Dive into the Cloud: Unveiling the (Ab)Usage of Serverless Cloud Function in the Wild

Yijing Liu*
Tsinghua University
Beijing, Beijing, China
liu-yj23@mails.tsinghua.edu.cn

Mingxuan Liu*
Zhongguancun Laboratory
Beijing, Beijing, China
liumx@mail.zgclab.edu.cn

Yiming Zhang
Tsinghua University
Beijing, Beijing, China
zhangyiming@tsinghua.edu.cn

Baojun Liu
Tsinghua University
Beijing, Beijing, China
lbj@tsinghua.edu.cn

Jia Zhang†
Tsinghua University
Beijing, Beijing, China
zhangjia2017@tsinghua.edu.cn

Geng Hong
Fudan University
Shanghai, Shanghai, China
ghong@fudan.edu.cn

Haixin Duan‡
Tsinghua University
Beijing, Beijing, China
duanhx@tsinghua.edu.cn

Min Yang
Fudan University
Shanghai, Shanghai, China
m_yang@fudan.edu.cn

Abstract

Serverless cloud functions transfer server management responsibilities to service providers, offering scalability and cost-efficiency. This convenience not only facilitates normal activities but also raises abuse concerns. So far, public understanding of real-world cloud functions remains limited. To fill this gap, we conducted an in-depth measurement study to uncover their practical usage and abuse. Through empirical analysis of nine leading providers (e.g., AWS, Tencent), we identified 531,089 function domains from a passive DNS dataset spanning April 2022 to March 2024. We first investigated the usage status of serverless cloud functions, showing the different practices between providers. Additionally, based on active requests to these functions, we pointed out privacy risks of unauthorized access and identified four abuse types, including covert C2 communication, hosting malicious websites, promoting illicit services, and abusing egress nodes as IP proxies. Alarmingly, 4.89% of cloud functions are being abused, with over 614k invocations recorded. Only four abused functions were flagged by existing threat intelligence systems, indicating critical gaps in security monitoring for serverless environments. Our work offers insights into the serverless cloud ecosystem and provides recommendations for better management. With responsible disclosure, we hope to raise awareness and improve protective measures against abuses among cloud function providers.

CCS Concepts

• **Networks** → **Cloud computing**; **Network measurement**; • **Security and privacy** → *Distributed systems security*.

Keywords

Serverless Function; Function Abuse; Usage Measurement

ACM Reference Format:
Yijing Liu, Mingxuan Liu, Yiming Zhang, Baojun Liu, Jia Zhang, Geng Hong, Haixin Duan, and Min Yang. 2025. Dive into the Cloud: Unveiling the (Ab)Usage of Serverless Cloud Function in the Wild. In *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25), October 28–31, 2025, Madison, WI, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3730567.3732915

1 Introduction

Serverless cloud function[1] represents an emerging cloud service model that allows users to execute code without the need to manage servers. Operating within the Function as a Service (FaaS) model, serverless functions are different from traditional service models like SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service). Their cost-effectiveness and ease of use make them especially attractive to developers [24]. To meet the growing demand, major cloud providers have introduced serverless function services such as AWS Lambda (2014) [8], Azure Functions (2016) [41], Google Cloud Functions (2017) [34], etc. The global serverless function market was valued at approximately $7.5 billion by 2022 [69], highlighting its growing significance within cloud infrastructure services.

However, serverless cloud functions not only attract benign usage [80] but also facilitate malicious activities. Similar to the abuse in other cloud infrastructures [35, 55, 56], adversaries hide malicious content behind the infrastructure of cloud services, such as domains, TLS certificates, and IP addresses. This is particularly concerning, as the infrastructures of cloud providers are frequently

*Both authors contributed equally to this research.
†Also with Quancheng Laboratory.
‡Also with QI-ANXIN Technology Research Institute.

[1]In our paper, the terms "serverless function", "cloud function", and "function" are used interchangeably and all refer "serverless cloud function".

IMC '25, Madison, WI, USA

ACM ISBN 979-8-4007-1860-1/2025/10
https://doi.org/10.1145/3730567.3732915

whitelisted by web application firewalls (WAFs) [86]. Unfortunately, the convenience and low cost of cloud functions exacerbate cloud resource abuse. Several reports [6, 70] indicate that serverless functions have been used to hide C2 (command and control) beacons, highlighting the risk of cloud functions being exploited in the real world. Given the growing market for cloud function services, understanding their extent of abuse and role in supporting malicious activities is crucial.

Existing studies [37, 46, 62, 74, 81] mainly focus on measuring the performance of serverless computing, relying on internal data center records [46] or a limited set of actively registered functions [81]. Additionally, most research centers on AWS or a few leading providers, resulting in a lack of comprehensive insights into the broader landscape. Therefore, there is no systematic research analyzing the potential misuse of serverless functions, and our understanding of their real-world utilization is limited. Fundamental questions about the number of registered cloud functions and usage patterns across different providers remain unanswered, let alone evaluations of their misuse potential. Since serverless functions operate on-demand and resources are activated only upon invocation [21], it is challenging to externally identify which functions are actively providing services.

Our work. Given that serverless functions primarily provide services through HTTP(s), passive DNS (PDNS) offers us a systematic perspective to examine the (ab)usage of serverless functions. Collaborating with a major Chinese DNS service provider, we leveraged their extensive PDNS data to collect serverless function domains. Specifically, to address the challenge arising from the black-box implementations, we first conducted empirical analysis on nine leading cloud providers offering serverless functions (e.g., AWS, Google, Tencent) [68, 77, 78]. By actively creating serverless functions and reviewing the development documentation, we validated the function URLs' formats of each provider. Filtered by the format, we ultimately identified 531,089 serverless function domain names in the PDNS data collected from April 2022 to March 2024. Based on the identified functions, we conducted a comprehensive investigation into the usage patterns and misuse potential of serverless functions across multiple providers to bridge these research gaps.

Main findings. With identified serverless functions, we first analyzed their usage patterns based on resolution records. Our analysis indicates that cloud functions are widely adopted, with over 1.55 billion invocations. Their active time is short and concentrated, suggesting they are primarily used for stateless, short-term tasks. To understand the current abuse status, we conducted controlled active probing of these domain names and recorded the returned results. We primarily focus on identifying instances of cloud function abuse involving illegal, malicious, or policy-violating activities. We identified four major abuse types, including covert C2 communication, hosting malicious websites, hiding illicit services, and using egress nodes as IP proxies. These abuses leveraged the scalability and flexibility of cloud functions to support low-cost and evasive operations. They manifested in eight concrete cases such as hosting gambling platforms, reselling OpenAI API keys, and providing proxy services to bypass geographic restrictions. These abuses affected 594 cloud function domains across major cloud providers like Tencent, AWS, Google, and Aliyun, with over 614k invocation requests, highlighting a real-world threat.

Furthermore, our work shows that serverless cloud functions are increasingly targeted by miscreants as a new cloud infrastructure, yet defenses against such misuse have not kept pace. Only 0.67% of the abused cloud functions were marked as malicious by VirusTotal [38], highlighting poor awareness among cloud providers and threat intelligence regarding serverless abuse. Moreover, our findings reveal privacy leakage risks from unauthorized access on cloud functions, identifying 394 instances of exposed sensitive data, including access tokens, passwords, and Personally Identifiable Information (PII). All identified abuses were responsibly reported to the affected vendors, with positive feedback from Tencent and AWS. Finally, based on our findings, we provided three recommendations for providers to better manage their cloud function services.

Contributions. Our contributions are as follows:

- *Real-world Usage Analysis:* Through the passive DNS dataset, we built the first systematical dataset of serverless functions in the wild based on domain formats, covering nine major providers over a span of two years. Our analysis confirmed the widespread adoption of serverless functions and investigated implementation differences across providers, especially ingress configurations.
- *Comprehensive Abuse Review:* Through active content analysis, we uncovered four abuse scenarios of cloud functions, highlighting their exploitation in the wild and their emerging role as infrastructure for the underground industry, which remains little recognized by existing threat intelligence.

2 Background

Serverless cloud functions allow users to run code without managing servers, offering scalability and flexibility. To provide a clear understanding of how these functions work, we describe their full lifecycle in three stages: deployment, invocation, and execution, as shown in Figure 1.

2.1 Deployment

As the initial stage in the cloud function lifecycle, the deployment process (Step (a)) involves creating and configuring the function. During this phase, developers can deploy their code to the cloud and define parameters such as environment variables, memory allocation, execution timeouts, and concurrency limits. There are three primary ways to create cloud functions. The first is via the web console, which allows code uploads in ".zip" format and includes online editing. For complex environments, functions can also be created from container images, providing flexibility in defining the environment. Another method is through the command-line interface (CLI), useful for automated deployments in CI/CD pipelines. Lastly, some providers like Tencent and Google integrate with IDEs such as Visual Studio Code, enabling developers to manage functions directly within their development environment. Each method offers distinct advantages for deploying serverless functions.

2.2 Invocation

Once deployed, the functions transition into the invocation phase (Step (b)). Users, not just function developers, can invoke the functions at any time in response to specific events or triggers. Generally, most cloud providers allow users to access that cloud function directly with an HTTP request. One common invocation method is

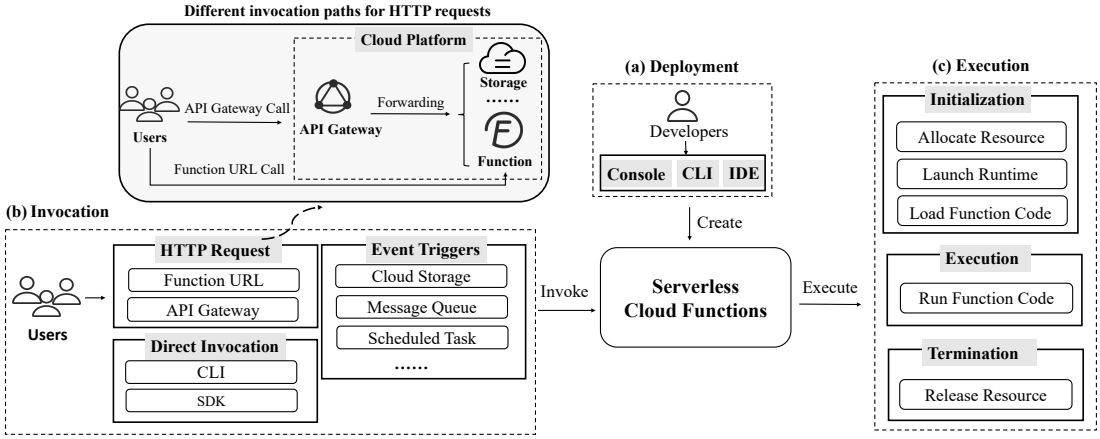

Figure 1: Workflow of using serverless cloud functions.

through a function URL, which creates a dedicated URL for each function, allowing users to invoke it via standard HTTP requests, such as GET or POST. Another HTTP invocation method uses API gateway, which binds functions as the backend and routes incoming HTTP requests through a generated REST API. Both methods facilitate the invocation of serverless functions, as illustrated in Figure 1. API Gateway incurs additional costs but offers advanced features such as caching, rate limiting, and custom authentication, making it suitable for applications that need robust API management. In contrast, function URLs provide a more lightweight HTTP endpoint with Identity and Access Management (IAM) authentication, which is ideal for quick invocations in smaller-scale applications.

In addition to HTTP(S) requests, serverless functions can be automatically triggered by specific events. Different cloud providers offer varying types of event triggers. Common triggers include file uploads to cloud storage, message queues (such as AWS SQS [4] or Google Pub/Sub [32]), and scheduled tasks. Besides these preset triggers, serverless functions can also be manually invoked through the web console or CLI for testing purposes. Furthermore, developers have the option to programmatically invoke serverless cloud functions from other cloud services or applications via Software Development Kits (SDKs). Since event-triggered functions lack exposed endpoints and cannot be directly invoked externally, our study focuses on those with HTTP(S) endpoints.

2.3 Execution

When an invocation occurs, the function runs on the cloud without user intervention (Step (c)). To facilitate this, the serverless provider dynamically allocates the necessary resources and launches an execution runtime environment within isolated instances, such as virtual machines or containers. The relevant function code and any required dependencies are then loaded. Subsequently, the function runs with the specified input parameters to perform the designated operations. After execution, the environment may be released of if no further invocations are pending, ensuring optimal resource utilization. Due to this working model, when a function is invoked after a period of inactivity, it undergoes a "cold start". This process can introduce latency, as it involves the above whole initialization phase. In contrast, if a function is invoked while an execution environment is still active, it performs a quicker "warm start", since the resources are already allocated and ready for processing.

Price Model. Notably, serverless functions offer developers a scalable, flexible, and cost-effective solution, allowing them to pay only for actual executions. Typically, providers charge based on the number of invocations and the computation costs during execution. The computation cost is typically calculated as the product of resource usage (in GB) and execution time (in seconds), measured in GB-seconds (GB-s). Potential resources include GPU usage, CPU usage, memory usage, and disk usage. Actually, serverless function pricing is relatively low, and most providers offer free trials. For example, AWS offers a free tier of 1 million requests and 400k GB-s resources per month, with additional requests charged at $0.20 per million and $0.0000166667 per GB-s. Similarly, Tencent offers a free three-month trial for new users. This flexible pricing model makes serverless functions appealing for a broad range of applications.

3 Methodology

To comprehensively reveal the (ab)use status of serverless functions, the initial step is to gather a substantial dataset of cloud functions in the wild. As shown in Figure 2, we began by empirically analyzing nine leading serverless function providers and extracted the URL formats of functions. Using these formats, we extracted relevant domain names from the passive DNS (PDNS) dataset, which captures historical DNS queries and responses observed at recursive resolvers. This data enabled analysis of serverless function usage patterns through DNS resolution records. We then used active probing to collect response content and assess potential abuse.

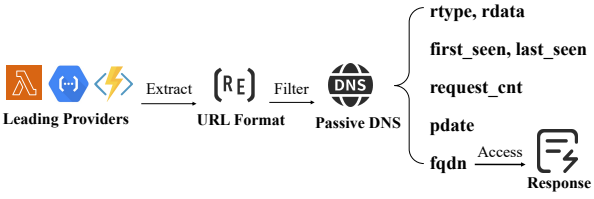

Figure 2: Overview of the data collection process.

3.1 Function URL Format Definition

As mentioned in Section 2, both function URLs and API Gateway could generate URLs as an HTTP(s) endpoint to expose serverless functions. However, due to the limitations of the API gateway (see Section 3.5), we only analyzed function URLs to ensure an accurate baseline measurement of the serverless function ecosystem.

Due to the lack of a unified standard for function URLs, implementations of each vendor are heterogeneous. Therefore, we conducted an empirical study to examine their practical designs, focusing on URL structure. Based on market share statistics [68, 77, 78], we selected nine major serverless function providers for our study, including AWS Lambda [8] and Aliyun Function Compute [2], as detailed in Table 1 ("Providers"). Notably, with the release of Google Cloud Functions (2nd gen) [19], Google now has two function URL formats, referred to as "Google" and "Google2". Specifically, to examine the practical implementations, we created two serverless functions on each provider and reviewed their development documentation. For Tencent, Aliyun, Azure, Oracle, and IBM, function URLs are automatically generated upon creation. In contrast, AWS, Kingsoft, and Google allow users to enable function URL invocation during setup, while Baidu requires users to create a separate HTTP trigger after creation and allows for a customizable path.

Based on our observations, we implemented provider-specific URL formats to identify cloud functions, as shown in Table 1. Despite differences across providers, most URLs follow the pattern $< \mathit{USER\text{-}Prefix} > . < \mathit{Domain\text{-}Suffix} > /< \mathit{Path} >$, where the $\mathit{Domain\text{-}Suffix}$ identifies the provider (e.g., *scf.tencentcs.com* for Tencent). The $\mathit{USER\text{-}Prefix}$ typically serves as an identifier for specific functions and may include user-defined components, such as function names ($[FName]$) and project names ($[PName]$), as exemplified by Aliyun. Alternatively, it can contain user-specific information, such as a user ID ($[UserID]$) or a randomly generated string ($[Random]$), as seen in AWS. Some providers also incorporate regional information ($[Region]$) in the $USER-Prefix$. Additionally, the $[FName]$, which can directly identify a cloud function, may also appear in the $Path$, as observed with providers like Google, Azure, IBM, and Oracle.

3.2 Serverless Function Identification

Based on the function URL formats, we identified functions used in real-world scenarios. We collaborated with 114 DNS, one of the leading DNS providers in China, to obtain its PDNS dataset collected from its extensive recursive resolvers, which handle about 600 billion DNS queries daily. While our dataset was obtained through research collaboration, similar PDNS datasets (e.g., Farsight DNSDB [66] and Umbrella [17]) are also available for others via paid services. In our study, considering the rise of serverless functions, we requested data from April 2022 to March 2024.

In passive DNS datasets, each record is a tuple, like <*fqdn* (fully qualified domain name), *type*, *rdata*, *first_seen*, *last_s een*, *request_cnt*, and *pdate*>, indicating that on date *pdate*, the domain *fqdn* was resolved to *rdata* with type *rtype*, and observed *request_cnt* times by recursive DNS resolvers. Each record is aggregated at the daily level and includes the first and last resolution timestamps on that date. In our study, we use the *request_cnt* field as an indicator of function invocations. Due to caching at local resolvers [47], this count represents a conservative lower bound on actual usage. For targeted analysis, we further aggregated records for each provider based on "fqdn". This aggregation allowed us to calculate key metrics for each cloud function, including its first and last observed times ("first_seen_all" and "last_seen_all"), the number of days it was invoked ("days_count"), cumulative invocation counts ("total_request_cnt") and the distribution of its resolution results.

Given the massive data size of the PDNS dataset, we converted the URL format into regular expressions and selected records where the "fqdn" matched these patterns. As shown in Table 1 ("Domain Regular Expression"), these expressions were constructed based on the defined elements from the URL format, such as the domain suffix, delimiters, and the length of random strings. To validate these patterns, we initially tested them on one day's data and refined the expressions until only valid cloud function domains were collected. We then expanded our data collection to identify DNS records related to serverless functions. Although we could not directly identify specific cloud functions for providers that embed function identifiers in the path (highlighted in blue), the data could still reflect the usage patterns of these providers' cloud functions. Unfortunately, Azure shares the domain suffix ("azurewebsites.net") with other web applications, making it difficult to distinguish serverless functions based on domain patterns alone. As a result, we excluded Azure (shown in gray) from our data collection.

3.3 Active Information Collection

Passive DNS data can assist in understanding the scale and trends of cloud function usage, but the specific usage purpose of cloud function remains unclear, especially in abuse scenarios. Due to our inability to access the original code of the cloud functions, we could not directly ascertain their intended usage. As an alternative, we attempted to actively access these functions and collect the responses they return, to laterally understand their usage scenarios and behaviors.

Active collection scope. As mentioned earlier, Google, IBM, Oracle and Azure's functions need to be accessed by specifying a specific path, which is not visible and available to us (highlighted in blue in Table 1). Consequently, our active data collection focused on the other five providers, including AWS, Google2, Tencent, Baidu, Aliyun and Kingsoft.

Active collection method. To infer the intended usage of serverless functions, we issued HTTP(S) requests to the collected function domains and recorded their responses. Given the two-year span of our dataset, some functions may behave differently now due to developers' initiative adjustments. As historical behaviors cannot be reconstructed, we treat the current response as the function content. To ensure ethical compliance, we limited access attempts and used parameter-free GET requests, minimizing interference with functions. We used the Python requests package to access each endpoint via HTTPS, falling back to HTTP on failure. Domains that failed both attempts were marked as unreachable. A uniform timeout of 60 seconds was applied, following the default settings of most providers [30, 39].

Table 1: The URL formats and domain regular expressions of serverless functions across providers.

Providers	Launch Year	USER-Prefix	Domain-Suffix	Path
		Domain Regular Expression		Generation Mode
Aliyun Function Compute [2]	2017	[FName][1]-[PName][2]-[Random].[Region]	fcapp.run	/
		^(.*)-(.*)-[a-z]{10}\.(.*)\.fcapp\.run$		Automatic
Baidu Cloud Function Compute [11]	2017	[Random].cfc-execute.[Region]	baidubce.com	/
		^[a-z0-9]{13}\.cfc-execute\.(.*)\.baidubce\.com$		Manual
Tencent Serverless Cloud Function [76]	2017	[UserID]-[Random]-[Region]	scf.tencentcs.com	/
		^[0-9]{10}-[a-z0-9]{10}-(.*)\.scf\.tencentcs\.com$		Automatic
Kingsoft Cloud Function [51]	2022	[Random].[Region]	ksyuncf.com	/
		^(.*)-(eu-east-1|cn-beijing-6).ksyuncf.com$		Optional
AWS Lambda [8]	2014	[Random].lambda-url.[Region]	on.aws	/
		^(.*)\.lambda-url\.(.*)\.on\.aws$		Optional
Google Cloud Function [34]	2017	[Region]-[PName]	cloudfunctions.net	[FName]
		^(asia|europe|us|australia|northamerica|southamerica)-(.*)-(.*)\.cloudfunctions.net$		Optional
Google Cloud Function (2nd gen)	2022	[FName]-[Random]-[Region]	a.run.app	/
		^(.*)-[a-z0-9]{10}-(.*)\.a\.run\.app$		Optional
IBM Cloud Function [42]	2016	[Region]	functions.appdomain.cloud	.../[FName]
		^(us-south|us-east|eu-gb|eu-de|jp-tok|au-syd)\.functions\.appdomain\.cloud$		Automatic
Oracle Cloud Functions [65]	2019	[Random].[Region]	oci.oraclecloud.com	.../[FName]
		^[a-z0-9]{11}\.(.*)\.functions\.oci\.oraclecloud\.com$		Automatic
Azure Function [10]	2016	[PName]	azurewebsites.net	.../[FName]?code=Key
		^(.*)\.azurewebsites\.net$		Automatic

[1] FName is short for Function Name.
[2] PName is short for Project Name.
[*] Due to PDNS's lack of URL path observation, we excluded gray-marked providers from data collection and blue-marked providers from active access.

3.4 Active Data Analysis

The goal of collecting cloud function responses is to infer their intended usage from an external perspective, particularly to identify potential misuse. In our study, we focus on abuse scenarios where adversaries deploy serverless functions on public cloud platforms to support malicious, illegal, or policy-violating activities (i.e., violating cloud provider terms of service). Attacks that exploit cloud platform vulnerabilities are beyond the scope of this study.

Identifying such misuse and extracting meaningful insights from large-scale data without prior knowledge is challenging. To address this, we first categorized the content into four types based on textual patterns: JSON, HTML, Plaintext, and Others. These types provide rough clues about function purposes. JSON often indicates API responses, HTML suggests webpage generation, and Plaintext may contain logs or textual output. To reduce manual effort, we applied hierarchical clustering within each content type to group similar responses. Each response was converted into a TF-IDF vector, and pairwise similarity was measured using cosine distance. We then used agglomerative clustering with average linkage [63], setting a 90% similarity threshold (cosine distance < 0.1) based on manual inspection. This effectively grouped near-duplicates while preserving meaningful differences, resulting in 4,512 clusters and significantly reducing manual workload. Subsequently, two security experts (one with 5–10 years of experience and another with 3–5 years) were involved. Their task was to identify cloud functions containing suspicious content or related to illicit activities (e.g., gambling, fraud, etc.). Following the independent reviews, a collaborative discussion was held to resolve disagreements and finalized confirmed misuse cases. Then, for each abuse type, we summarized the characteristic patterns to further identify more instances within our active access data. By combining the resolution records from the PDNS dataset, we confirmed the real-world impact of these abuses.

Exclusion of sensitive data. Considering that cloud functions may handle sensitive information like other cloud services [13], we proactively mitigated ethical risks before analysis. To this end, we initially used EarlyBird [28], a tool designed for detecting sensitive data within source code repositories, to scan for exposed personal or sensitive information in function content. Any identified sensitive data were anonymized using MD5 hashing, with a salt string of 10 random characters, which is detailed in Appendix A. Functions without sensitive data were treated as publicly accessible and proceeded to analyze their returned content.

3.5 Limitation

Using this data collection process, we constructed the currently known unique dataset targeting the cloud function domain names in the wild. However, we must acknowledge that our work still has certain limitations.

• **Selection of service providers.** In this work, our data collection focused on nine major cloud function providers. While we overlooked some other providers, the chosen nine maintain the largest market share in cloud functions [68, 77, 78], indicating they have the most substantial user bases and are therefore the most representative. Thus, we believe these providers reflect a diverse array of both Chinese and internationally recognized serverless products, offering a sufficiently broad view of the ecosystem.
• **Exclusion of API Gateway.** API Gateways are often used to trigger serverless functions. However, as independent products, they are not limited to serverless functions and could connect to diverse backend services. It is difficult to reliably identify whether the backend of an API gateway is a serverless function. Therefore, to avoid introducing false positives (i.e., other cloud services) into our observations of serverless functions, we excluded API Gateway from our study on the current (ab)usage of serverless functions.
• **Regional biases of passive DNS.** As with any passive DNS dataset, our analysis is constrained by the geographic location of the recursive resolvers [85]. The PDNS data we used primarily serves users in China, which may bias observations toward local usage patterns and affect certain trends or scale estimates in Section 4.1. However, since the cloud providers we analyzed operate globally, the observed practices and abuse patterns are not limited to this regional scope, and our findings still offer new insights into the (ab)use of cloud functions from a broader perspective.
• **Restricted active data.** Due to the limitations of the PDNS dataset, we can only observe the domains of serverless functions. As a result, some functions were unable to be invoked via our HTTP(S) requests because of missing paths or query parameters, reducing the size of the analyzable response set. Additionally, access control and function availability (affected by the two-year data collection window) further limited valid responses and complicated intent inference. Nonetheless, our non-intrusive probing approach offers a practical and ethical way to assess function (ab)usage at scale, and the breadth of the dataset supports preliminary yet meaningful insights. This remains the only viable way to infer function behavior from an external vantage point.

4 Overall Usage Status

In this section, we depict the usage status of serverless cloud functions through statistical analysis, including the trends in usage, practical ingress infrastructure, invocation pattern, and the current invocation status of these functions.

4.1 Evolving Trend of Serverless Function

Finding 1: Serverless cloud functions are widespread and evolve as the market grows. Providers' updates on service features and policies could directly impact their usage.

Adoption scale. Based on our identification method, we filtered a total of 536,181 serverless functions from nine providers in the PDNS dataset spanning from April 2022 to March 2024. From a temporal perspective, the overall usage of cloud functions exhibits a growth trend as shown in Figure 3. Notably, in April 2022, the introduction of function URLs in AWS Lambda [5] led to a substantial increase in newly observed functions. As introduced in Section 2.2, invoking cloud functions through the HTTP(S) endpoints begins with a DNS resolution query for the target function domain. Accordingly, we approximated the invocation number of cloud functions by counting the observed DNS resolution queries in the PDNS. Over the two years of data collection, the total invocation count for these 536k functions exceeded 1.55 billion times, indicating a significant adoption scale.

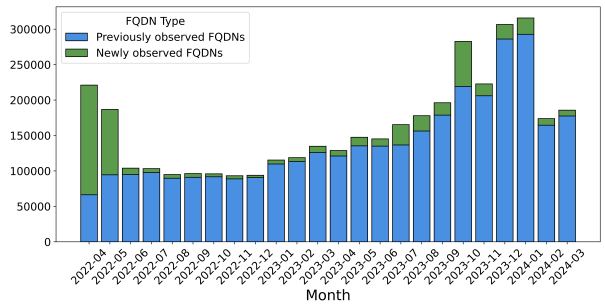

Figure 3: Monthly statistics of cloud function counts. ("Newly observed FQDNs" were calculated as the daily additions compared to the previous day. The figure shows their monthly cumulative totals.)

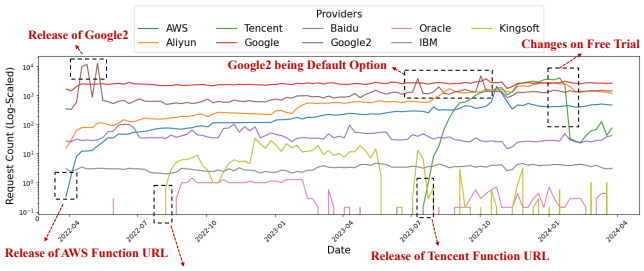

Figure 4: Invocation trends of different cloud functions.

Evolving trend. The evolution of serverless cloud functions is driven by market development. As shown in Figure 4, usage across most providers remains stable, with Google and Aliyun leading. Due to the regional limitations of PDNS, we refrain from estimating the market share of providers. Meanwhile, our data indicates the emergence of new players in the market. Kingsoft and Tencent began to appear with cloud function resolutions in August 2022 and August 2023, respectively, aligning closely with their official announcements regarding the release of function URL [31, 43]. Moreover, user adoption of cloud functions is heavily influenced by providers' strategies, with adjustments in these strategies quickly reflected in their invocation patterns. The sharp decline of Tencent in January 2024 was likely due to the alteration of its free trial quota distribution model [1]. Additionally, Google2 was released in February 2022 [19], just two months prior to our measurement period, which led to a slight spike in usage. Another significant increase was recorded in August 2023, coinciding with Google's announcement that Google2 had become the default option in the Google Cloud Console user interface [18]. In contrast, despite IBM's December 2023 announcement to deprecate its cloud functions

service [27], no significant decline was observed by March 2024, possibly due to IBM's continued support for existing functions until October 2024.

4.2 Practical Ingress Infrastructure

To enable dynamic resource allocation, providers use ingress nodes in data centers to route user requests. Our DNS resolution log data allows us to systematically reveal the ingress infrastructure configurations of each provider. The overall resolution types for each cloud function are shown in Table 2 ("Total"). When successfully resolved, cloud functions may correspond to an IP address (A or AAAA records) or a CNAME domain (CNAME records), both referred to as ingress nodes. While IPv6 adoption is increasing [23], our results show IPv4 is still the preferred method, with AAAA records (17.54%) significantly lower than A records (61.00%). AAAA records were only observed for AWS, Google, and IBM. Additionally, providers like Aliyun, Tencent, Baidu, and IBM favor load-balancing DNS with CNAMEs in over 70% of responses, whereas Kingsoft, AWS, Google, and Oracle always return direct A or AAAA records.

Finding 2: Ingress nodes exhibit a distinct geographical concentration. Functions within the same region are typically assigned to the same limited set of ingress nodes.

Regional service providing. Serverless cloud functions typically adopt a region-based architecture to enhance performance and reduce latency. A "region" refers to a geographic area served by one or more data centers, with providers allowing users to choose regions closer to their target audience. As detailed in Section 3.1, most providers embed region identifiers in function domain names, such as "cn-shanghai" (Shanghai, China) and "eu-west-1" (London, UK) in Aliyun functions. We extracted region information from these domain names and analyzed it alongside DNS resolution results. As calculated in Table 2 (''Regions"), the supported regions of providers vary significantly. For example, functions from Baidu are concentrated in three cities in China (i.e., Beijing, Shenzhen, and Suzhou), while global providers like AWS cover nearly all regions except Antarctica. Moreover, serverless functions within the same region are resolved to the same set of ingress nodes located in that area. Specifically, if a CNAME domain is resolved, it includes geographic parts that match the cloud function. For example, "gz" is the geographic part of one CNAME result from Tencent ("gz.scf.tencentcs.com"). Likewise, the location of resolved IP addresses also align with the cloud function's region.

We also counted the number of observed ingress nodes in each region. Most providers exhibited a concentrated pattern, with 1 to 3 fixed IP addresses or CNAME domains configured to handle all function requests for a region. The top 10 resolution results accounted for nearly all requests for each type, as shown in Table 2 ("Top10"), highlighting this characteristic. In contrast, AWS displays a more dispersed resolution pattern, featuring a greater number of unique results. Specifically, AWS has 2,082 IPv4 and 2,579 IPv6 ingress nodes in ap-northeast-1 (Tokyo), while other popular regions like eu-west-1 (Ireland) and us-east-1 (Virginia) also exceed 1,000 ingress nodes. In stark contrast, Google operates with only one ingress node and resolves all requests to a single IPv4/Ipv6 address regardless of region. Google2 expands this to four nodes but still ignores regional distinctions. Additionally, Google's ingress IP addresses are configured in anycast mode, routing requests to the nearest node, which differs from the region-based approach of other providers.

Finding 3: Ingress nodes for serverless cloud functions exhibits significant reliance on third-party network infrastructure, which may introduce potential security risks.

Reliance on third-party infrastructure. Third-party infrastructure is commonly used in cloud services to quickly and cost-effectively expand their global coverage [88]. We observed such dependencies in cloud functions as well. Generally, the resolved ingress nodes (i.e., IPs or CNAME domains) belong to the corresponding cloud providers, representing the data centers of these providers. However, Baidu and Kingsoft leverage the infrastructure from China's three major telecom operators (i.e., China Telecom [15], China Unicom [16], and China Mobile [14]) as the ingress nodes. Similarly, IBM utilizes Cloudflare [20], a global content delivery network (CDN) provider. While third-party services improve efficiency, improper management of such dependencies can pose significant security risks [45, 60].

4.3 Invocation Pattern of Serverless Task

Finding 4: The active time of cloud functions is short and relatively concentrated, indicating that they are generally employed for stateless short-term tasks.

Invocation frequency. Next, we analyzed the invocation patterns of cloud functions based on the "total_request_cnt" per function. Google, IBM, and Oracle were excluded from this analysis as their domains could not be uniquely associated with specific cloud functions. Regardless of the provider, invocation frequency demonstrated a long-tail distribution as shown in Figure 5. 78.14% of the functions were invoked less than five times, with the peak in the histogram occurring between 3 and 6 invocations, suggesting that most of the functions may have been used for one-off tasks or testing. Only 7.87% were called more than 100 times, which may be used for long-running tasks or in high-demand applications. While invocation frequency can provide some insight into the function's purpose, determining the exact use case based on frequency alone is difficult, as cloud functions are stateless and often follow random invocation patterns.

Lifespan. For the lifespans of the serverless functions, we calculated the time interval between "first_seen_all" and "last_seen_all" to represent the duration of time the serverless function was active. This part of the analysis also needs to exclude Google, IBM, and Oracle. The results show that only 0.34% (14 functions) remained active throughout the entire measurement period (730 days). In contrast, most functions (83.94%), appeared for fewer than five days, with 81.30% active for only a single day. This distribution aligns closely with invocation frequency, indicating that the majority of functions are likely intended for ad-hoc use, rather than for sustained or repeated tasks. The average lifespan was found to be 21.44 days, further underscoring the transient nature.

Table 2: The usage and resolution results of cloud functions across providers.

Providers	Domains	All Request	Regions	rtype=1 (A)			rtype=5 (CNAME)			rtype=28 (AAAA)		
				Total[1]	rdata_cnt[2]	Top10 [3]	Total	rdata_cnt	Top10	Total	rdata_cnt	Top10
Aliyun	59,404	440,860,944	21	27.96%	65	93.57%	72.04%	44	95.54%	0	1	100%
Baidu	753	17,005,075	3	22.47%	10	100%	77.53%	3	100%	0	/	/
Tencent	6,154	3,024,609	22	23.89%	35	95.70%	76.11%	36	92.03%	0	/	/
Ksyun	123	4,044	2	100.00%	4	100%	0	/	/	0	/	/
AWS	19,683	346,651,678	22	76.73%	10914	1.79%	0	/	/	23.27%	17312	2.14%
Google	120,603	543,330,521	37	76.41%	1	100%	0	/	/	23.59%	1	100%
Google2	324,343	199,308,250	37	66.75%	4	100%	0	/	/	33.25%	4	100%
IBM	6	107,421	6	10.15%	6	100%	87.55%	6	100%	2.30%	6	100%
Oracle	14	2,080,577	5	100.00%	31	57.97%	0	/	/	0	/	/

[1] Total represents the proportion of requests for this type.
[2] rdata_cnt indicates the total count of all possible "rdata" for this type.
[3] Top10 shows the percentage of total requests contributed by the top 10 most frequent "rdata".

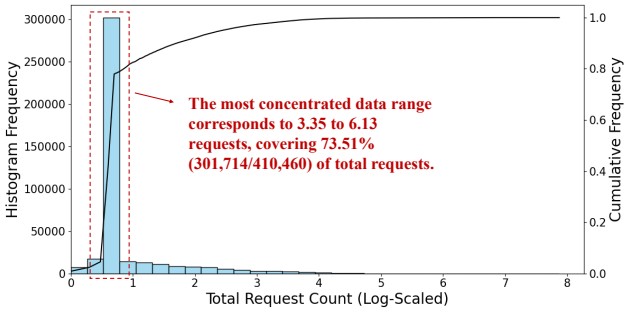

Figure 5: Cumulative and histogram distribution of functions total request counts (The values on the X-axis represent $\log_{10}$(Total Request Count)).

Furthermore, we examined the activity density of these functions by calculating the proportion of days with recorded invocations within their respective lifespans ($p = days\_count/(last\_seen\_all - first\_seen\_all)$). A notable 83.01% of the functions demonstrated continuous activity, with steady invocation (p=1), while the remaining functions showed intermittent usage patterns. In the most extreme cases, there were four functions exhibited lifespans exceeding 603 days (90% of the measurement period) with only one or two invocations. Despite their long lifespans, these functions were rarely used, further reinforcing that most cloud functions are designed for short and stateless tasks.

4.4 Current Invocation Status

Using the active data collection method in Section 3.3, we successfully connected with 410,460 cloud functions, covering major providers such as AWS, Aliyun, Baidu, Tencent, Kingsoft, and Google2. Notably, 2.03% of cloud functions were unreachable, due to network restrictions (internal access only) or timeouts. Additionally, 19.12% (1,597/8,351) of the unreachable functions were caused by DNS resolution failures following function deletions. All of these functions belong to Tencent, which is the only provider without wildcard resolution enabled for its primary domain ("scf.tencentcs.com"), making deleted cloud functions non-resolvable.

Among the reachable functions (402,109), 99.82% supported HTTPS, reflecting a strong emphasis on security. Figure 6 shows the response codes distribution. Notably, As shown in Figure 6, 89.31% of functions returned 404 (Not Found), likely due to our use of default GET requests without parameters. Other causes include missing paths or deleted functions. While most providers return 404 for deleted functions, AWS returns 403 (Forbidden) instead. Server errors, particularly 502 (Bad Gateway), accounted for 2.82%, with AWS exhibiting the highest proportion at 50.56%. These errors may result from unhandled programming exceptions or failures of dependent services [7, 40, 54]. Besides, only 3.14% of cloud functions returned 200 (Success) status codes, indicating successful invocations. Among them, 96.01% (12,138/12,642) were non-empty and were the primary focus of the abuse analysis based on content.

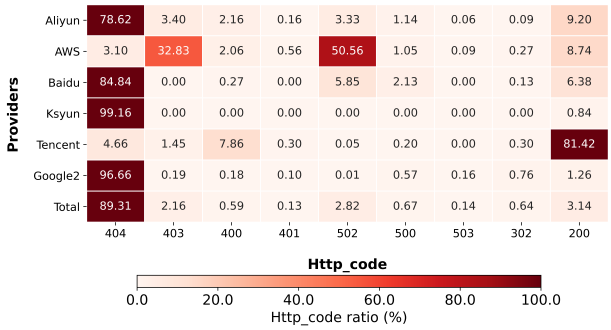

Figure 6: Distribution of top 10 frequent HTTP codes.

5 Abuse Status

In this section, we further analyzed the abuse status of 12,138 serverless functions with non-empty responses. Functions enabling malicious, illegal, or policy-violating activities were classified as abused. To mitigate ethical risks, we followed the method in Section 3.4 and

first anonymized sensitive information prior to large-scale review. Specifically, we identified 394 sensitive data, including 8 phone numbers, 5 national identification numbers, 82 access tokens, 156 API keys, 16 potential passwords, and 127 network identifiers (e.g., IP and MAC addresses). The exposure of tokens or keys is the most common issue, accounting for 60.4%.

Finding 5: Unauthorized access is prevalent in cloud functions and can lead to DoS and DoW threats, as well as significant risks of privacy leakage.

Unauthorized access threat. Such privacy leakages are attributed to unauthorized access, which is a prevalent issue in cloud environments [29, 50, 53]. According to the access status, only 0.13% of the functions returned a 401 (Unauthorized) status, suggesting that access restrictions are largely absent or overlooked. In this threat model, we assume that attackers can easily obtain domain names or URLs of serverless functions from open sources like GitHub or search engines. If these functions lack necessary access controls, attackers can directly send requests using web browsers and any other HTTP clients, leading to potential data leaks. Additionally, due to the billing model, this can result in unexpected charges for developers, known as Denial of Wallet (DoW) [48]. Attackers may also exploit unsecured functions to launch large-scale access requests, causing Denial of Service (DoS) attacks that disrupt normal user access.

After excluding the sensitive data, we started our content review by aggregating the data. Overall, JSON types were the most prevalent at 36.98%, followed by HTML at 31.54% and Plaintext at 30.34%. The "Others" category, comprising 1.15%, included JavaScript, XML, and PHP. Based on our clustering method detailed in Section 3.4, we got 4,512 clusters in total. Our review disclosed four abuse scenarios, including covert C2 communication, hosting malicious websites, hiding illicit services behind serverless functions, and abusing egress nodes as IP proxies.

5.1 Abuse I: Covert C2 Communication

Finding 6: Serverless cloud functions are exploited by malware attackers. We identified 16 active C2 servers on serverless platforms, generating 273,291 requests, highlighting their role as malware infrastructure.

C2 (Command and Control) server acts as a central point to control infected devices (e.g., in a botnet). To evade detection, attackers often use covert C2 channels to hide the communication. Recently, cloud functions have increasingly been abused for this purpose, serving as proxies between victims and the real C2 servers. Their architecture provides anonymity, as outbound traffic appears to come from trusted cloud providers, masking both source and destination. This makes detection more difficult, with malicious traffic routed through legitimate platforms. In practice, attackers embed C2 endpoints directly in the function code (e.g., Algorithm 1), enabling seamless relay without exposing real server IPs.

To identify C2 servers in cloud functions, we relied on communication fingerprints rather than response content. We used a fingerprint database from a well-known security company (QiAnXin [67]), which contains 26 signatures across 18 C2 families and offers broader protocol and port coverage than tools like CyberProbe [64] and C2Miner [25]. The fingerprints were constructed by clustering traffic from 850 C2 malware in a sandbox. Each fingerprint is based on the first full request-response pair after a TCP handshake and captures binary-level communication patterns of a malware family's C2 protocol, such as headers, token sequences, and field delimiters. These patterns can be repurposed as active probes that emulate family-specific C2 requests.

To match fingerprints, we connected to each cloud function domain on ports 80 (HTTP) and 443 (HTTPS), sent probe payloads for different malware families, and collected the responses. By matching the traffic fingerprints from these responses, we identified 16 functions used to conceal C2 communications, specifically linked to the Cobalt Strike and InfoStealer families. The majority of these functions were deployed on Tencent, with a single instance found on Google2. Notably, this method can only identify active C2 servers, limiting the detected number to a lower bound. However, with 273,291 total requests and an average of 112 calls per day to these functions, our findings demonstrate real-world abuse beyond previously reported speculation [6].

5.2 Abuse II: Hosting Malicious Websites

Finding 7: The convenience of cloud functions reduces the cost of deploying malicious websites. We identified 206 malicious websites hosted on cloud functions, covering various types of threats, with over 37k requests.

The minimal server management and pay-as-you-go pricing of cloud functions significantly lower deployment costs, making them attractive to malicious website operators. Additionally, because domain names and infrastructure are hosted by reputable providers, such abuse is harder to trace and block. To investigate this misuse in practice, we applied keyword-based filtering to identify potentially suspicious function responses. This filtering focused on domains commonly host illegal or malicious websites, including gambling, pornography, and cheating, which are prevalent in online abuse and generally exhibit clear semantic signals. Each candidate was then manually reviewed by two analysts, who assessed both page structure and content semantics to determine intent and functionality. Confirmation required clear indicators, such as gambling interfaces, pornographic content, cheating tools, or language and branding directly associated with these areas. Only cases with consistent agreement and clear evidence were labeled.

In total, we identified 206 malicious websites hosted on cloud functions, with a combined 37,774 resolution requests. These sites fall into three main categories. First, based on gambling-related keywords such as "Slot" and "Betting", we confirmed 194 such sites, as illustrated in Figure 8 in Appendix B. These sites are primarily hosted on Google2 and exhibit clear campaign consistency. They have highly similar structures, extensively use "google-site-verification" elements, and employ keyword stuffing techniques for search engine optimization (SEO). Further analysis of the gambling domain resolution records revealed a total of 24,979 calls, indicating widespread access. On average, their functions remained active for 311.39 days, with some lasting up to 544 days, suggesting

they went undetected while continuing to receive traffic. Additionally, using keywords such as "porn", "sex", and "av", we identified six pornographic websites, one adult gaming site, and one Google Maps-based directory of adult stores across Taiwan. The calls to these websites are distributed across 79 days between July 2022 and October 2023, with a total of 854 requests. Furthermore, we observed four cloud functions being utilized to create interactive front-end pages for cheating tools. These included applications for changing email addresses, age modification, and verification generators designed to bypass parental controls for minors, which are commonly used in online games. Although the number of such tools is limited, the passive DNS data reveals significantly higher usage, with a total of 11,941 calls.

5.3 Abuse III: Hidden Illicit Service

Finding 8: Cloud functions are used to disseminate hidden illegal services. They redirect users to illicit websites or contain promotional information in their responses.

Leveraging the ease of development and low-cost advantages as well, some cloud functions are being used to hide illegal services. We identified two methods of covert information dissemination within cloud function responses. One involves redirecting requests to a new link, while the other embeds promotional text with contact information in the response content. These methods guide users to the underlying illegal services. We observed 267 cloud functions involved in this activity, with a total of 123,086 requests.

Redirecting to concealed websites. Based on the active access confirmation, we identified 19 cloud functions that directly returned a redirect link within their response. As some functions contain multiple links, we confirmed 80 unique URLs in total. The majority of them were associated with FXBTG, an online forex and cryptocurrency trading platform that has been blacklisted by the Central Bank of Ireland since October 2018 [82]. Other hidden services included financial sites, token-based casinos, and a site that offers free novels via video format. These functions were frequently accessed, with a total of 16,652 invocations. The average active duration is 152.26 days, showing that they are relatively stable in directing traffic to concealed illegal services.

Moreover, abusers may also use scripts like "location.href" or <meta http-equiv="refresh"> tag for automatic redirects. The target domains can be static or dynamically generated randomly as exemplified in Table 4 in Appendix B, making tracking harder. Furthermore, once the concealed domain is blocked, abusers can easily promote a new one by updating the cloud function code. During our analysis, we extracted 13 redirected URLs. Excluding those that were inaccessible or redirected to well-known websites (e.g., www.sogou.com), three were flagged as potential malicious websites by McAfee Webadvisor [61], while one offers a gray service that allows users to watch ad-free anime and purchase game recharges and items. During the measurement period, each function was only called for 1 or 2 days, totaling 119 invocations. Despite low activity, these functions effectively conceal true domains and evade content-based detection.

Hidden promotion information. We identified a specific abuse scenario involving the promotion of OpenAI API Key sales, particularly in Aliyun serverless functions, with a total of 243 instances. These texts typically follow the format: "To purchase an API key (e.g., sk-s5S5BoV...), contact via *[contact information]*". We collected 28 distinct contact details, including WeChat, QQ, and email addresses. Repeated use of the same contact suggests group affiliation [58, 87], and our aggregation analysis shows clear clustering. Notably, the biggest group used one single WeChat across 157 functions. Miscreants registered a large number of cloud functions to promote the same information, maximizing exposure and dispersing risk, while keeping costs low. In addition to selling API keys, one group (comprising 14 functions) directly sold OpenAI accounts, claiming that for 10 RMB, users could purchase an account with an $18 credit for a trial.

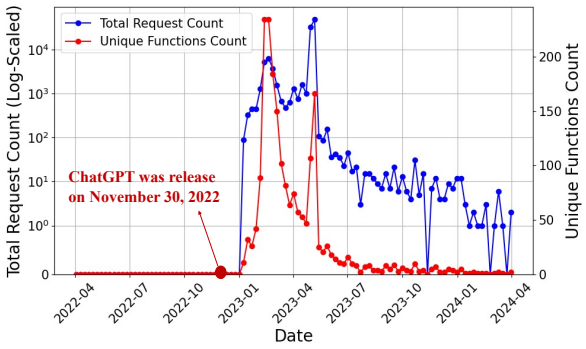

Figure 7: Trends in the misuse of cloud functions for the resale of OpenAI API Keys.

Through the passive DNS data, we observed that this abuse scenario first appeared in January 2023, just two months after the release of ChatGPT [84], and remained highly active until May 2023, as shown in Figure 7. The total request count for these functions reached 106,315, underscoring the scale of the abuse. We speculate the resale of OpenAI API keys is driven by restrictions on Chinese users, as OpenAI does not support local credit card payments. This has fueled a resale market for API keys and OpenAI accounts. Abusers profited through resale, with the largest abuse group explicitly stating that they earned 2 RMB for every 10 RMB spent in the transactions. Additionally, some reports suggest that sellers may use stolen credit cards to acquire the keys [44]. Furthermore, this unofficial service carries the risk of fraud, as buyers may pay for the keys but fail to receive the promised API access.

5.4 Abuse IV: Egress Nodes Abuse

Finding 9: Cloud functions exploit multiple egress IPs to create anonymous proxies, enabling them to support illegal services and bypass geographic restrictions.

In addition to the convenience of cloud functions, abusers could exploit the egress nodes of cloud functions. Due to the automatic scaling, providers dynamically allocate egress IP addresses when these functions make external network requests. Therefore, cloud

functions are an ideal infrastructure for deploying IP proxies, as each instance may be assigned different egress IP addresses.

Supporting illegal activities. According to the services agreements of cloud providers [3, 9, 12, 33, 52, 75], using serverless functions for proxy construction is allowed, but they must not be used for illegal activities. Through our analysis of serverless functions, we identified at least 20 functions acting as proxies for underground services, such as scraper services, a Ticketmaster puppeteer service that automatically purchases tickets on Ticketmaster, watermark-free downloads of TikTok videos, and free music downloads from Kuwo Music/QQ Music. These services violate both the terms of cloud providers and those of the target platforms [59]. From the perspective of the target platform, each request is from a different cloud IP addresses, effectively bypassing restrictions based on IP access frequency.

Bypassing geo-restrictions. Due to the geographic resolution, the outbound IPs usually align with the region of the cloud functions, regardless of the user's location [86]. This feature can be utilized to bypass regional restrictions. Within our data, OpenAI proxies are the most prevalent. By searching for keywords "OpenAI" and "ChatGPT", we manually confirmed 61 relevant functions and categorized them into two types. The first type includes 14 functions that create OpenAI frontends, offering users interfaces for interacting with OpenAI services. The second category includes 47 cloud functions that make simple requests to OpenAI, returning information such as API initialization or help messages. For example, *"This is a simple web application that interacts with OpenAI's chatbot API. Enter a message in the input box below"*. These functions act as intermediaries between users and OpenAI, forwarding user inputs to OpenAI while returning the generated responses back to users.

Similarly, we identified one GitHub proxy and four VPN proxies, which are also used to bypass geo-restrictions. Such abuse cases are particularly relevant in China, mainly due to the Chinese Great Firewall (GFW), which limit domestic users' access to OpenAI, GitHub[2], and other websites. By hosting serverless functions in specific regions, users can route their requests through the allocated IPs to circumvent these restrictions. We examined the region part of these function domains and observed that they were all configured in regions outside of China, further confirming the characteristics of this abuse. Moreover, these functions processed a total of 10,873 requests, indicating their significant usage.

5.5 Defense Gap of Serverless Abuse

Finding 10: The monitoring capabilities for serverless function abuse exist notably gaps, with only four identified cases flagged as malicious by VirusTotal.

By reviewing the returned content, we finally identified four abuse scenarios and eight specific cases, as summarized in Table 3. With rapid deployment, low costs, and multiple egress IPs, serverless functions offer a means to evade detection and conceal illegal actions. In total, among the content-rich functions we analyzed, 4.89% (594/12,138) were found to be abused, showing significant real-world impact with over 614k requests from users. Google and Aliyun are the most frequently abused providers, with a high number of functions used to host gambling websites and promote API key resale. All these abuses clearly violate cloud providers' terms of service [3, 9, 12, 33, 52, 75]. However, VirusTotal [38] flagged only four functions (used for C2 communication) as malicious, indicating insufficient detection from providers and threat intelligence services. Our work highlights the current state of cloud function abuse, showing that serverless cloud functions are increasingly targeted by cybercriminals and serve as a new infrastructure for illegal activities. We reported the abuse cases to affected providers and received supportive responses from Tencent and AWS. Although AWS noted the content was user-managed, they indicated a willingness to assist in review and remediation efforts. Overall, the responses reflect a recognition of the risks, and we hope this work can help raise broader awareness within the security community.

Table 3: Overall statistics of abused cloud functions.

Type	Cases	Functions	Requests
Abuse I	Hide C2 server	16	273,291
Abuse II	Gambling Website	194	24,979
	Porn-related Sites	8	854
	Cheating Tool	4	11,941
Abuse III	Redirct to New Domains	23	16,771
	Resale of OpenAI Key	243	106,315
Abuse IV	Illegal Service Proxy	20	170,195
	Geo-bypass Proxy	86	10,873
Total		594	614,219

6 Discussion

According to the measurement of serverless functions across nine providers, we gain insight into the fundamental usage patterns and identify potential abuse scenarios. In light of these findings, we would like to suggest the following recommendations to improve the management of cloud functions.

• **Strength the supervision of cloud function abuse.** Since all mentioned abuse cases stem from user actions, strengthening supervision is the most direct approach. While Aliyun and Tencent conduct random inspections as required by the Chinese government, we still observed ongoing misuse, suggesting that current measures are insufficient. Specifically, providers should incorporate additional review steps during the creation of functions. If the code shows typically abusive patterns, like hiding C2 communication, developers should be alerted to rectification. Sensitive terms related to underground industries should be monitored as well. While such measures may raise concerns about privacy violations, they can be mitigated through clear disclosures and obtaining informed consent from developers during function creation.

• **Secure the serverless architecture.** The security of the cloud function itself is crucial. "Warmonger" attack [86] highlighted how shared outbound IP addresses in serverless architectures can facilitate denial-of-service attacks. Our DNS-based analysis shows diverse resolution strategies across providers, with no unified standard. Some rely on third-party infrastructure, highlighting the need to secure these dependencies. Providers should also disable wildcard resolution and ensure DNS records are removed when functions

[2]Although GitHub is not blocked, its content delivery network (CDN) servers are located overseas, leading to slow access for users in China.

are deleted. Restricting resolution to active functions can better protect ingress nodes and reduce unnecessary load.

• **Enhancing the requirements of access control.** To mitigate potential data leaks from cloud functions, strengthening access control is crucial. Users should implement proper access restrictions and avoid embedding sensitive personal information in publicly accessible functions. The responsibility not only lies with users but also with platforms. Actually, providers such as Aliyun, AWS, and Google enforce default authentication when deploying functions, typically through IAM. Moreover, for AWS, if a user actively changes the setting to "None", a prominent red warning box appears to alert them that publicly accessible cloud functions may be flagged as a risk. In contrast, Baidu sets the default access to publicly accessible and provides no warning to users. Therefore, we recommend setting default access control and clearly informing users of the risks associated with making functions public. Alternatively, embedding authentication parameters in the default function URLs, as seen with Azure, could also enhance security. This shifts the responsibility for permission configurations to the provider, ensuring mandatory user verification upon each function invocation.

7 Related Work

Serverless functions have gained significant attention from academia and industry as a novel computing paradigm. In 2016, Hendrickson et al. [37] initiated the first analysis of AWS Lambda, highlighting that this new paradigm brings notable challenges to execution engines, databases, and schedulers. Following this foundational work, numerous measurements studies have expanded upon serverless computing, particularly focused on AWS Lambda [73], Azure Functions [71] and Huawei serverless services [46]. These studies emphasize the workload patterns to reveal the underlying architecture of FaaS platforms. Furthermore, Wang et al. [81] conducted a comparative analysis of AWS Lambda, Azure Functions, and Google Cloud Functions, offering insights into the performance differences across providers in terms of scalability, cold start latency, and resource efficiency. Similarly, Mcgrath et al. [62] developed a performance-oriented serverless computing platform to study serverless implementation considerations and provide a baseline for existing platform comparison. As serverless architectures evolve [74], further research have proposed optimization strategies on orchestration [57], startup latency [72, 79], and messaging mechanisms [22]. Moreover, there are ongoing efforts [26, 36] to integrate serverless functions with edge computing to empower the proliferation of IoT devices.

Our work offers a comprehensive analysis of serverless function usage and potential abuse across 9 leading cloud providers, serving as a foundational measurement reference for future improvements in the serverless ecosystem. While most studies focus on performance and analyze a limited number of providers without long-term observation, our research enables large-scale, cross-provider data analysis over two years. Regarding the revealed abuse risks, although some scenarios, like using proxies to circumvent censorship, are similar to those in CDNs [35, 83], this study is the first systematic examination of abuse cases within cloud functions, demonstrating that serverless functions are increasingly becoming a new infrastructure for malicious activities.

8 Conclusion

Through the analysis of passive DNS and the responses obtained from active access to cloud functions, we conducted a systematic measurement of the current state of the cloud function ecosystem. Our study reveals significant differences in the usage patterns and practical infrastructure among various providers. Furthermore, we conducted an initial analysis of potential abuse scenarios of cloud functions, uncovering privacy risks from unauthorized access. We identified four distinct misuse scenarios through manual review, revealing that 4.89% of these functions are being abused and have significant real-world impact. Furthermore, we have responsibly disclosed the identified abuse instances to the affected vendors to help them recognize and mitigate the risks of cloud function abuse. Additionally, based on our measurement findings, we offered valuable insights for providers to enhance the management of their serverless function offerings.

Acknowledgments

We thank our shepherd and the anonymous reviewers for their valuable feedback. This work is supported by the National Key Research and Development Program of China (No. 2023YFC3321303), National Natural Science Foundation of China (62102218, 62302258), and Zhongguancun Laboratory. Baojun Liu and Yiming Zhang are both the corresponding authors.

References

[1] Adjustment of the distribution model for free trial quotas of cloud functions. 2023-12. https://cloud.tencent.com.cn/document/product/583/73739.
[2] Aliyun Function Compute: A secure and stable, elastically scaled, O&M-free, pay-as-you-go, serverless computing platform. [n. d.]. https://www.alibabacloud.com/en/product/function-compute. (Access in October, 2024).
[3] Aliyun Service Agreement. [n. d.]. https://terms.aliyun.com/legal-agreement/terms/suit_bu1_ali_cloud/suit_bu1_ali_cloud201802281451_77479.html?spm=a2c4g.11186623.0.0.14686ad6jeyPNg. (Access in October, 2024).
[4] Amazon. [n. d.]. Amazon Simple Queue Service. https://aws.amazon.com/sqs/. (Access in October, 2024).
[5] Announcing AWS Lambda Function URLs: Built-in HTTPS Endpoints for Single-Function Microservices. 2022-04. https://aws.amazon.com/blogs/aws/announcing-aws-lambda-function-urls-built-in-https-endpoints-for-single-function-microservices/.
[6] Attack and Defend: Leveraging AWS Serverless Technology for End-to-End C2. 2022-9. https://www.youtube.com/watch?v=wl9GuvOFSKo.
[7] AWS Lambda randomly gives back 502 as status. [n. d.]. https://stackoverflow.com/questions/49810775/aws-lambda-randomly-gives-back-502-as-status. (Access in October, 2024).
[8] AWS Lambda: Run code without thinking about servers or clusters 2024-9. https://aws.amazon.com/lambda/.
[9] AWS Service Terms. [n. d.]. https://aws.amazon.com/cn/service-terms/. (Access in October, 2024).
[10] Azure Functions: Execute event-driven serverless code with an end-to-end development experience 2024-9. https://azure.microsoft.com/en-us/products/functions.
[11] Baidu Cloud Function Compute [n. d.]. https://cloud.baidu.com/doc/CFC/index.html. (Access in October, 2024).
[12] Baidu Cloud Service Agreement: User Rights and Obligations. [n. d.]. https://cloud.baidu.com/doc/Agreements/s/mjwvy1waw. (Access in October, 2024).
[13] Jack Cable, Drew Gregory, Liz Izhikevich, and Zakir Durumeric. 2021. Stratosphere: Finding vulnerable cloud storage buckets. In Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses. 399–411.
[14] China Mobile. [n. d.]. https://www.chinamobileltd.com/en/global/home.php. (Access in October, 2024).
[15] China Telecom. [n. d.]. https://www.chinatelecom-h.com/en/global/home.php. (Access in October, 2024).
[16] China Unicom. [n. d.]. https://www.chinaunicom.com.cn/. (Access in October, 2024).

[17] Cisco Umbrella Passive DNS. 2024-9. https://docs.umbrella.com/investigate/docs/passive-dns.

[18] Cloud Functions (2nd gen) is now the default choice in the Google Cloud console user interface on August 29, 2023. 2023-08. https://cloud.google.com/functions/docs/release-notes#August_29_2023.

[19] Cloud Functions has released Cloud Functions on February 14, 2022. 2022-02. https://cloud.google.com/functions/docs/release-notes#February_14_2022(2ndgen).

[20] Cloudflare 2024-9. https://www.cloudflare.com/.

[21] CNCF Serverless Whitepaper v1.0 2018-9. https://github.com/cncf/wg-serverless/tree/master/whitepapers/serverless-overview.

[22] Marcin Copik, Roman Böhringer, Alexandru Calotoiu, and Torsten Hoefler. 2023. Fmi: Fast and cheap message passing for serverless functions. In Proceedings of the 37th International Conference on Supercomputing. 373–385.

[23] Jakub Czyz, Mark Allman, Jing Zhang, Scott Iekel-Johnson, Eric Osterweil, and Michael Bailey. 2014. Measuring ipv6 adoption. In Proceedings of the 2014 ACM Conference on SIGCOMM. 87–98.

[24] Datadog. 2023. The state of serverless. https://www.datadoghq.com/state-of-serverless/.

[25] Ali Davanian, Michail Faloutsos, and Martina Lindorfer. 2024. C2Miner: Tricking IoT Malware into Revealing Live Command & Control Servers. In Proceedings of the 19th ACM Asia Conference on Computer and Communications Security. 112–127.

[26] Eyal De Lara, Carolina S Gomes, Steve Langridge, S Hossein Mortazavi, and Meysam Roodi. 2016. Hierarchical serverless computing for the mobile edge. In 2016 IEEE/ACM Symposium on Edge Computing (SEC). IEEE, 109–110.

[27] Deprecation overview. [n. d.]. https://cloud.ibm.com/docs/openwhisk?topic=openwhisk-dep-overview. (Access in October, 2024).

[28] EarlyBird. [n. d.]. https://github.com/americanexpress/earlybird. (Access in October, 2024).

[29] Soufian El Yadmani, Olga Gadyatskaya, and Yury Zhauniarovich. 2024. The File That Contained the Keys Has Been Removed: An Empirical Analysis of Secret Leaks in Cloud Buckets and Responsible Disclosure Outcomes. In 2025 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 9–9.

[30] Function timeout. [n. d.]. https://cloud.google.com/functions/docs/configuring/timeout. (Access in October, 2024).

[31] Function URL Overview. 2023-08. https://cloud.tencent.com/document/product/583/96099.

[32] Google. [n. d.]. Google Pub/Sub. https://cloud.google.com/pubsub?hl=en. (Access in October, 2024).

[33] Google Cloud Platform/SecOps Terms of Service. [n. d.]. https://cloud.google.com/terms/. (Access in October, 2024).

[34] Google Cloud Run Functions (formerly known as Cloud Functions) 2024-9. https://cloud.google.com/functions.

[35] Run Guo, Jianjun Chen, Baojun Liu, Jia Zhang, Chao Zhang, Haixin Duan, Tao Wan, Jian Jiang, Shuang Hao, and Yaoqi Jia. 2018. Abusing CDNs for fun and profit: Security issues in CDNs' origin validation. In 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS). IEEE, 1–10.

[36] Adam Hall and Umakishore Ramachandran. 2019. An execution model for serverless functions at the edge. In Proceedings of the International Conference on Internet of Things Design and Implementation. 225–236.

[37] Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. 2016. Serverless computation with {OpenLambda}. In 8th USENIX workshop on hot topics in cloud computing (HotCloud 16).

[38] Hispasec Sistemas Company. [n. d.]. Virus Total. https://www.virustotal.com/gui/home/search. (Access in October, 2024).

[39] How can the HTTP timeout for function computing be set to a longer duration? [n. d.]. https://developer.aliyun.com/ask/571611. (Access in October, 2024).

[40] How do I troubleshoot HTTP 502 and HTTP 500 status code (server-side) errors from AWS Lambda? [n. d.]. https://repost.aws/knowledge-center/lambda-troubleshoot-invoke-error-502-500. (Access in October, 2024).

[41] How much do the top-ranking dating apps really make? Let's take a look at their profit-making tactics. [n. d.]. https://www.163.com/dy/article/F9UGKSLL051181GK.html. (Access in October, 2024).

[42] IBM Cloud Functions: Functions-as-a-Service (FaaS) platform based on Apache OpenWhisk. [n. d.]. https://cloud.ibm.com/functions/. (Access in October, 2024).

[43] Introduction of Kingsoft Function URL. 2022-08. https://docs.ksyun.com/documents/42034.

[44] Is there a risk in hiring someone to top up an OpenAI API account? Could the account be banned? 2024-04. https://chatgptboke.com/is-there-any-risk-in-finding-someone-to-recharge-openai-api.html.

[45] Wayne Jansen, Tim Grance, et al. 2011. Guidelines on security and privacy in public cloud computing. (2011).

[46] Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Darlow, Jianfeng Wang, and Adam Barker. 2023. How does it function? characterizing long-term trends in production serverless workloads. In Proceedings of the 2023 ACM Symposium on Cloud Computing. 443–458.

[47] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. 2001. DNS performance and the effectiveness of caching. In Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement. 153–167.

[48] Daniel Kelly, Frank G Glavin, and Enda Barrett. 2021. Denial of wallet—defining a looming threat to serverless computing. Journal of Information Security and Applications 60 (2021), 102843.

[49] Erin Kenneally and David Dittrich. 2012. The menlo report: Ethical principles guiding information and communication technology research. Available at SSRN 2445102 (2012).

[50] Beom Heyn Kim and David Lie. 2015. Caelus: Verifying the consistency of cloud services with battery-powered devices. In 2015 IEEE Symposium on Security and Privacy. IEEE, 880–896.

[51] Kingsoft Cloud Function [n. d.]. https://www.ksyun.com/nv/product/KCF. (Access in October, 2024).

[52] Kingsoft Cloud Service Agreement. [n. d.]. https://docs.ksyun.com/documents/42028?type=3. (Access in October, 2024).

[53] Ralph LaBarge and Thomas McGuire. 2013. Cloud penetration testing. arXiv preprint arXiv:1301.1912 (2013).

[54] Lambda Function URL is throwing 502. [n. d.]. https://repost.aws/questions/QU85XAcFFZRR-xatC-RhwfAQ/lambda-function-url-is-throwing-502. (Access in October, 2024).

[55] Xiaojing Liao, Sumayah Alrwais, Kan Yuan, Luyi Xing, XiaoFeng Wang, Shuang Hao, and Raheem Beyah. 2016. Lurking malice in the cloud: Understanding and detecting cloud repository as a malicious service. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 1541–1552.

[56] Xiaojing Liao, Chang Liu, Damon McCoy, Elaine Shi, Shuang Hao, and Raheem Beyah. 2016. Characterizing long-tail SEO spam on cloud web hosting services. In Proceedings of the 25th International Conference on World Wide Web. 321–332.

[57] David H Liu, Amit Levy, Shadi Noghabi, and Sebastian Burckhardt. 2023. Doing more with less: Orchestrating serverless applications without an orchestrator. In 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23). 1505–1519.

[58] Mingxuan Liu, Yiming Zhang, Baojun Liu, Zhou Li, Haixin Duan, and Donghong Sun. 2021. Detecting and characterizing SMS spearphishing attacks. In Proceedings of the 37th Annual Computer Security Applications Conference. 930–943.

[59] Yijing Liu, Yiming Zhang, Baojun Liu, Haixin Duan, Qiang Li, Mingxuan Liu, Ruixuan Li, and Jia Yao. 2024. Tickets or Privacy? Understand the Ecosystem of Chinese Ticket Grabbing Apps. In 33rd USENIX Security Symposium (USENIX Security 24). 5107–5124.

[60] MasterCard DNS Error Went Unnoticed for Years [n. d.]. https://krebsonsecurity.com/2025/01/mastercard-dns-error-went-unnoticed-for-years/. (Access in March, 2025).

[61] McAfee Webadvisor. [n. d.]. https://www.mcafee.com/en-us/safe-browser/mcafee-webadvisor.html. (Access in October, 2024).

[62] Garrett McGrath and Paul R Brenner. 2017. Serverless computing: Design, implementation, and performance. In 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). IEEE, 405–410.

[63] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. arXiv preprint arXiv:1109.2378 (2011).

[64] Antonio Nappa, Zhaoyan Xu, M Zubair Rafique, Juan Caballero, and Guofei Gu. 2014. CyberProbe: Towards Internet-Scale Active Detection of Malicious Servers.. In NDSS.

[65] Oracle Cloud Functions [n. d.]. https://www.oracle.com/cloud/cloud-native/functions/. (Access in October, 2024).

[66] Passive DNS historical internet database: Farsight DNSDB. 2024-9. https://www.farsightsecurity.com/solutions/dnsdb/.

[67] QAX - The Official Cyber Security Services and Anti-Virus Software Sponsor of the Olympic and Paralympic Winter Games Beijing 2022. [n. d.]. https://en.qianxin.com/. (Access in October, 2024).

[68] Serverless Cloud Computing: Are there actually No Servers? 2023-06. https://chaoskyle.com/serverless-cloud-computing.

[69] Serverless Cloud Function Market. [n. d.]. https://www.linkedin.com/pulse/serverless-cloud-function-market-data-dive-research-analysis-paazc/. (Access in October, 2024).

[70] ServerlessC2. 2022-9. https://github.com/hackerob/ServerlessC2.

[71] Mohammad Shahrad, Rodrigo Fonseca, Inigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In 2020 USENIX annual technical conference (USENIX ATC 20). 205–218.

[72] Paulo Silva, Daniel Fireman, and Thiago Emmanuel Pereira. 2020. Prebaking functions to warm the serverless cold start. In Proceedings of the 21st International Middleware Conference. 1–13.

[73] Josef Spillner. 2019. Quantitative analysis of cloud function evolution in the AWS serverless application repository. arXiv preprint arXiv:1905.04800 (2019).

[74] Davide Taibi, Nabil El Ioini, Claus Pahl, and Jan Raphael Schmid Niederkofler. 2020. Patterns for serverless functions (function-as-a-service): A multivocal

literature review. (2020).
[75] Tencent Serverless Cloud Function Service Agreement. [n. d.]. https://cloud.tencent.com/document/product/583/59194. (Access in October, 2024).
[76] Tencent Serverless Cloud Functions. [n. d.]. https://www.tencentcloud.com/products/scf. (Access in October, 2024).
[77] Top 5 Serverless Platforms That Take Off. 2022-12. https://www.techmagic.co/blog/top-serverless-platforms.
[78] Top Serverless Platforms in 2023. 2023-10. https://chrisbateson80.medium.com/top-serverless-platforms-in-2023-2fde4104441d.
[79] Dmitrii Ustiugov, Plamen Petrov, Marios Kogias, Edouard Bugnion, and Boris Grot. 2021. Benchmarking, analysis, and optimization of serverless function snapshots. In Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 559–572.
[80] Ao Wang, Jingyuan Zhang, Xiaolong Ma, Ali Anwar, Lukas Rupprecht, Dimitrios Skourtis, Vasily Tarasov, Feng Yan, and Yue Cheng. 2020. {InfiniCache}: exploiting ephemeral serverless functions to build a {cost-effective} memory cache. In 18th USENIX conference on file and storage technologies (FAST 20). 267–281.
[81] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. 2018. Peeking behind the curtains of serverless platforms. In 2018 USENIX annual technical conference (USENIX ATC 18). 133–146.
[82] Warning on Unauthorised Investment Firm/Investment Business Firm issued. 2018-10. https://www.centralbank.ie/news/article/warning-on-unauthorised-investment-firm-investment-business-firm-issued8Oct2018.
[83] Mingkui Wei. 2021. Domain Shadowing: Leveraging Content Delivery Networks for Robust {Blocking-Resistant} Communications. In 30th USENIX Security Symposium (USENIX Security 21). 3327–3343.
[84] Wikipedia of ChatGPT [n. d.]. https://en.wikipedia.org/wiki/ChatGPT. (Access in October, 2024).
[85] Qinge Xie, Shujun Tang, Xiaofeng Zheng, Qingran Lin, Baojun Liu, Haixin Duan, and Frank Li. 2022. Building an Open, Robust, and Stable {Voting-Based} Domain Top List. In 31st USENIX Security Symposium (USENIX Security 22). 625–642.
[86] Junjie Xiong, Mingkui Wei, Zhuo Lu, and Yao Liu. 2021. Warmonger: inflicting denial-of-service via serverless functions in the cloud. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 955–969.
[87] Yiming Zhang, Baojun Liu, Chaoyi Lu, Zhou Li, Haixin Duan, Shuang Hao, Mingxuan Liu, Ying Liu, Dong Wang, and Qiang Li. 2020. Lies in the air: Characterizing fake-base-station spam ecosystem in china. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 521–534.
[88] Minqi Zhou, Rong Zhang, Wei Xie, Weining Qian, and Aoying Zhou. 2010. Security and privacy in cloud computing: A survey. In 2010 sixth international conference on semantics, knowledge and grids. IEEE, 105–112.

A Ethics

Although our institution does not have an Institutional Review Board (IRB), we proactively obtained approval from the organization that served similar functions in our institution and strictly adhered to ethical guidelines [49] throughout the experiment. Firstly, for the passive DNS data collection, the data we used contained no personally identifiable information. Specifically, fields such as client IP addresses were excluded from our dataset. Secondly, for the active access to serverless functions, to minimize the impact on the function owner, we did not collect any function code and limited our requests to fewer than three per function. Since most providers offer a free usage tier, our requests were unlikely to incur significant charges for the function owners. We conducted tests on our own cloud functions to ensure that the impact on function owners was minimal. Additionally, on the servers initiating our probe requests, we launched a web service on port 80. The site provided an explanation of the experiment's purpose, along with the researcher's contact information. We offered an opt-out option for participants (cloud function owners), and if they opted out, we would stop accessing their functions and discard all related data to ensure it was not used for further analysis. Lastly, for the returned content of functions, the responses from each active request were securely stored locally. Considering that cloud functions may contain sensitive data, as mentioned in Section 3.4, we identified and hashed the potentially sensitive data prior to large-scale analysis. Importantly, we did not analyze any sensitive information directly, thereby mitigating the ethical risks. To reduce potential harm and highlight the security implications, we reported identified abuse cases to the affected service providers and received positive feedback from Tencent and AWS.

B Supplementary chart

The following are some supplementary figures and tables provided for a better understanding of the paper. As described in Section 5, we identified four types of cloud function abuse, including covert C2 communication, hosting malicious websites, concealing illicit services, and using egress nodes as IP proxies. Algorithm 1 shows the implementation logic of a cloud function used to hide a C2 server, Figure 8 provides examples of gambling websites hosted on cloud functions, and Table 4 exemplifies typical methods of concealing illicit services via redirection.

Algorithm 1: Code Structure for Hiding C2 Servers

Input : HTTP request details
Output: Response from hidden C2 server

```
C2 ← "http://[C2_server_ip]:[port]"
path ← event['path']
headers ← event['headers']
params ← event['queryString']
body ← event['body']
if event['httpMethod'] == 'GET' then
    resp ← GET(C2 + path, headers, params)
else
    resp ← POST(C2 + path, body, headers, params)
response ← { "isBase64Encoded": True, "statusCode":
  resp.status_code, "headers": resp.headers, "body":
  base64_encode(resp.content) }
return response
```

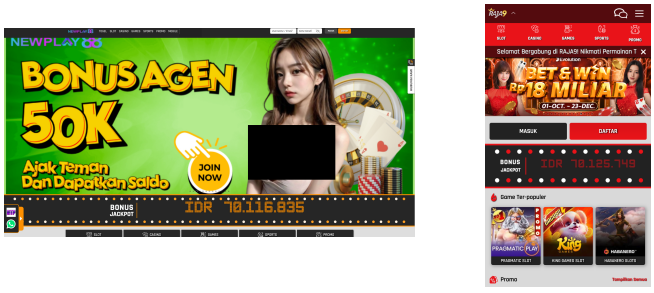

Figure 8: Cloud functions can be used to host malicious websites. The screenshots show the typical gambling sites on cloud functions.

Table 4: Cloud functions can be exploited to promote illegal services, primarily by redirecting users to target domains. These target domains may be generated either statically or dynamically, as shown in the examples.

Target Domains	Examples
Static Display	`location.href = "http://dlcy.zeldalink.top/wlxcList.html"`
Random Splicing	`var Rand = Math.round(Math.random() * 999999)` `location.href="https://"+Rand+".yerbsdga.xyz"`

Random Selection	`const urls =[`
	`  'https://polaris.zijieapi.com/luckycat/super_inviter/v1/invite_code/..',`
	`  'https://www.bilibili.com/',`
	`  'https://www.bilibili.com/',`
	`  'https://www.bilibili.com/', ]`
	`const url = urls[Math.floor(Math.random() * urls.length)]`
	`location.href = url`