS4C EVALUATION

We tested a series of experiments to check if our IRDS4C system really works. The results of these tests are explained in this section. We set up IRDS4C on Google Cloud Platform [37], as seen in Figure 2, and applied and tested it on (I-P-S-aaS) models. To properly test IRDS4C, we built different test environments and ran three experiments focusing on ransomware and intrusions. Experiment 1: We tested IRDS4C against various ransomware samples. The goal was to measure how fast it detects threats and how accurate it is in identifying different types of ransomware attempt 2: We took the same ransomware samples from attempt 1 and tested them using two other well known methods: File Hashing and File Entropy techniques. Then, we compared their results with IRDS4C's performance to see which method was better. Experiment 3: We checked how well IRDS4C can catch intruders. To do this, we made SaaS and PaaS portals publicly available on the internet and observed if IRDS4C could detect unauthorized access attempts in a smart way. These experiments helped us understand IRDS4C's strengths and how well it performs in real world scenarios.

The IRDS4C framework was subjected to rigorous testing to evaluate its effectiveness in detecting and mitigating ransomware and intrusion activities within cloud environments. The evaluation was conducted using a diverse set of ransomware samples, including prominent variants such as WannaCry, Ryuk, and Sodinokibi, across different cloud models.

For the experimental setup, the IRDS4C framework was deployed on the Google Cloud Platform and tested with cloud layers. Multiple experiments were conducted using a diverse set of ransomware samples and intrusion scenarios [36].

VMware Setup:

- Software Version: VMware Workstation Pro 17.0
- Virtual Machines: Each machine had:
  - 4 CPUs
  - 8 GB RAM
  - 100 GB SSD storage
  - Network: Bridged mode with individual virtual network adapters
- Snapshots: We created snapshots right after installing and setting up everything. This way, every test started from the exact same point.

Operating Systems Used:

- WinServer_2019
- Ubuntu Server 20.04.3 LTS

OS Configuration:

- Both systems were fully updated as of January 2025.
- Windows Defender antivirus was turned off to focus on how IRDS4C itself detects threats.
- Ubuntu's built-in firewall (UFW) was enabled with basic settings, and SELinux was set to permissive mode for controlled testing.
- We ran tests on both default setups and hardened security setups to see how different security settings affect IRDS4C's detection.

Test Environment Setup:

- Network**:** All virtual machines were linked together using a dedicated virtual switch inside VMware. Each had a unique static IP address, and firewall rules were set clearly to allow ransomware and simulated attack activities.
- Simulating Attacks: We used specific scripts to run ransomware samples (7 samples), and for intrusions like SQL injections or privilege escalation, we used standard security testing tools like Metasploit and OWASP ZAP.

This approach let us accurately see how well IRDS4C spots and reacts to threats.

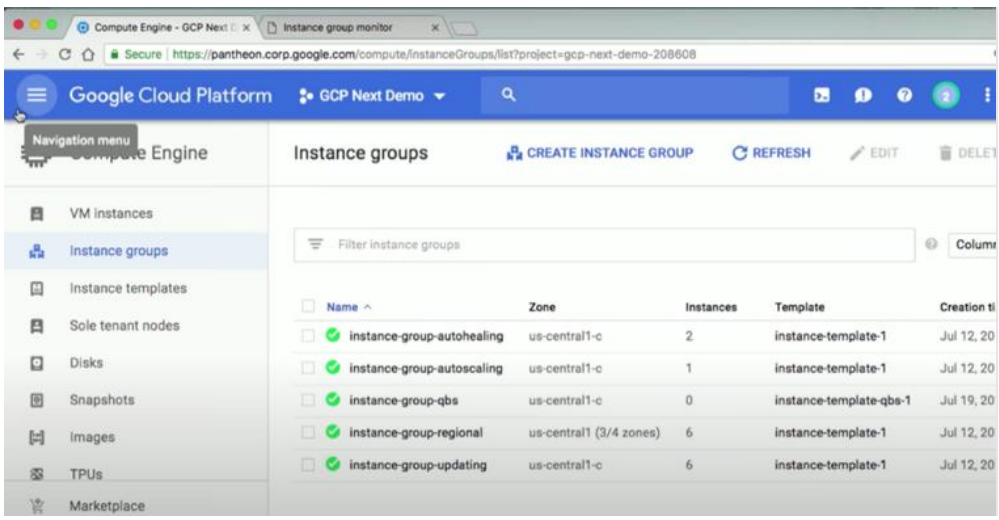

Figure 2. Google cloud platform dashboard

Table 4. IRDS4C and ransomware detection

Family	# Sample	Detected After	Detection Rate	Type(s) of Attack		
				File Encryption	Stealing Info	Deleting Files
BadRabbit	20	12 sec	20/20	√		√
Phobos (RedLine)	11	9 sec	11/11	√	√	
Sodinokibi (REvil)	6	15 sec	6/6	√	√	√
Ryuk	6	10 sec	6/6	√	√	
WannaCry	4	10 sec	4/4	√		√
Notpetya	3	17 sec	3/3		√	√
Netwalker	3	14 sec	3/3	√		√
Total	53		53/53			

Table 5. Results

Ransomware Family	IRDS4C Detection Time	File Hashing Detection Time	File Entropy Detection Time	IRDS4C Detection Score	File Hashing Score	File Entropy Score
BadRabbit	12 sec	50 sec	52 sec	20/20	20/20	18/20
Phobos (RedLine)	9 sec	35 sec	11 sec	11/11	11/11	10/11
Sodinokibi (REvil)	15 sec	55 sec	18 sec	6/6	6/6	3/6
Ryuk	10 sec	40 sec	17 sec	6/6	6/6	5/6
WannaCry	10 sec	49 sec	40 sec	4/4	4/4	4/4
NotPetya	17 sec	30 sec	50 sec	3/3	3/3	3/3
Netwalker	14 sec	37 sec	24 sec	3/3	3/3	3/3

5.1 Attempt 1: Challenges ransomware detection

The total of 53 samples from seven notorious ransomware families BadRabbit, NotPetya, Phobos (RedLine), WannaCry, Sodinokibi (REvil), Ryuk, and Netwalker were used to evaluate the IRDS4C framework [38]. These samples were sourced from reputable malware analysis services and executed in a controlled cloud environment [39] as illustrated in Table 4 and Figure 3.

Table 4 and Figure 3 summarize the detailed evaluation results of IRDS4C's detection capability against several prevalent ransomware families. Table 3 specifically includes more information of ransomware samples tested per family, including BadRabbit (20 samples), Phobos (RedLine) (11 samples), Sodinokibi (REvil) (6 samples), Ryuk (6 samples), WannaCry (4 samples), NotPetya (3 samples), and Netwalker (3 samples). It also documents IRDS4C's response times, measured in seconds, and clearly indicates the malicious behaviors executed by each ransomware type, such as file encryption, information theft, and file deletion. Figure 4 visually illustrates the detection speeds of IRDS4C for each ransomware type, facilitating a direct comparison.

IRDS4C's detection performance was consistently strong, identifying ransomware activity rapidly, typically between 9 and 17 seconds after initial execution. Phobos ransomware showed the fastest detection at 9 seconds, while NotPetya, known for its complexity, had the longest detection time at 17 seconds. These differences in detection times can be attributed to each ransomware's distinctive behavior patterns, encryption methods, and file-access mechanisms. The consistent accuracy (100% detection) and rapid response demonstrated by IRDS4C underline its effectiveness and reliability, suggesting that employing deception-based techniques significantly enhances ransomware detection capabilities in cloud-based systems.

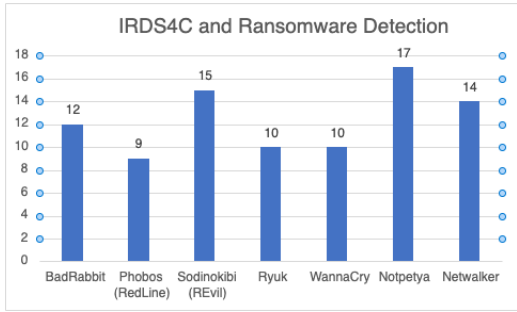

Figure 3. The detection, in seconds, per ransomware families which IRDS4C detected

5.2 Experiment 2: Comparing detection techniques

IRDS4C was compared with existing detection mechanisms like *file_hashing* and *file_entropy* techniques as illustrated in Figure 4 and Table 5. The same samples were used to assess detection times and accuracy [36].

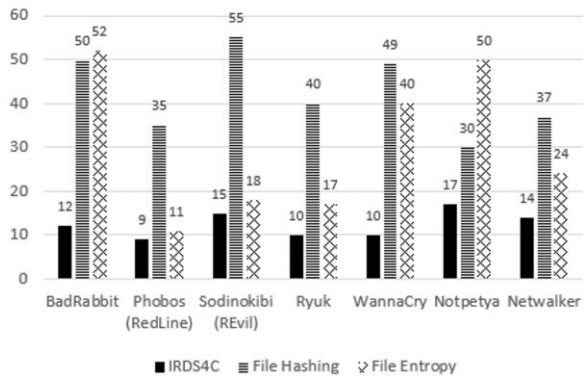

Figure 4. IRDS4C against the other methods for detection per sec [36]

5.2.1 Performance metrics

- Detection Rate: Percentage of successful detections.
- *Mean_Time_to_Detect* (MTTD): The average time from attack initiation to detection.
- False Positive Rate: Percentage of benign activities incorrectly flagged.
- Resource Utilization: CPU, memory, and network overhead.

Measuring the Mean Time to Detect (MTTD) [40] for the IRDS4C framework, file hashing, and file entropy involves calculating the average time taken to detect ransomware threats across each method. This can help demonstrate the effectiveness and efficiency of the IRDS4C system in comparison to traditional methods. Here's how to approach it:

Steps to Measure MTTD

Data Collection:

- Gather the detection times for each ransomware family for IRDS4C, file hashing, and file entropy from your experimental results.
- Create a dataset that lists the detection times for each method.

Calculate MTTD for Each Method:

- The Mean Time to Detect is calculating using the equation:

$$MTTD = \frac{\sum Detection\ Times}{N} \quad (1)$$

where, _N is the *total_number* of detected rasnomware for that method, all the results for the MTTD mentioned in Table 6.

Table 6. Mean Time to Detect (MTTD) for 3 detection methods

Detection Method	MTTD (Seconds)
IRDS4C	12.43
File Hashing	42.29
File Entropy	30.29

Results:

- Detection Rate: IRDS4C achieved 100% detection.
- MTTD: IRDS4C averaged 12.43 seconds, outperforming traditional methods.
- False Positive Rate [41]: IRDS4C recorded 0% false positives in static environment such as servers and SCADA systems. The zero false positive rate is ensured because only ransomware or attackers will access these canary files, which act as tripwires. Created in unused directories and hidden from regular users, they remain untouched by legitimate processes. Any access attempt signals malicious activity, such as file enumeration, unauthorized access, or ransomware encryption, making detection highly reliable.
- Resource Efficiency: Minimal overhead introduced.

5.3 Attempt 3: Evaluating intruders' detection

The objective of this attempt was to evaluate how well IRDS4C detects intruders trying to break into cloud systems. We set up honey files, honey tokens, and honey endpoints, then let three attackers try to access sensitive information on our test machines. We focused on monitoring the behavior of these three experienced intruders using a five-layer detection system.

- Layer 1 - Decoy Files: We developed fake folders/files with tempting names, like *Webconfig password.docx* and *Username and password.xls*, to trick intruders into interacting with them.
- Layer 2 - Decoy Tokens: We placed fake items like misleading directories, such as */wbconfign* and */admin*, to lure attackers looking for restricted access points.
- Layer 3 - Decoy Servers: We set up honey servers, such as *Honey Web* and *Honey DB*, to see if intruders would interact with them.
- Layer 4 - Decoy Partitions: We created fake storage partitions to detect access attempts.
- Layer 5 - Decoy Shared Folders: We added decoy folders to track if attackers attempted to open shared files.

5.3.1 Intruder testing results

- Intruder 1: Entered the cloud test environment at 11:00 AM. IRDS4C detected the breach after 15 minutes (11:15 AM) via Decoy Files. The attacker tried opening *Webconfig password.docx* and also accessed *Honey Web & Honey DB* servers, the decoy partition, and the decoy folder. However, they ignored Decoy Tokens and never accessed real system resources.
- Intruder 2: Started at 2:40 PM, detected 7 minutes later at 2:47 PM. This attacker interacted with Decoy Files (*Username and password.xlsx*), and also accessed Decoy Tokens, particularly the fake URL */webconfig*. They also interacted with decoy servers, partitions, and folders but never reached real files.
- Intruder 3: More careful and strategic than the others, this attacker started at 3:00 PM and was detected 13 minutes later at 3:13 PM. Instead of touching Decoy Files, they interacted with Decoy Tokens first, such as the */admin* fake directory. They also accessed *Honey Web & Honey DB* servers, the decoy partition, and decoy folders but did not reach real files as listed in Table 7.

Table 7. IRDS4C intrusion detection results

Detection Type	Intruder 1	Intruder 2	Intruder 3
Start Access Time	11:00_AM	2:40_PM	3:00_PM
Detection Time	11:15 AM (15 min)	2:47 PM (7 min)	3:13 PM (13 min)
Fake Files	*Webconfig password.docx*	*Username and password.xlsx*	Not Accessed
Fake Tokens	-	*/webconfig* fake URL	*/admin* fake URL
Fake Servers	*Honey Web & Honey DB*	*Honey Web & Honey DB*	*Honey Web & Honey DB*
Fake Partition	*(G:) Fake partition*	*(G:) Fake partition*	*(G:) Fake partition*
Fake Shared Folder	*Web_share_files*	*Web_share_files*	*Web_shar_files*
Total Fake Resources Accessed	4/5	5/5	4/5
Real Resources Accessed	0	0	0

These tests show that IRDS4C successfully tricked and detected all intruders before they could access real files. Each attacker followed a different approach, but the system identified and flagged all of them based on their interactions with our decoy system as illustrated in Figure 5.

	Intruder 1 Detected By	Intruder 2 Detected By	Intruder 3 Detected By
Detection Technique 1 *Decoy Files*	X	X	Didn't interact
Detection Technique 2 *Decoy Tokens*	Didn't interact	X	X
Detection Technique 3 *Decoy Servers*	X	X	X
Detection Technique 4 *Decoy Partition*	X	X	X
Detection Technique 5 *Decoy Share Folder*	X	X	X

Figure 5. Intruders' detection matrix

6. CONCLUSIONS AND FUTURE WORK

While traditional security measures play an essential role in protecting systems, they are no longer enough to counter modern cyber threats, especially as attacks become more advanced and specifically targeted. Cloud computing has now become a fundamental part of daily life, with almost everyone using cloud system through the laptops or cellphone devices. Therefore, a new and improved approach is needed to detect 0day attacks, particularly ransomware, and identify any form of intrusion or unauthorized access in cloud computing.

In this paper, we introduced IRDS4C, a framework designed to catch hackers and ransomware using deception techniques, such as honeycombs and tokens. The IRDS4C system relies on honeypots and decoy resources to monitor and detect threats within Win OS without consuming excessive memory, RAM/CPU, or other cloud resources.

Unlike other security solutions, IRDS4C is designed to work across different cloud system layers. We examined its functionalities and compared its performance with commonly used ransomware detection techniques, such as file hashing and entropy-based methods. Our results demonstrated that IRDS4C achieved higher accuracy rates and detection scores. Furthermore, our intrusion detection tests showed that IRDS4C was capable of identifying intruders early, preventing them from accessing any real files or sensitive data.

While IRDS4C has demonstrated strong capabilities in rapidly detecting ransomware and intrusion attempts within cloud environments, it currently has notable limitations. Specifically, the system is optimized exclusively for Windows environments and the NTFS file system [42, 43], limiting its immediate applicability across platforms utilizing Linux, macOS, SCADA, or other file systems. Recognizing these constraints, future research will focus on expanding IRDS4C's compatibility to these additional platforms and integrating advanced detection techniques such as Windows API hooking to support diverse file systems. Furthermore, incorporating artificial intelligence and machine learning approaches will enhance IRDS4C's ability to detect zero-day threats and complex cyber-attacks more effectively.

Future developments will also focus on exploitation detection techniques and signature-based threat detection to improve security. Moreover, an automated response mechanism may be introduced, allowing the system to take immediate actions upon detecting suspicious activities.

To further enhance its threat detection capabilities, we will integrate Artificial Intelligence (AI) methods to identify anomalies and unusual behavior in cloud systems. This approach will improve the framework's ability to detect and prevent sophisticated cyber threats. Additionally, machine learning will be employed to recognize and neutralize zero-day threats, making IRDS4C even more effective in combating modern cyberattacks.

REFERENCES

[1] Aradi, Z., Bánáti, A. (2025). The role of honeypots in modern cybersecurity strategies. In 2025 IEEE 23rd World Symposium on Applied Machine Intelligence and Informatics (SAMI), Stará Lesná, Slovakia, pp. 000189-000196. https://doi.org/10.1109/SAMI63904.2025.10883300

[2] Tulashvili, Y., Kosheliuk, V. (2025). Orchestrating honeypot deployment in lightweight container platforms to improve security. International Science Journal of Engineering & Agriculture, 4(1): 1-13. https://doi.org/10.46299/j.isjea.20250401.01

[3] Millar, S., McLaughlin, N., del Rincon, J.M., Miller, P. (2021). Multi-view deep learning for zero-day Android malware detection. Journal of Information Security and Applications, 58: 102718.

[4] Otoum, Y., Nayak, A. (2021). As-Ids: Anomaly and signature based Ids for the Internet of Things. Journal of Network and Systems Management, 29(3): 23. https://doi.org/10.1007/s10922-021-09589-6

[5] Almotiri, Sultan H. (2025). AI driven IOMT security framework for advanced malware and ransomware detection in SDN. Journal of Cloud Computing, 14(1): 19.

[6] Einy, S., Oz, C., Navaei, Y.D. (2021). The anomaly-and signature-based IDS for network security using hybrid inference systems. Mathematical Problems in Engineering, 2021(1): 6639714. https://doi.org/10.1155/2021/6639714

[7] Subramanya, K.P.D., Ramakrishna, P.K.H. (2025). A study on ransomware attack and its detection techniques. AIP Conference Proceedings, 3278(1): 020015. https://doi.org/10.1063/5.0262111

[8] Jabid, T., Masum, S., Shams, R.A., Chowdhury, A., Islam, M.M., Ferdaus, M.H., Ali, M.D., Islam, M. (2025). A Brief History of Ransomware. CRC Press, pp. 3-17.

[9] Zeleke, S.N., Jember, A.F., Bochicchio, M. (2025). Integrating explainable AI for effective malware detection in encrypted network traffic. arXiv preprint arXiv:2501.05387. https://doi.org/10.48550/arXiv.2501.05387

[10] Sharma, S., Kaul, A. (2021). VANETs cloud: Architecture, applications, challenges, and issues.

Archives of Computational Methods in Engineering, 28: 2081-2102. https://doi.org/10.1007/s11831-020-09447-9
[11] Mohanan, S., Sridhar, N., Bhatia, S. (2022). Comparative Analysis of Cloud Computing Security Frameworks for Financial Sector. In: Yang, XS., Sherratt, S., Dey, N., Joshi, A. (eds) Proceedings of Sixth International Congress on Information and Communication Technology. Lecture Notes in Networks and Systems, vol 236. Springer, Singapore. https://doi.org/10.1007/978-981-16-2380-6_90
[12] Nakkeeran, M., Mathi, S. (2022). A generalized comprehensive security architecture framework for IoT applications against cyber-attacks. In: Raje, R.R., Hussain, F., Kannan, R.J. (eds) Artificial Intelligence and Technologies. Lecture Notes in Electrical Engineering, vol 806. Springer, Singapore. https://doi.org/10.1007/978-981-16-6448-9_46
[13] Glover, C. (2021). Ransomcloud: How and why ransomware is targeting the cloud. https://techmonitor.ai/technology/cybersecurity/ransomcloud. Accessed on Jun. 15, 2025.
[14] Mos, M.A., Chowdhury, M.M. (2020). The growing influence of ransomware. In 2020 IEEE International Conference on Electro Information Technology (EIT), Chicago, IL, USA, pp. 643-647. https://doi.org/10.1109/EIT48999.2020.9208254
[15] Prajapati, P., Shah, P. (2022). A review on secure data deduplication: Cloud storage security issue. Journal of King Saud University-Computer and Information Sciences, 34(7): 3996-4007. https://doi.org/10.1016/j.jksuci.2020.10.021
[16] Malin, C.H., Gudaitis, T., Holt, J.H., Kilkger, M. (2017). Deception in the Digital Age. Academic Press.
[17] Reidegeld, K. A., Eisenacher, M., Kohl, M., Chamrad, D., Körting, G., Blüggel, M., Meyer, H.E., Stephan, C. (2008). An easy-to-use Decoy Database Builder software tool, implementing different decoy strategies for false discovery rate calculation in automated MS/MS protein identifications. Proteomics, 8(6): 1129-1137. https://doi.org/10.1002/pmic.200701073
[18] Whitham, B. (2013). Canary files: Generating fake files to detect critical data loss from complex computer networks. In Second International Conference on Cyber Security, Cyber Peacefare and Digital Forensic (CyberSec2013), Kuala Lumpur, Malaysia, pp. 170-179.
[19] Sudha, I., Kannaki, A., Jeevidha, S. (2014). Alleviating internal data theft attacks by decoy technology in cloud. International Journal of Computer Science and Mobile Computing, 3(3): 217-222.
[20] Virvilis, N., Vanautgaerden, B., Serrano, O.S. (2014). Changing the game: The art of deceiving sophisticated attackers. In 2014 6th International Conference on Cyber Conflict (CyCon 2014), Tallinn, Estonia, pp. 87-97. https://doi.org/10.1109/CYCON.2014.6916397
[21] Kahlhofer, Mario, Matteo Golinelli, and Stefan Rass. (2025). Koney: A cyber deception orchestration framework for kubernetes. arXiv preprint arXiv:2504.02431. https://doi.org/10.48550/arXiv.2504.02431
[22] Mondal, A., Goswami, R.T. (2021). Enhanced Honeypot cryptographic scheme and privacy preservation for an effective prediction in cloud security. Microprocessors and Microsystems, 81: 103719. https://doi.org/10.1016/j.micpro.2020.103719
[23] Gill, K.S., Saxena, S., Sharma, A. (2020). GTM-CSec: Game theoretic model for cloud security based on IDS and honeypot. Computers & Security, 92: 101732. https://doi.org/10.1016/j.cose.2020.101732
[24] Aydeger, A., Saputro, N., Akkaya, K. (2020). Cloud-based deception against network reconnaissance attacks using SDN and NFV. In 2020 IEEE 45th Conference on Local Computer Networks (LCN), Sydney, NSW, Australia, pp. 279-285. https://doi.org/10.1109/LCN48667.2020.9314797
[25] Joshi, Y.S., Mahajan, H., Joshi, S.N., Gupta, K.P., Agarkar, A.A. (2021). Signature-less ransomware detection and mitigation. Journal of Computer Virology and Hacking Techniques, 17(4): 299-306. https://doi.org/10.1007/s11416-021-00384-0
[26] Davies, S.R., Macfarlane, R., Buchanan, W.J. (2021). Differential area analysis for ransomware attack detection within mixed file datasets. Computers & Security, 108: 102377. https://doi.org/10.1016/j.cose.2021.102377
[27] Faghihi, F., Zulkernine, M. (2021). RansomCare: Data-centric detection and mitigation against smartphone crypto-ransomware. Computer Networks, 191: 108011. https://doi.org/10.1016/j.comnet.2021.108011
[28] El-Kosairy, A., Azer, M.A. (2018). Intrusion and ransomware detection system. In 2018 1st International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, pp. 1-7. https://doi.org/10.1109/CAIS.2018.8471688
[29] Negi, P.S., Garg, A., Lal, R. (2020). Intrusion detection and prevention using honeypot network for cloud security. In 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, pp. 129-132. https://doi.org/10.1109/Confluence47617.2020.9057961
[30] Lee, S., Kim, H.K., Kim, K. (2019). Ransomware protection using the moving target defense perspective. Computers & Electrical Engineering, 78: 288-299. https://doi.org/10.1016/j.compeleceng.2019.07.014
[31] Gundoor, T.K. (2025). Identifying nobel features in non-portable executable malware files. IAENG International Journal of Computer Science, 52(1): 121.
[32] Shokouhinejad, H., Razavi-Far, R., Mohammadian, H., Rabbani, M., Ansong, S., Higgins, G., Ghorbani, A.A. (2025). Recent advances in malware detection: Graph learning and explainability. arXiv preprint arXiv:2502.10556. https://doi.org/10.48550/arXiv.2502.10556
[33] Morić, Z., Dakić, V., Regvart, D. (2025). Advancing CYBERSECURITY with honeypots and deception strategies. In Informatics, 12(1): 14. https://doi.org/10.3390/informatics12010014
[34] Saxena, M.A., Ubnare, G., Dubey, A. (2019). Virtual public cloud model in honeypot for data security: A new technique. In Proceedings of the 2019 5th International Conference on Computing and Artificial Intelligence, pp. 66-71. https://doi.org/10.1145/3330482.3330516
[35] Kara, I., Aydos, M. (2022). The rise of ransomware: Forensic analysis for windows based ransomware attacks. Expert Systems with Applications, 190: 116198. https://doi.org/10.1016/j.eswa.2021.116198
[36] El-Kosairy, A., Abdelbaki, N. (2023). Deception as a service: Intrusion and ransomware detection system for cloud computing (IRDS4C). Advances in Computational

Intelligence, 3(3): 9. https://doi.org/10.1007/s43674-023-00056-0

[37] Poongodi, T., Beena, T.L.A., Sumathi, D., Suresh, P. (2022). Behavioral malware detection and classification using deep learning approaches. In Applications of Computational Intelligence in Multi-Disciplinary Research, pp. 29-45. https://doi.org/10.1016/B978-0-12-823978-0.00015-0

[38] Lai, A.C.T., Ke, P.F., Ho, A. (2025). Ransomware IR model: Proactive threat intelligence-based incident response strategy. arXiv preprint arXiv:2502.01221. https://doi.org/10.48550/arXiv.2502.01221

[39] Any Run. (2022). Ryuk analysis by Any.Run. https://app.any.run/tasks/077ab638-12e2-4a5e-95fc-302be8eb60f4/.

[40] Chatterjee, S. (2025). Using SIEM and SOAR for real-time cybersecurity operations in oil and gas. International Journal of Innovative Research and Creative Technology, 6(2): 1-11.

[41] Kaur, P., Kaur, H. (2025). Honeypots and honeynets: Investigating attack vectors. In Emerging Trends in Computer Science and Its Application, pp. 192-197. CRC Press.

[42] Peng, Z., Feng, X., Tang, L., Zhai, M. (2015). A data recovery method for NTFS files system. In International Conference on Applications and Techniques in Information Security, pp. 379-386.

[43] Chang, X., Hao, F., Wu, J., Feng, G. (2019). File recovery of high-order clearing first cluster based on FAT32. In: Vaidya, J., Zhang, X., Li, J. (eds) Cyberspace Safety and Security. CSS 2019. Lecture Notes in Computer Science, Springer, Cham, 11982. https://doi.org/10.1007/978-3-030-37337-5_38