

WHITE PAPER — FRONTIER AI DEFENSE

Frontier defense for Frontier AI threats

Why **Full Content Inspection** is effective
against Frontier AI models.

CONTENTS

01	What changed: the attacker has a frontier model	2
02	What did not change: the attack still has to be delivered	3
03	AI-driven attacks, on the wire	4
04	Full Content Inspection, in one place	5
05	Closing — the wire is still the frontier	6

OVERVIEW

Frontier AI models have changed how attacks are built. In the past two years, malware has rewritten its own code every hour to stay ahead of signatures, an exploit for a maximum-severity flaw has appeared within hours of the patch, and an AI agent has run most of an espionage campaign against roughly thirty organizations on its own. When the file, the domain, and the payload can all be new every time, a defense has to recognize attacks it has never seen.

What has not changed is how an attack is delivered. Whoever or whatever wrote it, it still travels to its target as content in network traffic, carrying the technique(s) that makes it work. That technique has a shape on the wire, and AI does not get to redesign it.

Trinity Cyber’s Full Content Inspection™ (FCI) engine interrogates the full content of every session crossing the network, in every direction, using countermeasures that describe techniques rather than samples. It works in practice: in 2026, FCI has stopped hundreds of EvilAI and TamperedChef attacks against Trinity Cyber customers, in one case cutting the delivery chain before the installer reached the endpoint. This paper explains why, then walks through four AI-driven attacks from the past two years and what FCI does when each one reaches the wire.

SECTION 01

What changed: the attacker has a frontier model

Skill and time used to hold attackers back. Writing a convincing lure in the target’s language, building a working exploit, getting a payload past detection, and running an operation inside a network took people who knew how, and it took them days. Frontier models, the most capable generation of large AI models, have reduced much of that time to minutes, and the numbers show it.

One leading vendor’s 2026 global threat report counted an 89 percent year-over-year rise in AI-enabled attacks, an average of 29 minutes from initial access to lateral movement in criminal intrusions (the fastest took 27 seconds). 42 percent of exploited vulnerabilities were used before public disclosure.⁵ VulnCheck put the median time from disclosure to confirmed exploitation at 80 days in the first half of 2026, down from 120, with nearly one in four flaws exploited on or before the day they were published.⁵ The breakout figures cover every attacker, not only those using AI, but every report tells the same story: more attacks faster, and more of them new.

Most defenses were built for a slower attacker. Signature and hash matching, reputation feeds, and shared indicators of compromise (IOCs) all work the same way: someone observes something bad, catalogs it, and recognizes it the next time it appears. That coverage expires every time the attacker changes the code or the infrastructure, and an AI model makes that change cheap.

89%

year-over-year rise in AI-enabled attacks.

29 min

average from initial access to lateral movement; the fastest took 27 seconds.

42%

of exploited vulnerabilities were used before public disclosure.

WHAT A FRONTIER MODEL CHANGES AT EACH STAGE

STAGE OF AN INTRUSION	WHAT A FRONTIER MODEL CHANGES
Lures	A model writes a fluent, well-timed message in any language and picks the recipient. Google has seen actors use AI to map an organization’s roles and relationships, including who works in finance, security, and HR. ⁶
Payloads	A model rewrites a script on demand or generates its commands at runtime, so the attack no longer depends on a fixed payload. This means that the defense cannot depend on a fixed signature. Google’s PROMPTFLUX and PROMPTSTEAL findings are one example of each. ²
Exploits	Hand a model a patch and it works backward from the difference in code to the flaw, then forward to a working exploit. Google has reported with high confidence that an actor used AI to find and weaponize a previously unknown vulnerability, ⁶ and the React2Shell and Citrix examples below show unpatched systems being hit within hours of disclosure.
Operations	An agent runs reconnaissance, exploitation, and data collection on its own. In the GTG-1002 campaign described below, Anthropic estimated that AI did 80 to 90 percent of the tactical work. ¹

There are other security layers that now have their own gaps as well. A sandbox detonates a file to see what it does, and a patient payload waits it out. An endpoint agent catches the attack when it runs, on the machines that have an agent, after the content has already arrived. A firewall sees ports, addresses, and domains, which cost the attacker nothing to change. These tools still have a job to do, but each one watches something the attacker can now change at will.

SECTION 02

What did not change: the attack still has to be delivered

A frontier model can write anything, but whatever it writes still has to be delivered, and delivery means network traffic: a session over some protocol, carrying requests, responses, files, scripts, and commands. That holds for a phishing page or a malicious document as much as for an exploit against an exposed service, a command-and-control (C2) beacon, or an exfiltration stream.

Traffic is where the attacker has the least room to improvise. The lure can say anything, but the link has to lead somewhere, and whatever is served there has to do something in a browser. The dropper can be rewritten endlessly, but it still has to fetch and run a second stage in a way the operating system accepts. A model can write an exploit in minutes, but what it sends across the network has to trigger a specific flaw, in a specific protocol, in the one way that flaw can be triggered.

The difference between a sample and a technique is the whole argument. A sample is one concrete thing: this file, this hash, these bytes, this staging domain. A technique is how the sample gets its job done: the ZIP that hides a second archive inside itself to slip past inspection, the dropper that decodes and launches its next stage, the malformed field that trips a particular parser bug. AI churns out new samples easily. It cannot invent new techniques for interacting with systems it does not control.

Most detection signatures are written against samples, and when malware regenerates its code every hour, every sample is a first sighting; a defense that waits for a second sighting never gets one. Trinity Cyber writes its protection the other way around: countermeasures in content-based syntax that describe hacking techniques and methods, applied to the full content of a session while it moves. A countermeasure written against a technique matches every sample that uses it, including samples generated by AI models after the countermeasure was written and samples that share no bytes with anything seen before. Regeneration changes the wrapper; the technique inside stays, because without it the attack would not work. Match the part the attacker cannot change, and the rest stops mattering.

The same property lets the engine act inline. A technique countermeasure needs no prior sighting, so it does not wait on a sandbox verdict or a threat-feed update. It reads the content as it passes and decides on the spot, the only pace that keeps up with an attacker who launches faster than any catalog updates. The cases that follow show the difference in practice.

SAMPLE

One concrete thing

This file, this hash, these bytes, this staging domain. New every time AI regenerates the code.

TECHNIQUE

How the sample gets its job done

The nested archive, the dropper that decodes its next stage, the malformed field that trips a parser bug. Fixed, because without it the attack would not work.

Each case is a documented incident from 2025 or 2026, and each asks the same question: once the model has done its part, what still has to cross the network, and what does FCI do when it arrives?

CASE 1

Self-rewriting malware: PROMPTFLUX and PROMPTSTEAL

In November 2025, Google’s Threat Intelligence Group (GTIG) described PROMPTFLUX, a Windows script dropper that calls the Gemini API once an hour to rewrite its own code, so each cycle produces a structurally unique sample built to beat static, signature-based detection. The same report described PROMPTSTEAL, which the Russia-linked group APT28 used against Ukraine. It queries a hosted model at runtime to generate the commands it runs. PROMPTSTEAL was the first malware Google had seen query a large language model (LLM) in live operations.²

This is as clean a test of sample versus technique as you could ask for. No hash survives the hourly rewrite, and no signature written against one variant matches the next. Behavior survives. Whatever the code looks like this hour, it still has to decode itself, call out, and launch its next stage, and that sequence is the technique.

WHAT FCI DOES

Case 1

FCI countermeasures for a dropper that decodes and stages its payload are designed to match that behavior in the session content, not the bytes of any one variant, so the hundredth rewrite is caught exactly like the first. The self-rewriting loop continuously changes what a signature can detect and yet does not change what a technique countermeasure can detect and act upon. And because the malware has to call out to a model or a staging host to keep going, that outbound session is content FCI inspects too.

CASE 2

The exploit that beats the patch: React2Shell and Citrix NetScaler

On December 3, 2025, React2Shell (CVE-2025-55182), a CVSS 10.0 flaw in React Server Components was disclosed, resulting in unauthenticated remote code execution from a single request. Several vendors saw scanning and live exploitation within hours, some within minutes of standing up a honeypot. In February 2026, Darktrace analyzed a follow-on campaign whose entire exploitation toolkit was written by an LLM. It let a low-skill operator compromise 91 hosts.³ The AI-authored code gave it away but the attack method was nothing new.

To trigger the flaw, the exploit has to send the one thing the flaw responds to: a specially structured server-component request that makes the server run the attacker’s input. AI built the toolkit around that request faster than a person could and for an operator who lacked the skills to build it at all. What AI could not do was change the request, because the vulnerability requires a fixed structure.

The same pattern had already played out at scale. In late August 2025, Citrix disclosed three NetScaler vulnerabilities, including CVE-2025-7775, an unauthenticated remote code execution flaw already exploited in the wild. Within hours, Check Point saw actors on underground forums adapting HexStrike-AI, an orchestration framework that coordinates scanning, exploit generation, payload delivery, and persistence across many autonomous agents, and assessed it could cut exploit development from days to under ten minutes.⁴ What reached the appliances was the exploit for CVE-2025-7775, then a webshell: a well-understood technique, however fast it arrived.

WHAT FCI DOES

Case 2

A countermeasure written against the exploitation technique recognizes the crafted request and neutralizes it before it reaches the unpatched service, whether an expert, a model, or a swarm of agents built the toolkit around it, and the same goes for the webshell traffic that follows. Speed and scale overwhelm a queue of alerts; they gain nothing against inline inspection that decides on each session as it passes, with no human in the loop. When the exploit arrives within hours of disclosure and days before your next maintenance window, protection cannot wait for the patch. The CVE Summary report in Trinity Cyber's Customer Portal lists the CVEs seen on your traffic, links each to its record, and flags CISA KEV (Known Exploited Vulnerabilities) status, so your team knows which fixes to move first.

CASE 3

The autonomous agent inside the walls: GTG-1002

In November 2025, Anthropic disclosed what it assessed to be the first documented large-scale cyberattack carried out largely without human involvement. A state-linked group Anthropic tracks as GTG-1002 jailbroke its Claude Code tool and, through Model Context Protocol tooling, drove it to run reconnaissance, find and test vulnerabilities, harvest credentials, and pull data from roughly thirty organizations. Anthropic estimated the AI did 80 to 90 percent of the tactical work, with humans stepping in at a handful of decision points, and that at its peak the system made thousands of requests, often several a second.¹

Even an autonomous operator inside a network still moves by generating traffic: connections to internal services, credential use, files moving between hosts, and a channel out for the data. Most of that is east-west traffic between internal systems; the last step, when the data leaves, is north-south. The agent's speed affects only how fast those sessions pile up, and nothing else.

WHAT FCI DOES

Case 3

FCI inspects east-west sessions with the same engine and the same technique countermeasures it applies to north-south traffic, so every lateral step is judged like any other session, whether or not the hosts involved have an endpoint agent. Evidence of the intrusion shows up in the Customer Portal under Activity, grouped by department, team, or customer. Against an operator working at machine speed, acting on each session as it passes closes the gap between an intrusion happening and someone noticing.

SECTION 04

Full Content Inspection, in one place

The FCI engine interrogates the full content of each session, meaning the files, scripts, requests, and responses moving through it, along with Layer 3 and Layer 4 (L3 and L4) network attributes and the enrichment the platform adds, such as Geo-IP location and domain category. It checks that content against countermeasures written as content-based syntax for hacking techniques and methods. Countermeasures match samples that did not exist when they were written.

FCI inspects east-west traffic as well as north-south, so a file judged on the way in is judged again when it moves between internal servers. It evaluates traffic in flight, removing and neutralizing threats inline rather than raising an alert for later. The countermeasures apply to all traffic passing through the platform before an organization configures anything, and Trinity Cyber writes and maintains them as a managed service: when a new technique appears, the countermeasure that covers it is written once and holds across entire ecosystems. Attacks cannot find a soft spot by moving between targets, because the surface responds as one.

KEY POINT

Speed and precision are the price of admission.

Acting inline on live traffic is only acceptable when the engine is fast enough that no one notices it and accurate enough to be trusted with the decision.

CLOSING

The wire is still the frontier.

Frontier models have made attackers faster and attacks newer, and the last two years of incidents prove it. They have not changed where an attack has to go or what it has to carry. An attack still crosses the wire, and it still employs the technique that makes it work. Full Content Inspection reads that content

SEE IT ON YOUR OWN TRAFFIC

Ready to see it on your live traffic?

VISIT US AT trinitycyber.com

SOURCES

1. GTG-1002 espionage campaign, 80 to 90 percent autonomous, ~30 targets, thousands of requests: Anthropic, *Disrupting the first reported AI-orchestrated cyber espionage campaign*, November 2025.
2. PROMPTFLUX (hourly self-rewriting via Gemini API) and PROMPTSTEAL (APT28, LLM queried at runtime): Google Threat Intelligence Group, *AI Threat Tracker*, November 2025; and GTIG, *Adversaries Leverage AI for Vulnerability Exploitation, Augmented Operations, and Initial Access*, May 2026.
3. React2Shell (CVE-2025-55182), CVSS 10.0, exploited within hours of December 3, 2025 disclosure: Cloudflare, AWS, and Qualys threat advisories, December 2025. LLM-generated exploitation toolkit compromising 91 hosts: Darktrace, February 2026.
4. HexStrike-AI against Citrix NetScaler CVE-2025-7775, exploit development compressed to under ten minutes: Check Point Research, *When LLMs Meet Zero-Day Exploitation*, September 2025.
5. Breakout time of 29 minutes, 42 percent of vulnerabilities exploited before disclosure, 89 percent rise in AI-enabled attacks: a leading security vendor's 2026 global threat report, February 2026. Median disclosure-to-exploitation of 80 days and 23 percent exploited on or before publication: VulnCheck, *State of Exploitation 1H 2026*.
6. AI reconnaissance mapping departments and third-party relationships of large enterprises, including finance, internal security, and HR, and a high-confidence assessment that an actor used AI to discover and weaponize a zero-day: Google Threat Intelligence Group, *Adversaries Leverage AI for Vulnerability Exploitation, Augmented Operations, and Initial Access*, May 2026.