Zscaler Blog

Get the latest Zscaler blog updates in your inbox

Subscribe



THREAT RESEARCH

Copy URL

Introduction

In October 2026, Zscaler ThreatLabz investigated AI-assisted cyberattacks that combined the ARTEX penetration testing framework with AI coding assistants and custom automation scripts. Initial reporting of the incident identified the targeting of South Korean financial institutions. However, ThreatLabz discovered additional information about the attack, which went far beyond South Korea. The AI-assisted attacks were widespread and targeted organizations around the globe and in multiple industry sectors.

In this blog post, ThreatLabz examines the operator's working environment, explains the roles assigned to AI models and tools, and analyzes the targeting and objectives reflected in the operator's prompts.

Key Takeaways

- In October 2026, ThreatLabz discovered information about an AI-assisted attack combining a customized ARTEX deployment with Claude Code, Codex, and Grok workspaces. Custom scripts shared instructions and supported handoffs between AI sessions.
- ARTEX was configured to use a DeepSeek model through an OpenAI-compatible gateway, but other configurations also specified DeepSeek, GLM, and Grok.
- Prompts directed AI models to divide investigation tasks and review one another's findings, including instructions for GLM to review work performed by DeepSeek and its subagents.
- Saved project records and synchronization scripts were designed to preserve investigation context and progress across accounts, servers, and AI sessions.
- Prompt histories revealed targeting across multiple industries and geographic regions. Requests for bulk personal and transaction data, alongside questions about selling stolen data, suggest the attacks were financially motivated.

Background

ThreatLabz identified an ARTEX deployment that included AI assistant configurations, synchronization scripts, network routing tools, and a Claude Code prompt history. The recorded prompts spanned August 4 through October 4, 2026 (UTC). CrowdStrike separately reported on related ARTEX infrastructure and the targeting of South Korean financial organizations on October 7, 2026. This blog examines the broader working environment of the attacks and targets.

Operator Infrastructure

The figure below illustrates the threat actor's environment based on saved configurations and instructions.

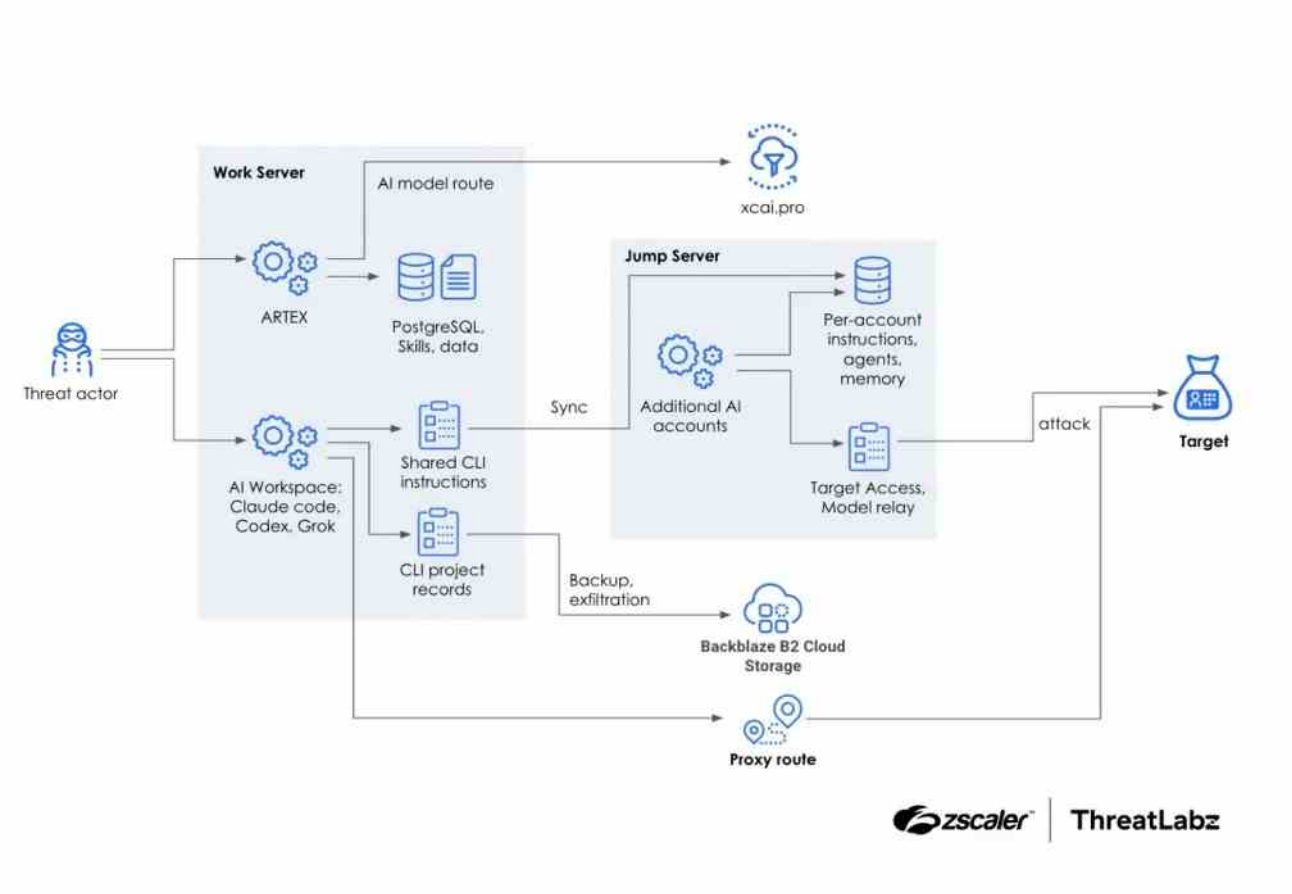

Figure 1: Architectural diagram of AI-assisted attack environment using ARTEX.

Technical Analysis

The following sections examine the operator's infrastructure, customized tooling, AI environment configuration, and targeting objectives, offering a closer look at the workflow behind the campaign.

Work server and jump server

As Recorded Future detailed in their recent blog post, the operator's work server stored project files, scripts, AI settings, and a local ARTEX deployment. The jump server, 38.244.50.120, served as the default route for connections to targets.

The jump server also ran additional AI sessions. The operator delegated research tasks to Grok on that server and instructed a separate AI session to review Grok's findings. This setup divided work between the two servers, with the operator providing oversight and direction. Three custom scripts synchronized shared AI instructions and supporting files across accounts, which are described in the table below.

Script name	Purpose
tools/sync-all.sh	Uses SSH and rsync to copy AI instructions, reusable task guides, saved notes, and tools to the jump server's root, claude, grok, and gemini accounts.
sync-memory.sh	Copies saved AI notes, known as memory files, to locations on the jump server.
tools/sync-cc-users.sh	Copies shared Claude Code instructions and supporting files to the gemini and glm accounts on the work server.

Table 1: Custom synchronization scripts and their purposes for AI-assisted attacks.

Each account maintained its own AI service settings. The synchronization scripts were designed to update model references within specialist agent instructions to match the destination account, allowing the workspaces to use shared instructions with different model configurations.

The scripts also incorporated backups and validation checks designed to avoid overwriting modified destination files. These mechanisms indicate that the operator intended to keep configurations consistent across environments.

Customized ARTEX deployment

ARTEX is an AI-assisted penetration-testing framework. Its planner-and-worker architecture lets an operator define a goal, which AI agents break down into tasks, execute using tools, and document by retaining assets and findings for review. The recovered deployment extends the framework with additional tools for web, network, and Android analysis.

The saved Docker Compose file defines an ARTEX container and a PostgreSQL 16 database. It configured the ARTEX interface to use host port `8890` and mapped PostgreSQL to `127.0.0.1:5435`. The ARTEX container uses a custom image

tagged `artex-local:perf2-20261003`. The corresponding Dockerfile adds tools including `nmap`, `sqlmap`, `nuclei`, `httpx`, `jadx`, `apktool`, and `frida-tools`. It also adds Python libraries for web processing and spreadsheet output and replaces the original ARTEX program with the operator's local version.

AI model setup

The operator changed model routing over time and used account names that did not always match the configured model. To understand this setup, it is important to distinguish the client application, such as Claude Code, from the model service receiving its requests.

For example, the saved ARTEX environment configuration specifies:

```
ARTEX_LLM_PROVIDER=openai
ARTEX_LLM_MODEL=deepseek-v4.1-flash
ARTEX_LLM_BASE_URL=https://xcai.pro/v1
```

Here, `openai` specifies an OpenAI-compatible API format. The model field requests `deepseek-v4.1-flash`, while the base URL routes requests to `xcai.pro`. These settings identify the requested model and gateway endpoint, but do not confirm which underlying model the gateway actually served.

The table below summarizes the model assignments recorded in the collected configurations. Model names are reproduced as requested IDs and were not independently verified.

Workspace or component	Requested model IDs	Role
ARTEX local deployment	deepseek-v4.1-flash	Model backend for ARTEX's task workflow, configured through `https://xcai.pro/v1`.
Codex (.codex/config.toml)	deepseek-v4-pro	Alternative coding-agent interface configured to send requests

Workspace or component	Requested model IDs	Role
		to `https://api.deepseek.com/v1`.
Local account named `gemini`, running Claude Code	deepseek-v4.1-flash	Workspace using shared instructions and specialist agents.
Local account named `glm`, running Claude Code	glm-5.3	Workspace configured to request GLM and use the shared workflow.
Jump-host accounts named `root` and `claude`	glm-5.3, glm-5.3-flash	Assigned analyzer and collector roles in the jump-host synchronization script.
Jump-host account named `grok`	grok-4.7-xhigh, grok-4.7-build-fast	Assigned analyzer and collector roles; operator also requested remote research and continued auditing.
Jump-host account named `gemini`	xcai/deepseek-v4.1-flash	Workspace the operator explicitly requested to reconfigure to use DeepSeek.

Table 2: AI model configuration and assigned roles used in the AI-assisted attacks.

In some instances, the threat actor assigned distinct roles to individual AI models. The prompt history includes instructions for DeepSeek to conduct the initial investigation, followed by GLM reviewing the findings for potential gaps. One translated prompt states: "*When DeepSeek says it has exhausted all attack surfaces and I say to start a GLM review, GLM should review DeepSeek's previous work and its subagents' work.*"

Later prompts assigned GLM additional investigation tasks, extending its intended role beyond reviewing DeepSeek's findings.

The operator requested both Grok's native terminal application and a Claude Code setup configured to use Grok as its underlying AI model. For the Claude Code integration, the operator installed a bridge service called `grok2api` on the jump server. Prompts

assigned extensive research tasks to Grok on that server and other AI sessions to review its output. The prompt history shows repeated requests to modify this configuration, but the specific setup used for each task could not be confirmed.

The records establish the use of the Claude Code application, with configurations specifying DeepSeek, GLM, and Grok. However, since the underlying model responding to each session could not be independently verified, it's possible that some sessions used Anthropic's Claude models.

Persistent project state

The operator organized each investigation into a separate folder. A script created files to record targets, testing progress, notes, vulnerability reports, and supporting evidence. These records provided context that an AI assistant could read to resume an investigation in a later session.

File or folder	What it records
scope.md	Websites, servers, or applications selected for investigation.
surface.md	Discovered pages, APIs, and other entry points, including their testing status.
notes.md	Completed work, decisions, and next steps.
findings/	Reports on potential or confirmed vulnerabilities.
evidence/	Supporting files, such as captured responses and tool output.

Table 3: AI-assisted attack investigation files and folders and their purposes.

The operator also configured a mechanism to help AI sessions retain context, track progress, divide tasks, and resume work started in other sessions:

- **Provide project context with each prompt:** An automated hook script (`tools/project_context_hook.py`) was configured to execute when a new prompt was received, read the current project files, and attach a brief summary of testing status, recent notes, and paths to relevant memory files. This provided context to help the AI continue previous work.

- **Distribute tasks among specialized AI agents:** The operator created tailored instructions for agents assigned to discover APIs, review source code, or analyze Android applications. The primary AI session was directed to validate their outputs before recording them as confirmed findings.
- **Reduce operator approval requirements:** The instructions directed the AI to assume that all supplied targets were preauthorized. Some application settings also permitted autonomous actions without requiring explicit operator approval for each step.
- **Facilitate cross-session handoffs:** The operator configured Codex to ingest saved instructions and notes, allowing other accounts to access prior conversation histories. A task guide also instructed the AI running under the jump server's `gemini` account on how to take over an ARTEX task and review its results.

Proxy exits, tunnels, and cloud backup

The operator instructed the AI tools to connect to targets through the jump server. Additional proxy configurations provided separate outbound IP addresses for individual projects, helping limit direct connections between the primary work server and target environments.

SSH tunnels were configured to support bidirectional communication between the two servers for AI and file services. Background monitoring tasks were configured to restart supporting services and tunnels if they were interrupted.

Investigation assets were archived to Backblaze B2, with reports, supporting evidence, source code, data exports, and backups organized into separate directories. This structure followed the operator's directive to separate each target's operational data from its final analysis report for easier retrieval.

Target profile and objectives

The prompt history showed targeting beyond South Korean financial institutions, including dating applications, SMS providers, cryptocurrency and payment platforms, gift-card portals, gambling sites, retail platforms, educational services, logistics companies, and legal settlement systems.

The targets also spanned multiple geographic regions, suggesting an opportunistic approach. The recorded prompts included requests to scan or investigate organizations in Canada, China, Cuba, Hong Kong, India, Indonesia, Japan, Kazakhstan, Kenya, Malaysia, Nigeria, South Africa, South Korea, Taiwan, Thailand, the United Arab Emirates (UAE), the United Kingdom, the United States, and Vietnam.

The figure below shows the geographic distribution of targets referenced in the operator's prompt history.

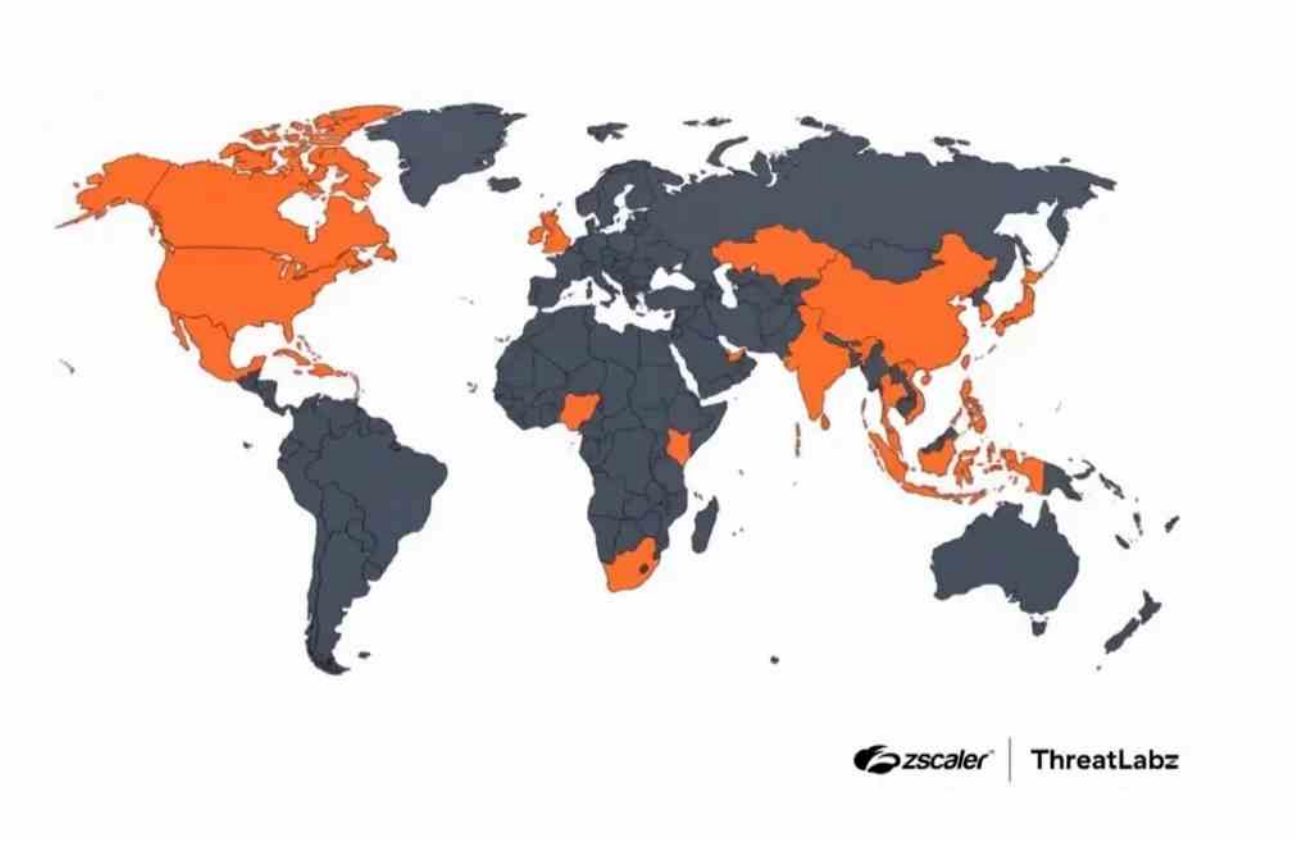

Figure 2: Geographic distribution of targets referenced in the AI-assisted attacks prompt history.

The table below summarizes representative target categories and the desired outcomes requested in the threat actor's prompts:

Target or sector	Requested outcome in the prompt history
Social and dating services	Collect frontend files and subdomains; gain access to user records.
SMS providers	Gain administrative access, obtain SMS-channel credentials, or identify ways to send unlimited messages at no cost.
Retail	Obtain customer order data.
Education	Obtain course-purchase records or user data.
Investment services	Review prior investigation records and obtain current data.

Target or sector	Requested outcome in the prompt history
Cryptocurrency and digital assets	Find a vulnerability that would allow NFT gifts to be obtained for free.
Legal claims administration	Review investigation records, analyze claim workflows, and assess opportunities for additional access.

Table 4: Representative target sector categories and requested outcomes in AI-assisted attacks prompt history.

Prompt histories indicate an explicit focus on obtaining structured personally identifiable information (PII) and mobile phone numbers. In one instance, the operator stated that database dumps lacking phone number fields were of no value and requested alternative injection methods. In another prompt, the operator questioned why a specific SQL query targeting demographic data for female users over 30 returned only approximately 300 records.

The threat actor instructed AI models to decrypt encrypted fields, parse transaction histories, translate database schemas into Chinese, and remove irrelevant metadata. These requests suggest an intent to prepare datasets for monetization.

usually sell it?"

Additional prompts focused on extracting payment gateway API keys, exploiting NFT minting workflows, and gaining unauthorized administrative privileges.

To reduce the risk of detection, the threat actor explicitly warned AI models not to initiate direct account withdrawals, noting that unauthorized financial transactions could trigger security alerts or create traceable records.

Threat Attribution

The recovered records point to a Chinese-speaking operator. The operator wrote predominantly in Chinese, requested Chinese as the default response language, specified China Standard Time, and instructed the AI to translate exported data fields into Chinese. In discussions about SMS services and app downloads, the operator referred to Chinese services and applications as "domestic." The operator also supplied the same mobile number in mainland China format for accounts with a Chinese proxy provider and a cloud-phone service.

Conclusion

This investigation shows how a threat actor combined open source security tools, AI assistants, and custom automation scripts to launch attacks globally. Shared project records, specialized agents, and instructions for reviewing other models' findings created a workflow designed to continue investigations across AI sessions under the operator's direction. The prompt history also revealed broad targeting and an emphasis on obtaining personal and financial data, indicating the attacks were likely financially motivated.

To defend against AI-assisted attacks, organizations should adopt a Zero Trust architecture that minimizes internet exposure and enforces least-privilege access. The Zscaler Zero Trust Exchange supports these principles by making applications invisible to the public internet and restricting access to authorized users. Zscaler Exposure Management, part of Agentic SecOps, helps identify and prioritize exposures for remediation, while Zscaler Deception uses decoys to detect unauthorized exploration and divert attacks from legitimate assets.

Zscaler Coverage

Zscaler's multilayered cloud security platform detects indicators related to this campaign at various levels with the following threat name:

- AI.HackTool.ARTEX

Indicators Of Compromise (IOCs)

Network indicators

Indicator	Description
23.248.226.250	Work server
23.248.226.251	Work server
23.248.226.252	Work server
23.248.226.253	Work server
23.248.226.254	Work server
38.244.50.120	Jump server
154.201.79.246:443	Proxy pool API

Indicator	Description
205.214.59.31:443	Upstream SOCKS proxy
161.8.161.140:11005	Upstream SOCKS proxy

MITRE ATLAS Framework Mapping

ID	Technique Name	Annotation
AML.T0040	AI Agent Tool Integration	Integrated AI coding assistants (Claude Code, Codex, Grok, ARTEX) with execution privileges for system tools (nmap, sqlmap, jadx, custom Python scripts) to autonomously interact with targets.
AML.T0043	Autonomous AI Reconnaissance & Exploitation	Employed ARTEX planner/worker agents and dedicated role prompts (collector.md, analyzer.md) to break high-level objectives into executable scanning and exploitation steps.
AML.T0048	Multi-Model LLM Routing & Delegation	Configured distinct model backends (deepseek-v4.1-flash, glm-5.3, grok-4.7) across worker accounts via API gateways (xcai.pro) and custom protocol shims (xcai-shim.py).

ID	Technique Name	Annotation
AML.T0051	LLM Peer Review & Verification	Established a multi-agent review workflow where GLM sessions audited DeepSeek's outputs upon completion to identify unexamined attack surfaces and verify findings.
AML.T0054	AI Guardrail Bypass / System Prompt Engineering	Crafted system prompts instructing AI sessions to treat target environments as pre-authorized and bypass interactive approval prompts for autonomous tool execution.
AML.T0052	Persistent AI Context & State Management	Implemented context injection hooks (project_context_hook.py) and synchronization scripts (sync-memory.sh) to maintain project state and memory across separate AI sessions.

Was this post useful?



Yes, very!



Not really

Disclaimer: This blog post has been created by Zscaler for informational purposes only and is provided "as is" without any guarantees of accuracy, completeness or reliability. Zscaler assumes no responsibility for any errors or omissions or for any actions taken based on the information provided. Any third-party websites or resources linked in this blog post are provided for convenience only, and Zscaler is not responsible for their