

The HazyBeacon Protocol – How Malware Weaponizes Amazon Web Services (AWS) Lambda Function URLs

By Aniket Harne

Published: 2026-06-02 · Archived: 2026-06-04 02:00:29 UTC

Key Takeaways

- HazyBeacon (CL-STA-1020) targets Southeast Asian government networks by abusing AWS Lambda Function URLs configured with AuthType: NONE as stealth command-and-control relays.
- Attackers use stolen IAM credentials to deploy Lambda functions that proxy malware communications through trusted AWS domains.
- Organizations can reduce exposure by enforcing identity-centric access controls, enabling global CloudTrail logging, enabling VPC flow telemetry, and implementing Service Control Policies that restrict Lambda Function URL exposure, all supported by continuous configuration monitoring.

The Rise of Cloud-Native Command and Control (C2)

Command and control (C2) infrastructure traditionally lived outside the victim environment. Malware beacons to attacker-operated servers hosted on rented VPS infrastructure or compromised websites, and defenders focused on identifying those endpoints through IP reputation, domain intelligence, and network blocking.

Cloud computing has started to blur that boundary. In some campaigns, the infrastructure issuing commands is no longer owned by the attacker at all. It is deployed inside legitimate cloud platforms using stolen IAM credentials and standard service features such as serverless compute and public HTTPS endpoints.

The HazyBeacon campaign documented by Palo Alto Networks Unit 42 in July 2025 illustrates this shift. Infected systems communicated with AWS Lambda Function URLs hosted inside Amazon infrastructure. To most network defenses, the traffic appeared as ordinary HTTPS traffic to a trusted cloud service, while the Lambda function itself operated as a relay for an attacker-controlled command-and-control infrastructure.

Qualys Insights

Borrowed Infrastructure Attacks

HazyBeacon reflects an emerging model of cloud-enabled cyber operations. Instead of hosting command servers on attacker-controlled systems, adversaries deploy infrastructure inside compromised cloud accounts.

In this model, the victim environment becomes part of the attacker's operational stack.

A typical Borrowed Infrastructure Attack follows four stages:

1. A credential compromise grants access to a cloud account.
2. Infrastructure is deployed using legitimate cloud APIs.
3. Malware communications are routed through trusted cloud endpoints.
4. Attribution becomes difficult because the infrastructure is owned by an unrelated organization.

This approach offers several operational advantages for attackers. Cloud platforms provide globally distributed infrastructure, elastic scaling, and low-friction deployment through legitimate APIs, allowing to blend malware traffic into legitimate cloud activity.

The Malware Payload

Once installed on a victim's endpoint, HazyBeacon operates as a lightweight downloader and execution framework designed to remain flexible and evasive.

Its primary functions include:

1. System Enumeration: Gathering hostname, IP address, user privileges, and operating system (OS) version.
2. Remote Task Execution: Receiving encrypted commands to download further payloads or execute shell commands.
3. Data Exfiltration: Uploading stolen documents and captured keystrokes.

The Infrastructure Pivot

Rather than hosting dedicated command servers, the attackers behind HazyBeacon compromised unrelated AWS accounts and deployed lightweight Lambda-based relays inside those environments.

These accounts often belonged to development teams or smaller organizations where IAM credentials had been exposed or poorly governed. Once inside the environment, the attackers deployed Lambda functions that proxied malware communications between infected systems and the real backend command infrastructure.

The communication chain typically follows this pattern:

- Malware (Victim A) sends an encrypted HTTP POST to the Lambda function URL (Victim B's Account).
- Lambda strips the headers, logs the basic metadata (for the attacker's analytics), and forwards the payload to the attacker's backend server.
- The attacker's backend server responds to the Lambda, which relays the command back to the Malware.

This "middleman" architecture makes attribution incredibly difficult. The malware victim sees traffic going to Amazon infrastructure. The attacker's real server sees requests originating from Amazon infrastructure. The victim or the AWS account hosting the Lambda often has no awareness that its resources are participating in a global espionage network until they receive an abuse notice or a massive bill.

Why Lambda Function URLs Are the Ultimate C2 Proxy

To understand why this attack is so effective, it is important to examine the feature being exploited, i.e., AWS Lambda Function URLs, introduced in April 2022.

Before Function URLs, exposing a Lambda function to the public internet required setting up an Amazon API Gateway or an Application Load Balancer (ALB). These services are powerful but complex, logging-heavy, and incur additional costs. They leave a larger footprint.

AWS Lambda Function URLs allow developers to expose a serverless function through a dedicated HTTPS endpoint without configuring API Gateway or a load balancer.

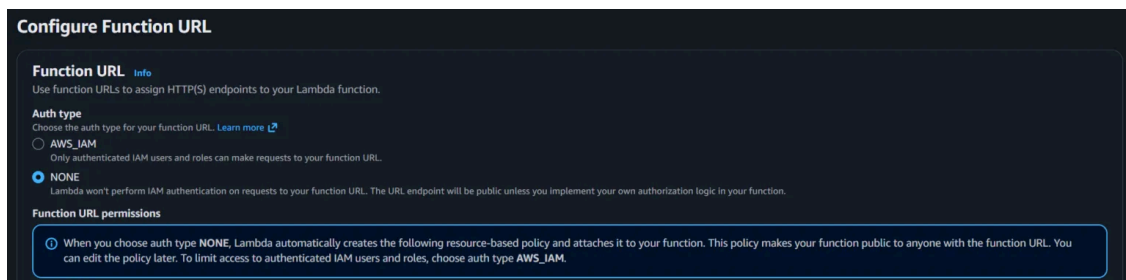
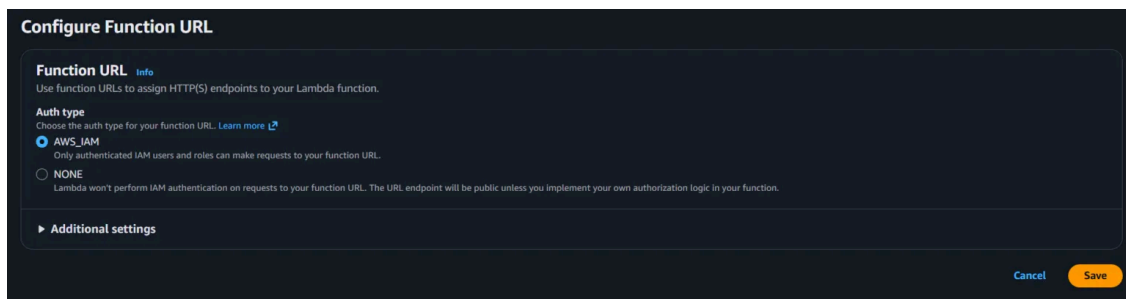
A typical endpoint looks like:

`https://<url-id>.lambda-url.<region>.on.aws`

Crucially, they support two authentication modes:

1. **AWS_IAM**: Requires the caller to sign requests with valid IAM credentials.
2. **NONE**: Allows unauthenticated, public access from anywhere on the internet.

Threat actors prefer the second option, **AuthType: NONE**, because it allows them to deploy a public HTTPS relay inside AWS infrastructure within seconds.



The “Lookalike” Problem

Because the domain ends in `on.aws`, the endpoint inherits the trust associated with Amazon Web Services domains. To a Security Operations Center (SOC) analyst reviewing network telemetry, the traffic appears indistinguishable from routine AWS activity.

The Cloud Infrastructure Kill Chain

The deployment of a HazyBeacon command node follows a predictable Cyber Kill Chain rooted in basic cloud hygiene failures. It rarely involves zero-day exploits against AWS itself; instead, attackers exploit basic identity and configuration weaknesses within cloud environments.

The attack typically unfolds in several phases:

Phase 1: Reconnaissance – Credential Harvesting

1. Credential harvesting provides initial access.
2. Theft of AWS access keys from public GitHub repositories, developer phishing campaigns, or malware harvesting ~/.aws/ credentials.
3. Credentials are typically static IAM keys created long ago and never rotated.
4. No interaction with AWS infrastructure occurs at this stage.

Phase 2: Weaponization – Access Validation

1. Access validation follows.
2. Stolen credentials are validated using low-noise API calls.
3. Common commands include `aws sts get-caller-identity` and `aws iam list-attached-user-policies`.
4. The attacker enumerates permissions such as `lambda:CreateFunction` and `lambda:CreateFunctionUrlConfig`.

Phase 3: Delivery and Exploitation – Infrastructure Deployment

1. Infrastructure deployment occurs next.
2. Legitimate AWS APIs are used as the delivery mechanism.
3. A zipped Python or Node.js payload is uploaded as a Lambda function.
4. Functions are given benign names (e.g., `UpdateWorker`) to evade casual inspection.
5. Deployment often occurs in regions with reduced scrutiny, such as `sa-east-1` or `eu-north-1`.

Phase 4: Installation & Command and Control

1. Command channel activation occurs.
2. Persistence is established by creating a Lambda Function URL.
3. The Function URL is configured with `AuthType: NONE`.
4. The URL is integrated into the attacker's C2 infrastructure.

Phase 5: Actions on Objectives

1. Once active, the compromise spreads through the Lambda function.
2. The victim's AWS account becomes an unwitting C2 relay for malware communications.
3. Thousands of command-and-control requests are processed through the function per hour.
4. The activity appears operationally normal within most enterprise environments.

MITRE ATT&CK Deep Dive: Mapping the Serverless Attack Surface

This campaign maps directly to several techniques in the MITRE ATT&CK framework for cloud environments.

Tactic	Technique ID	Technique Name	Context in HazyBeacon
Initial Access	T1078.004	Valid Accounts Cloud Accounts	Usage of stolen, static IAM Access Keys to enter the cloud environment.
Execution	T1648	Serverless Execution	Creating new Lambda functions that persist independently of the compromised user session.
Defense Evasion	T1564	Hide Artifacts	Deploying in unused regions; using benign naming conventions (“BackupHandler”, “ImageResizer”).
Command & Control	T1102	Web Service	Using AWS Lambda as the communication channel to blend with legitimate web traffic.
Command & Control	T1090	Proxy	The Lambda function serves purely as a hop point to obscure the true destination.

This mapping illustrates that the attack does not rely on sophisticated exploits. It relies on predictable weaknesses in identity governance and infrastructure monitoring.

How to Prevent Cloud Infrastructure from Becoming Command Infrastructure

Campaigns like HazyBeacon reveal how attackers repurpose cloud services for command channels; the defensive response must focus on the layers that govern how infrastructure is created and used. It requires visibility and control across three layers of the cloud operating model: identity, the control plane, and infrastructure behavior.

Identity Hygiene is the New Security Perimeter

In cloud environments, identity effectively defines the security boundary. Campaigns like HazyBeacon depend on compromised IAM credentials that allow attackers to deploy Lambda functions and expose public Function URLs. Controls that disable unused access keys, enforce regular key rotation, and require multi-factor authentication significantly reduce this attack surface. Controls that disable unused access keys, enforce regular key rotation, and require multi-factor authentication significantly reduce the likelihood that attackers can deploy malicious infrastructure.

Strong credential governance removes one of the primary prerequisites for serverless command infrastructure.

Control Plane Visibility Reveals Infrastructure Deployment

If identity controls fail, detection shifts to the control plane.

Cloud environments produce detailed telemetry for every infrastructure action. Services such as AWS CloudTrail record API calls used to create resources such as Lambda functions and Function URLs. When logging is enabled across all regions, attempts to deploy infrastructure in less monitored locations become visible.

Monitoring anomalous API activity can often reveal compromised credentials during reconnaissance or privilege testing, before attackers establish operational command infrastructure.

Infrastructure Behavior Exposes Command Relays

Even though Lambda is serverless, it still generates observable network patterns.

Organizations that route Lambda workloads through Virtual Private Clouds can collect flow telemetry that exposes proxy-like behavior. A command relay typically produces a near one-to-one relationship between inbound and outbound requests, as the function forwards traffic between malware clients and attacker infrastructure.

Such patterns can reveal when a Lambda function is operating as a relay rather than executing a legitimate workload. Broader configuration weaknesses, including overly permissive security groups, also tend to correlate with higher rates of identity compromise.

High-Signal Controls for Detecting and Restricting Lambda Abuse

Beyond detection, organizations should actively restrict the conditions that allow attackers to expose serverless infrastructure to the internet.

Two controls are particularly effective:

1. Enforcing Zero Trust Function Policy

Organizations can implement Service Control Policies (SCPs) at the AWS Organization level that prevent the creation of Function URLs with `AuthType: NONE` unless the function is exclusively approved through tagging or policy exceptions (e.g., `PublicFacing: True`).

This approach enforces a zero-trust model for public serverless endpoints. Even if an attacker gains access to valid IAM credentials, the API call required to expose a Lambda function publicly will fail.

2. Monitoring Cost Anomalies in Serverless Workloads

Command infrastructure generates volume. A command relay supporting large numbers of infected systems can produce thousands or even millions of Lambda invocations.

Monitoring service-level cost anomalies can therefore act as an additional detection signal. Granular AWS budget alerts for Lambda compute usage can reveal unexpected spikes in invocation activity, particularly in nonproduction regions. Such patterns often indicate infrastructure abuse, including command relays or unauthorized compute workloads.

Operationalizing Cloud Infrastructure Defense with Qualys TotalCloud™

Applying these controls consistently across large cloud environments requires continuous visibility across accounts, identities, and serverless workloads.

[Qualys TotalCloud™](#) helps organizations identify exposed IAM credentials, monitor Lambda Function URL creation, detect risky configuration drift, and surface abnormal serverless activity from a unified cloud posture management layer.

By correlating identity, infrastructure, and workload telemetry, TotalCloud helps security teams identify the conditions that enable cloud-native command infrastructure before those resources become operational attack relays.

Closing the Door on Cloud-Native Malware

The HazyBeacon campaign demonstrates how cloud infrastructure can be repurposed into operational command infrastructure when identity and configuration controls fail.

For attackers, cloud environments are the perfect hideout: high trust, low cost, and endless scale. For defenders, however, the cloud offers a unique advantage: Visibility. Unlike on-premises networks, where a rogue server might hide under a desk, in the cloud, every asset, every API call, and every configuration change is logged and leaves a record.

When organizations enforce strong identity governance, maintain visibility across the cloud control plane, and monitor infrastructure behavior, they remove the conditions that allow attackers to turn legitimate infrastructure into operational malware.

See how Qualys Kubernetes and Container Security helps organizations modernize lifecycle visibility for containerized applications and Kubernetes environments

Frequently Asked Questions (FAQs)

What is HazyBeacon?

HazyBeacon is a Windows backdoor that uses AWS Lambda Function URLs as a command-and-control relay. Malware communicates with a Lambda endpoint, which forwards instructions and stolen data to the attacker's backend server.

Why are AWS Lambda Function URLs used for command and control

When configured with **AuthType NONE**, Lambda Function URLs create public HTTPS endpoints. Because they run on trusted AWS domains, malicious traffic can blend in with normal cloud traffic.

Does HazyBeacon exploit a vulnerability in AWS?

No. The attack relies on stolen IAM credentials and misconfigured permissions, not flaws in AWS services.

Why is this type of command infrastructure hard to detect?

Traffic to AWS infrastructure appears as legitimate encrypted HTTPS traffic, making it difficult for traditional network defenses to distinguish malicious activity.

How do attackers deploy Lambda-based command relays?

Attackers use compromised IAM credentials to create Lambda functions and expose them through public Function URLs.

How can organizations reduce the risk of this attack?

By enforcing strong identity governance, enabling CloudTrail logging, restricting public Function URLs, and monitoring unusual Lambda activity.

Source: <https://blog.qualys.com/qualys-insights/2026/06/02/hazybeacon-aws-lambda-function-url-command-control-abuse>