Covert Channels in Shared Cloud Hosting: A Case Study on CPU Load-Based Data Exfiltration

Akshat Kumar, Anuj Singh
Department of Computer Science and Engineering
National Institute of Technology Delhi
Delhi, India
242211003@nitdelhi.ac.in,242211004@nitdelhi.ac.in

***Abstract*—Covert channels are secret ways to send data that can break the security of virtual machines (VMs) and containers in cloud systems. These channels use shared hardware, especially CPU cores, to pass information between two programs without being noticed. One program can change how much it uses the CPU to send signals, and another program can watch the CPU load to read those signals. This makes it possible for attackers to steal private data like passwords or secure tokens.**

This paper looks at how covert channels work when they use the CPU load. We explain how attackers can change CPU usage to send hidden messages. Tools like /proc/stat can be used to watch the CPU and understand these messages. We also talk about how these attacks can happen between containers by measuring time delays. Real examples, like the CCCV and CDMA-based attacks, are studied to show how dangerous these methods can be.

The results show that these covert channels can leak very private operations by just watching how busy the CPU is. For example, if someone is hashing a password or signing a token, the attacker might guess this activity based on CPU usage. This makes CPU-based covert channels a big risk for cloud and shared systems.

***Index Terms*—Covert Channels, Network Security, Storage Covert Channels, Timing Covert Channels,Distributed Covert Channel,Data Exfiltration, Traffic Normalization, Machine Learning, Malware Communication.**

I. Introduction

Covert channels are hidden ways of sending information. These paths are not made for regular communication. Attackers use them to send secret data by hiding it in normal network traffic [1]. This helps them steal sensitive information or perform attacks without being noticed. These channels are very hard to detect, which makes them dangerous in cybersecurity.In cloud computing, many users share the same physical computer. They do this using virtual machines (VMs) or containers. This is called shared hosting. Each user should be fully isolated and safe from others. But covert channels can break this isolation. They create secret paths between different users' systems. SXAS For example, an attacker can run a low-privileged VM or container on the same machine as a victim. If they both share the same CPU cores—like those in the same NUMA node—the attacker can watch the CPU's behavior. This helps the attacker guess what the victim is doing. The attacker might learn if the victim is typing a password, using an API, or doing other private actions like hashing.

Types of Covert Channels There are two major types of covert channels. The first is the **Storage Covert Channel**, which hides data in unused spaces of network packets. For instance, secret data can be embedded within rarely used fields of TCP/IP headers. The second is the **Timing Covert Channel**, which conveys hidden messages by altering the timing of events. An example of this is sending network packets with intentionally varied delays—short or long—to encode information covertly.

System Architecture

In cloud computing, multiple virtual machines (VMs) or containers often run on the same physical server. Although these VMs appear to be isolated from one another, they actually share physical resources such as the CPU, memory, and cache. Due to this shared hardware, covert channels can emerge, enabling secret communication between otherwise isolated systems.

A. Shared Hosting Architecture

In a shared hosting architecture, many virtual machines or containers operate on a single physical server. These environments share hardware components like CPU cores and system memory. Despite having isolated software environments, the shared hardware can create unintended interactions. Such interactions make it possible for covert communication channels to be established, which may compromise system security.

This setup allows someone in one VM to quietly observe what is happening in another VM by watching CPU behavior. This behavior can be used to send secret messages between VMs without being detected..

B. Covert Channels Explained Covert channels are Covert channels are secret ways to send messages inside a computer.

This paper talks mostly about timing channels that use the CPU. These tricks are simple and don't need to hack any software. They work just by using parts of the computer that are shared.

C. CPU Load-Based Timing Channels These covert channels work by watching how long the CPU takes to do a task. If a process uses a lot of CPU power (like when encrypting data or checking passwords), the CPU gets slower. The attacker looks for these slowdowns to figure out what's happening secretly. It's like quietly listening to someone without leaving any signs.

II. Literature Review

In cloud computing, many virtual machines (VMs) and containers run on the same physical computer. They share resources such as the CPU, memory, and cache. Even though software mechanisms are designed to keep them isolated, attackers can exploit these shared hardware components to secretly steal data.

One such method is called CCCV, introduced by Okamura and Oyama in 2010. This approach creates a covert communication channel between two virtual machines by manipulating CPU load. The goal of such techniques is to infer secret behaviors like password hashing, keystroke patterns, or JSON Web Token (JWT) signing activities without any direct network communication.

Researchers have explored various types of covert channels, including network-based, storage-based, timing-based, and hybrid channels that combine multiple techniques. In containerized environments, similar CPU-based covert channels have been demonstrated, particularly in Docker systems where misconfigurations in Linux settings allow such vulnerabilities to surface.

Advanced covert communication models such as DYST utilize normal third-party traffic to hide their secret exchanges. This strategy makes the detection of covert communication significantly more difficult.

Khadse et al. (2024) discussed numerous types of covert channels and proposed the use of machine learning methods to detect them [2]. Okhravi et al. (2010) demonstrated that combining storage and timing covert channel techniques not only makes them more difficult to detect but also improves their transmission efficiency [3]. Wendzel et al. (2025) introduced a novel technique known as a history covert channel. Instead of transmitting large volumes of new data, this method refers to previously transmitted data already present on the network, making the covert communication even more discreet and harder to uncover.

detect. [4].

Several research papers support the existence of this threat. In such scenarios, the **attacker** is typically a virtual machine (VM) or container with limited permissions. The **victim** is another VM running on the same server, often performing sensitive operations such as typing passwords or accessing authentication tokens.

A key **assumption** in these attacks is that both the attacker and the victim are executing on the same physical CPU core. Under this condition, the attacker attempts to infer the victim's actions without any direct communication, by analyzing shared resource usage patterns such as CPU or cache behavior.

III. Working

In a covert channel, information is hidden by either altering parts of the data packet that are not normally checked (e.g., header fields) or by adjusting the timing of messages. For example, a sender might delay a message by a few milliseconds to indicate a '1' and send on time for a '0' [5].

Many methods have been suggested to stop covert channel attacks that use shared CPU resources. These methods are grouped into three types: protections built into the system, actions taken by cloud service providers, and tools that help find these attacks.

A. System-Level Defenses

One important system-level defense is the disabling of hyper-threading. This prevents multiple tasks from running on the same physical CPU core simultaneously, reducing the chance that an attacker and victim will share the same core. Another effective method is core pinning, where critical tasks are assigned dedicated CPU cores. This ensures that they do not share computational resources with untrusted or unknown tasks. Additionally, scheduler isolation can be employed, which uses advanced scheduling techniques such as CPU sets to keep sensitive virtual machines (VMs) isolated from potential attackers.

B. Cloud Provider Defenses

Cloud providers have implemented several strategies to defend against covert channel attacks. One such technique is noise injection, where artificial delays or dummy tasks are introduced to obscure the actual CPU usage patterns, making it difficult for attackers to infer sensitive information. Another approach is virtual machine (VM) migration, in which VMs are periodically moved across different CPU cores. This disrupts the timing patterns that attackers rely on to extract data.

y

TABLE I
LITERATURE REVIEW: SHARED HOSTING PROVIDERS AND ASSOCIATED RISKS

Study	Focus Area	Findings	Risks Identified
Mirheidari et al. (2012)	Security mechanisms in shared hosting	Evaluated Apache security modules like suEXEC and suPHP for isolating users	Inadequate user isolation can allow cross-site code injection
Tajalizadehkhoob et al. (2017)	Statistical analysis of shared hosting security	Hosting providers significantly influence patching frequency and TLS configuration	Poor patching practices and insecure software defaults increase attack surface
Chizhov & Fesenko (2025)	Strategic evaluation of hosting services	Compared shared hosting with cloud-based alternatives regarding service flexibility	Shared hosting suffers from weak QoS, poor scalability, and performance degradation
Korczyński et al. (2015)	Behavior of shared hosting providers	Assessed how providers respond to abuse complaints like malware hosting	Many providers exhibit high tolerance for abuse and slow response to incidents
Jonker et al. (2016)	Hosting market analysis	Found that competitive pricing pressures result in cost-cutting on security practices	Limited monitoring, poor service-level agreements, and insecure configurations
Zembruzki et al. (2021)	Centralization of hosting infrastructure	Discovered over 50% of domains are hosted by the top 1% of providers	Creates single points of failure and increases risk of large-scale compromises

TABLE II
LITERATURE REVIEW: COVERT CHANNELS IN CPU OUT-OF-ORDER EXECUTION AND ASSOCIATED RISKS

Study	Focus area	Findings	Risks identified
IChannels: Exploiting Current Management Mechanisms to Create Covert Channels in Modern Processors [**?**]	Exploitation of current management mechanisms in CPUs	Demonstrated that power-hungry instructions can be used to create covert channels across different cores and threads, achieving higher bandwidth than previous methods.	Potential for unauthorized data transmission between isolated processes, challenging traditional isolation mechanisms.
BandwidthBreach: Unleashing Covert and Side Channels through Cache Bandwidth Exploitation [**?**]	Cache bandwidth exploitation in modern CPUs	Introduced covert channels by deliberately causing cache congestion, achieving high data transmission rates and bypassing existing defenses.	Enables extraction of sensitive information, such as cryptographic keys, through cache-based covert channels.
AutoCC: Automatic Discovery of Covert Channels in Time-Shared Hardware [**?**]	Automated detection of covert channels in shared hardware resources	Proposed a methodology using formal property verification to automatically discover covert channels at the register-transfer level.	Highlights the prevalence of covert channels in shared hardware, emphasizing the need for automated detection tools.
ConBOOM: A Configurable CPU Microarchitecture for Speculative Covert Channel Mitigation [**?**]	Mitigation of speculative execution-based covert channels	Presented a CPU microarchitecture design that mitigates covert channels arising from speculative execution, such as Spectre Variant 1.	Addresses vulnerabilities in speculative execution, reducing the risk of data leakage through cache-based covert channels.
Covert Channel Communication as an Emerging Security Threat in 2.5D/3D Integrated Systems [**?**]	Covert channels in advanced integrated systems	Discussed the emergence of covert channels in multi-core processors and integrated systems, emphasizing the challenges in detection and mitigation.	Highlights the increasing risk of covert channels in complex system architectures, necessitating enhanced security measures.

Furthermore, telemetry and anomaly detection systems are used to continuously monitor the system's behavior.

C. Detection Challenges

Most network systems cannot detect these changes easily. [6] Traditional firewalls and antivirus tools focus on the visible threats, while covert channels remain unnoticed. Machine learning is now being used to find unusual patterns and detect covert activity [7].

C. Tooling and Setup

The system called CCCV (Covert Channels using CPU load between Virtual Machines).

(*B. Encoding Strategy*

In the encoding strategy, the covert channel relies on interpreting CPU usage patterns as binary signals. When the victim performs CPU-intensive tasks, such as encryption operations, it is interpreted as a binary '1'. Conversely, when the victim is idle or running lightweight tasks, it represents a binary '0'. The attacker monitors this behavior by repeatedly measuring the time it takes to execute small tasks. Longer execution times suggest the CPU is busy (indicating a '1'), while shorter times suggest the CPU is idle (indicating a '0'). By performing these

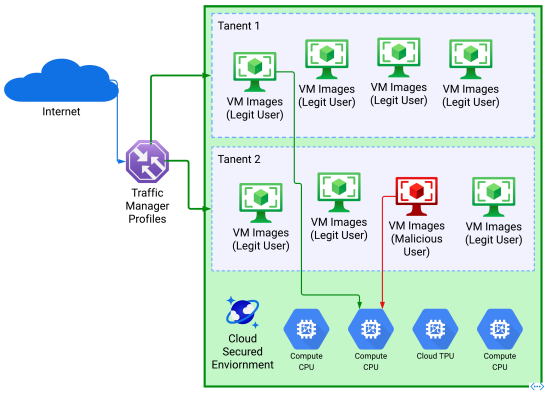

Fig. 1. Architecture of a Covert Channel Communication System

measurements over and over, the attacker can construct a covert binary message composed of 1s and 0s [2]. tim) and receiver (attacker) are in separate VMs. .Both VMs are pinned to the same CPU core, which makes the timing changes easier to notice [8]. The attacker watches how fast or slow the CPU works and rebuilds the hidden message. his method works on different platforms like Xen, KVM/QEMU, and Docker containers. These tests showed that this method can work with over 90. While this setup is cost-effective, it introduces hidden risks. This setup is cheap and easy, but it is risky. For example, if one VM uses the CPU a lot, it slows down the others. An attacker can watch this slowdown and steal information using timing tricks. [9].

D. Improving Reliability

To enhance the reliability of covert communication, certain methods are used. One such method involves the use of spread-spectrum techniques like Code Division Multiple Access (CDMA), which helps to reduce the effect of unwanted signals or noise. Additionally, attackers often take multiple measurements and compute an average, which improves the accuracy of the inferred information and reduces the impact of random fluctuations in CPU performance.

E. Transmission and Encoding

We now explain how such covert channel attacks are structured. In this architecture, two virtual machines (VMs) interact secretly. The first is the attacker VM, which operates with low privileges. This VM continuously monitors CPU performance metrics, trying to detect timing variations that may indirectly expose activities on the victim's side. The second is the victim VM, which has higher privileges and performs sensitive operations such as encryption algorithms. These operations impact the CPU load in a way that can be observed by the attacker, enabling the covert transmission of information.

F. Real-World Usage

Cybercriminals and malware developers often use covert channels for various malicious purposes. These include sending instructions to malware, which is commonly referred to as Command and Control (C2), stealing private or confidential information, and bypassing firewalls or other security mechanisms without detection.

A well-known example is the **Mirai Botnet**, which exploited poorly secured Internet of Things (IoT) devices to launch large-scale Distributed Denial of Service (DDoS) attacks.

G. Countermeasures

To defend against covert channels, several countermeasures can be implemented. One method is the use of traffic normalization, which involves modifying packet headers to a standardized format. This makes it easier to detect and eliminate any hidden messages embedded within network traffic. Another strategy is the application of machine learning techniques to identify unusual patterns or behaviors in the traffic that may indicate secret communication.

Additionally, network pumps and active monitors can be deployed to observe and control network activity. These tools help prevent covert channels from being established or used. However, these countermeasures are still evolving and require further research and development to become highly effective [10].

H. Distributed Covert Channels (DCC)

Distributed Covert Channels (DCCs) are smart and hidden ways to send secret information. They are hard to find because:

These covert channels often exhibit complex behaviors that make them harder to detect and stop. They may use multiple data paths or communication flows to transmit information. Over time, they can also adapt and change their operational patterns. Additionally, such covert activities are frequently distributed across various devices or networks, increasing their stealth and resilience [11].

IV. Conclusions

Covert channels are hidden methods used to steal or send secret messages without being detected. They operate within regular computer traffic, making it difficult for standard security tools to spot them. As the number of connected devices, such as those in the Internet of Things (IoT), increases, better detection methods are needed. The future of detecting covert channels should focus on leveraging advanced technologies, like machine learning and AI, to identify these attacks and protect systems with fewer resources.

Impact and Detection Challenges

Covert channels can cause significant security issues. They can be used to steal sensitive information, such as passwords, encryption keys, or private tokens. One of the major challenges is that these attacks do not leave obvious traces in the network, making them difficult to detect. Traditional security tools, like firewalls and antivirus programs, often cannot identify these hidden threats. These attacks operate by manipulating the way the CPU functions, but in a manner that appears normal to the system. As a result, the system does not recognize anything unusual, and no warnings are logged, making it harder to catch the attack.

How to Defend Against Covert Channels

There are several strategies to prevent covert channel attacks. One approach is to reduce the shared usage of CPU resources. This can be achieved by turning off features like Simultaneous Multithreading (SMT), which prevents programs from sharing CPU cores. Sensitive tasks can also be assigned to specific CPU cores, ensuring that they do not share resources with potentially untrusted tasks. Tools such as Linux cgroups can help isolate tasks and prevent interference.

Another defense is to disrupt timing-based attacks. This can be done by introducing random delays or noise, which interferes with the attacker's ability to measure precise timing. Virtual machines (VMs) can also be moved around regularly to disrupt the timing patterns that attackers rely on. Keeping track of the location of VMs and the distribution of resources helps identify patterns that might be exploited in attacks.

Finally, specialized tools can be used to detect unusual activity. Intel Performance Monitoring Units (PMUs) can be employed to identify strange behavior in CPU performance. Monitoring CPU usage and thread handling for abnormal patterns can provide early warning signs. Additionally, advanced technologies such as machine learning can be used to detect attempts at secret communication by recognizing patterns in the system's behavior.

References

[1] A. Singh and A. Kumar, "Eraser: An exploit-specific monitor to prevent malicious communication channel," Georgia Institute of Technology, Tech. Rep. GIT-CERCS-04-28, 2004.

[2] K. Okamura and Y. Oyama, "Load-based covert channels between xen virtual machines," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 173–180.

[3] H. Okhravi, S. Bak, and S. T. King, "Design, implementation and evaluation of covert channel attacks," in *2010 IEEE International Conference on Technologies for Homeland Security (HST)*. IEEE, 2010, pp. 481–487.

[4] S. Wendzel, T. Schmidbauer, S. Zillien, and J. Keller, "Dyst (did you see that?): An amplified covert channel that points to previously seen data," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[5] C. Heinz, M. Zuppelli, and L. Caviglione, "Covert channels in transport layer security: Performance and security assessment." *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 12, no. 4, pp. 22–36, 2021.

[6] N. Matyunin, N. A. Anagnostopoulos, S. Boukoros, M. Heinrich, A. Schaller, M. Kolinichenko, and S. Katzenbeisser, "Tracking private browsing sessions using cpu-based covert channels," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2018, pp. 63–74.

[7] S.-P. W. Shield and V. D. Gligor, "Auditing the use of covert storage channels in secure systems," in *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Computer Society, 1990, pp. 285–285.

[8] S. Wendzel and W. Mazurczyk, "Poster: An educational network protocol for covert channel analysis using patterns," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1739–1741.

[9] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 44–57, 2007.

[10] J. Tian, H. Ma, D. Gao, and X. Kuang, "Cachealarm: Monitoring sensitive behaviors of android apps using cache side channel," *IEEE Transactions on Dependable and Secure Computing*, 2025.

[11] M. A. Khadse and D. M. Dakhane, "A review on network covert channel construction and attack detection," *Concurrency and Computation: Practice and Experience*, vol. 37, no. 1, p. e8316, 2025.