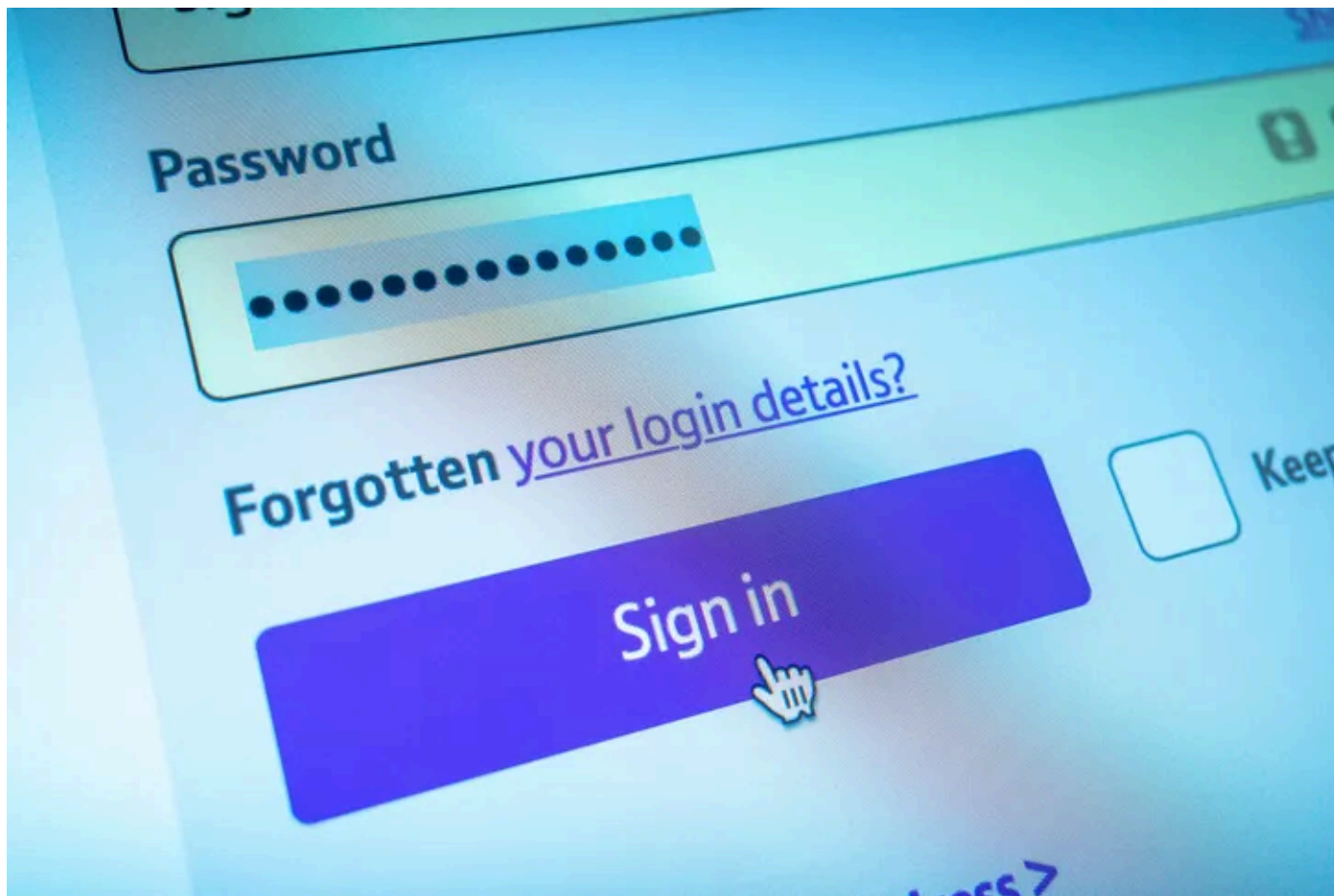


Xillen Stealer Updates to Version 5 to Evade AI Detection

 darktrace.com/blog/xillen-stealer-updates-to-version-5-to-evade-ai-detection

Tara Gould

November 20, 2025



Introduction

Python-based information stealer “Xillen Stealer” has recently released versions 4 and 5, expanding its targeting and functionality. The cross-platform infostealer, originally reported by [Cyfirma](#) in September 2025, targets sensitive data including credentials, cryptocurrency wallets, system information, browser data and employs anti-analysis techniques.

The update to v4/v5 includes significantly more functionality, including:

- Persistence
- Ability to steal credentials from password managers, social media accounts, browser data (history, cookies and passwords) from over 100 browsers, cryptocurrency from over 70 wallets
- Kubernetes configs and secrets
- Docker scanning
- Encryption
- Polymorphism

- System hooks
- Peer-to-Peer (P2P) Command-and-Control (C2)
- Single Sign-On (SSO) collector
- Time-Based One-Time Passwords (TOTP) and biometric collection
- EDR bypass
- AI evasion
- Interceptor for Two-Factor Authentication (2FA)
- IoT scanning
- Data exfiltration via Cloud APIs

Xillen Stealer is marketed on Telegram, with different licenses available for purchase. Users who deploy the malware have access to a professional-looking GUI that enables them to view exfiltrated data, logs, infections, configurations and subscription information.

Figure 1: Screenshot of the Xillen Stealer portal.

Technical analysis

The following technical analysis examines some of the interesting functions of Xillen Stealer v4 and v5. The main functionality of Xillen Stealer is to steal cryptocurrency, credentials, system information, and account information from a range of stores.

Xillen Stealer specifically targets the following wallets and browsers:

AITargetDetection

Figure 2: Screenshot of Xillen Stealer's AI Target detection function.

The 'AITargetDetection' class is intended to use AI to detect high-value targets based on weighted indicators and relevant keywords defined in a dictionary. These indicators include "high value targets", like cryptocurrency wallets, banking data, premium accounts, developer accounts, and business emails. Location indicators include high-value countries such as the United States, United Kingdom, Germany and Japan, along with cryptocurrency-friendly countries and financial hubs. Wealth indicators such as keywords like CEO, trader, investor and VIP have also been defined in a dictionary but are not in use at this time, pointing towards the group's intent to develop further in the future.

While the class is named 'AITargetDetection' and includes placeholder functions for initializing and training a machine learning model, there is no actual implementation of machine learning. Instead, the system relies entirely on rule-based pattern matching for detection and scoring. Even though AI is not actually implemented in this code, it shows how malware developers could use AI in future malicious campaigns.

Figure 3: Screenshot of dead code function.

AI Evasion

Figure 4: Screenshot of AI evasion function to create entropy variance.

'AIEvasionEngine' is a module designed to help malware evade AI-based or behavior-based detection systems, such as EDRs and sandboxes. It mimics legitimate user and system behavior, injects statistical noise, randomizes execution patterns, and camouflages resource usage. Its goal is to make the malware appear benign to machine learning detectors. The techniques used to achieve this are:

- **Behavioral Mimicking:** Simulates user actions (mouse movement, fake browser use, file/network activity)
- **Noise Injection:** Performs random memory, CPU, file, and network operations to confuse behavioral classifiers
- **Timing Randomization:** Introduces irregular delays and sleep patterns to avoid timing-based anomaly detection
- **Resource Camouflage:** Adjusts CPU and memory usage to imitate normal apps (such as browsers, text editors)
- **API Call Obfuscation:** Random system API calls and pattern changes to hide malicious intent
- **Memory Access Obfuscation:** Alters access patterns and entropy to bypass ML models monitoring memory behavior

PolymorphicEngine

As part of the "Rust Engine" available in Xillen Stealer is the Polymorphic Engine. The 'PolymorphicEngine' struct implements a basic polymorphic transformation system designed for obfuscation and detection evasion. It uses predefined instruction substitutions, control-flow pattern replacements, and dead code injection to produce varied output. The `mutate_code()` method scans input bytes and replaces recognized instruction patterns with randomized alternatives, then applies control flow obfuscation and inserts non-functional code to increase variability. Additional features include string encryption via XOR and a stub-based packer.

Collectors

DevToolsCollector

Figure 5: Screenshot of Kubernetes data function.

The 'DevToolsCollector' is designed to collect sensitive data related to a wide range of developer tools and environments. This includes:

IDE configurations

- VS Code, VS Code Insiders, Visual Studio
- JetBrains: IntelliJ, PyCharm, WebStorm
- Sublime

- Atom
- Notepad++
- Eclipse

Cloud credentials and configurations

- AWS
- GCP
- Azure
- Digital Ocean
- Heroku

SSH keys

Docker & Kubernetes configurations

Git credentials

Database connection information

- HeidiSQL
- Navicat
- DBeaver
- MySQL Workbench
- pgAdmin

API keys from .env files

FTP configs

- FileZilla
- WinSCP
- Core FTP

VPN configurations

- OpenVPN
- WireGuard
- NordVPN
- ExpressVPN
- CyberGhost

Container persistence

Figure 6: Screenshot of Kubernetes inject function.

Biometric Collector

Figure 7: Screenshot of the 'BiometricCollector' function.

The 'BiometricCollector' attempts to collect biometric information from Windows systems by scanning the C:\Windows\System32\WinBioDatabase directory, which stores Windows Hello and other biometric configuration data. If accessible, it reads the contents of each file, encodes them in Base64, preparing them for later exfiltration. While the data here is typically encrypted by Windows, its collection indicates an attempt to extract sensitive biometric data.

Password Managers

The 'PasswordManagerCollector' function attempts to steal credentials stored in password managers including, OnePass, LastPass, BitWarden, Dashlane, NordPass and KeePass. However, this function is limited to Windows systems only.

SSOCollector

The 'SSOCollector' class is designed to collect authentication tokens related to SSO systems. It targets three main sources: Azure Active Directory tokens stored under TokenBroker\Cache, Kerberos tickets obtained through the klist command, and Google Cloud authentication data in user configuration folders. For each source, it checks known directories or commands, reads partial file contents, and stores the results as in a dictionary. Once again, this function is limited to Windows systems.

TOTP Collector

The 'TOTP Collector' class attempts to collect TOTP from:

- Authy Desktop by locating and reading from Authy.db SQLite databases
- Microsoft Authenticator by scanning known application data paths for stored binary files
- TOTP-related Chrome extensions by searching LevelDB files for identifiable keywords like "gauth" or "authenticator".

Each method attempts to locate relevant files, parse or partially read their contents, and store them in a dictionary under labels like authy, microsoft_auth, or chrome_extension. However, as before, this is limited to Windows, and there is no handling for encrypted tokens.

Enterprise Collector

The 'EnterpriseCollector' class is used to extract credentials related to an enterprise Windows system. It targets configuration and credential data from:

- VPN clients
 - Cisco AnyConnect, OpenVPN, Forticlient, Pulse Secure
- RDP credentials
- Corporate certificates

- Active Directory tokens
- Kerberos tickets cache

The files and directories are located based on standard environment variables with their contents read in binary mode and then encoded in Base64.

Super Extended Application Collector

The 'SuperExtendedApplication' Collector class is designed to scan an environment for 160 different applications on a Windows system. It iterates through the paths of a wide range of software categories including messaging apps, cryptocurrency wallets, password managers, development tools, enterprise tools, gaming clients, and security products. The list includes but is not limited to Teams, Slack, Mattermost, Zoom, Google Meet, MS Office, Defender, Norton, McAfee, Steam, Twitch, VMWare, to name a few.

Bypass

AppBoundBypass

This code outlines a framework for bypassing App Bound protections, Google Chrome's cookie encryption. The 'AppBoundBypass' class attempts several evasion techniques, including memory injection, dynamic-link library (DLL) hijacking, process hollowing, atom bombing, and process doppelganging to impersonate or hijack browser processes. As of the time of writing, the code contains multiple placeholders, indicating that the code is still in development.

Steganography

The 'SteganographyModule' uses steganography (hiding data within an image) to hide the stolen data, staging it for exfiltration. Multiple methods are implemented, including:

- Image steganography: LSB-based hiding
- NTFS Alternate Data Streams
- Windows Registry Keys
- Slack space: Writing into unallocated disk cluster space
- Polyglot files: Appending archive data to images
- Image metadata: Embedding data in EXIF tags
- Whitespace encoding: Hiding binary in trailing spaces of text files

Exfiltration

CloudProxy

Figure 8: Screenshot of the 'CloudProxy' class.

The CloudProxy class is designed for exfiltrating data by routing it through cloud service domains. It encodes the input data using Base64, attaches a timestamp and SHA-256 signature, and attempts to send this payload as a JSON object via HTTP POST requests to cloud URLs including AWS, GCP, and Azure, allowing the traffic to blend in. As of the time of writing, these public facing URLs do not accept POST requests, indicating that they are placeholders meant to be replaced with attacker-controlled cloud endpoints in a finalized build.

P2PEngine

Figure 9: Screenshot of the P2PEngine.

The 'P2PEngine' provides multiple methods of C2, including embedding instructions within blockchain transactions (such as Bitcoin OP_RETURN, Ethereum smart contracts), exfiltrating data via anonymizing networks like Tor and I2P, and storing payloads on IPFS (a distributed file system). It also supports domain generation algorithms (DGA) to create dynamic .onion addresses for evading detection.

After a compromise, the stealer creates both HTML and TXT reports containing the stolen data. It then sends these reports to the attacker's designated Telegram account.

Xillen Killers

Figure 10: Xillen Killers.

Xillen Stealer appears to be developed by a self-described 15-year-old "pentest specialist" "Beng/jaminButton" who creates TikTok videos showing basic exploits and open-source intelligence (OSINT) techniques. The group distributing the information stealer, known as "Xillen Killers", claims to have 3,000 members. Additionally, the group claims to have been involved in:

- Analysis of Project DDoSia, a tool reportedly used by the NoName057(16) group, revealing that rather functioning as a distributed denial-of-service (DDoS) tool, it is actually a remote access trojan (RAT) and stealer, along with the identification of involved individuals.
- Compromise of doxb.in in October 2025.
- Discovery of vulnerabilities on a Russian mods site and a Ukrainian news site

The group, which claims to be part of the Russian IT scene, use Telegram for logging, marketing, and support.

Conclusion

While some components of XillenStealer remain underdeveloped, the range of intended feature set, which includes credential harvesting, cryptocurrency theft, container targeting, and anti-analysis techniques, suggests that once fully developed it could become a sophisticated stealer. The intention

to use AI to help improve targeting in malware campaigns, even though not yet implemented, indicates how threat actors are likely to incorporate AI into future campaigns.

Edited by Ryan Traill (Analyst Content Lead)

Appendicies

Indicators of Compromise (IoCs)

395350d9c9bf32cef74357fd9cb66134 - confid.py

F3ce485b669e7c18b66d09418e979468 - stealer_v5_ultimate.py

3133fe7dc7b690264ee4f0fb6d867946 - xillen_v5.exe

<https://github.com/BengaminButton/XillenStealer>

<https://github.com/BengaminButton/XillenStealer/commit/9d9f105df4a6b20613e3a7c55379dcbf4d1ef465>

MITRE ATT&CK

ID Technique

T1059.006 - Python

T1555 - Credentials from Password Stores

T1555.003 - Credentials from Password Stores: Credentials from Web Browsers

T1555.005 - Credentials from Password Stores: Password Managers

T1649 - Steal or Forge Authentication Certificates

T1558 - Steal or Forge Kerberos Tickets

T1539 - Steal Web Session Cookie

T1552.001 - Unsecured Credentials: Credentials In Files

T1552.004 - Unsecured Credentials: Private Keys

T1552.005 - Unsecured Credentials: Cloud Instance Metadata API

T1217 - Browser Information Discovery

T1622 - Debugger Evasion

T1082 - System Information Discovery

T1497.001 - Virtualization/Sandbox Evasion: System Checks

T1115 - Clipboard Data

T1001.002 - Data Obfuscation: Steganography

T1567 - Exfiltration Over Web Service

T1657 - Financial Theft

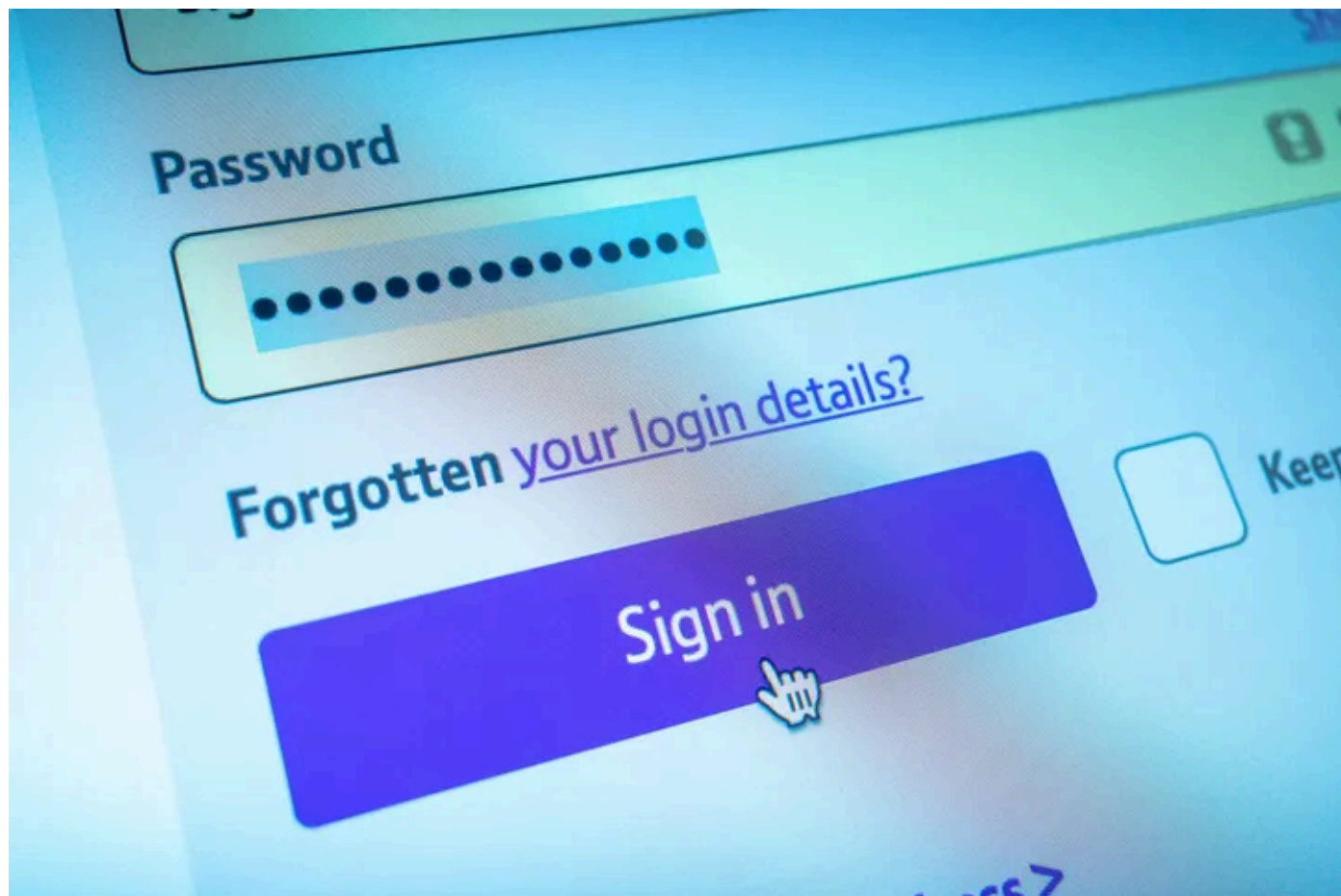
Inside the SOC

Darktrace cyber analysts are world-class experts in threat intelligence, threat hunting and incident response, and provide 24/7 SOC support to thousands of Darktrace customers around the globe.

Inside the SOC is exclusively authored by these experts, providing analysis of cyber incidents and threat trends, based on real-world experience in the field.

Share this post

Latest blogs



Unmasking Vo1d: Inside Darktrace's Botnet Detection

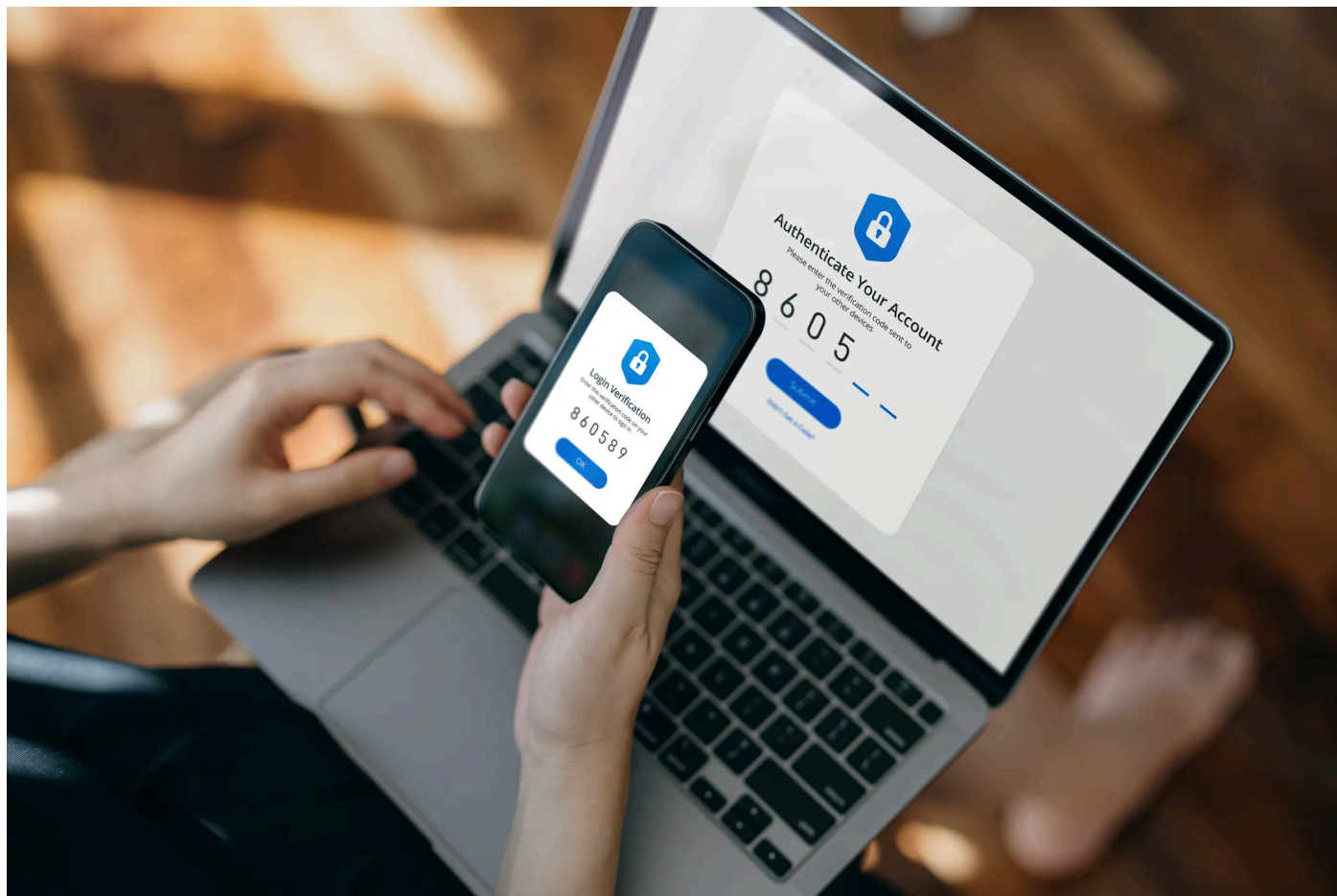
Network

•

November 12, 2025

Christina Kreza

Cyber Analyst



Managing OT Remote Access with Zero Trust Control & AI Driven Detection

OT

•

November 20, 2025

Pallavi Singh

Product Marketing Manager, OT Security & Compliance