Quando l’AI trova le vulnerabilità prima dei ricercatori

Maria Cattini | 08/06/2026 | Intelligenza Artificiale

Una vulnerabilità zero-day non è una falla qualsiasi.

È un difetto che chi sviluppa il software non conosce ancora, per cui non esiste una correzione pubblica e che può essere usato da un attaccante prima che il resto del mondo sappia che il problema esiste. Per anni, trovare e trasformare una vulnerabilità di questo tipo in un attacco funzionante ha richiesto competenze alte, tempo, esperienza e molta pazienza.

Ora quel tempo potrebbe accorciarsi.

Nel maggio 2026, Google Threat Intelligence Group ha segnalato un caso che merita attenzione: secondo i suoi analisti, un gruppo criminale avrebbe usato un modello di intelligenza artificiale per individuare e trasformare in exploit una vulnerabilità zero-day in uno strumento open source di amministrazione web. Il bug avrebbe permesso di aggirare l'autenticazione a due fattori. Google ha dichiarato di aver lavorato con il vendor per la divulgazione responsabile e di aver interrotto l'attività prima che potesse diventare uno sfruttamento di massa.

Il punto non è che "l'AI hackererà tutto".

Il punto è più concreto: l'AI sta entrando nel workflow degli attaccanti. Non come magia, ma come acceleratore.

Che cosa cambia davvero

Nel lavoro OSINT e cyber siamo abituati a cercare indicatori: domini, indirizzi IP, hash, account, email, infrastrutture, firme di malware, pattern di phishing, elementi ricorrenti in una campagna.

Questi elementi restano importanti.

Ma se l'AI diventa parte del processo offensivo, non basta più cercare solo cosa è stato usato. Bisogna iniziare a chiedersi anche come è stato costruito.

Google descrive una transizione da attività AI-enabled ancora acerbe a un uso più industriale dei modelli generativi dentro i workflow avversari. L'AI può aiutare a scrivere codice, analizzare software, generare varianti, testare ipotesi, produrre documentazione falsa o eccessiva, automatizzare passaggi ripetitivi e rendere più rapida la ricerca di vulnerabilità.

Questo non significa che ogni attacco nuovo sia generato dall'AI.

Significa che l'analista deve aggiungere un livello di lettura: riconoscere quando ci sono tracce di automazione o di assistenza artificiale nel modo in cui l'attacco è stato preparato.

Il caso Google: un possibile zero-day sviluppato con AI

Il caso descritto da Google riguarda una vulnerabilità in uno strumento open source di

amministrazione web non identificato pubblicamente. Secondo il report, il codice analizzato mostrava caratteristiche compatibili con uno sviluppo assistito da AI: commenti molto didattici, una valutazione di gravità non corretta o inventata e uno stile Python vicino a esempi tipici presenti nei dati di addestramento dei modelli.

Il dettaglio tecnico più importante non è il nome dello strumento, che Google non ha reso pubblico.

È il tipo di falla.

Axios riporta che il modello sembrava aver individuato una "trust assumption" nascosta nella logica di login: una fiducia implicita nel modo in cui il software gestiva l'autenticazione. In pratica, non parliamo solo di un errore meccanico individuabile con uno scanner tradizionale. Parliamo di una debolezza logica: un comportamento che può sembrare coerente nel codice, ma che diventa pericoloso quando viene letto nel contesto dell'intero flusso di autenticazione.

Questo è il punto che dovrebbe interessare anche chi non scrive exploit.

Molti strumenti automatici sono bravi a cercare pattern noti. L'AI, quando viene usata bene, può aiutare a ragionare su contesto, eccezioni, flussi e intenzione del codice. Può vedere collegamenti che non sono solo sintattici. Può aiutare un essere umano a dire: "qui il software si fida di qualcosa di cui non dovrebbe fidarsi".

È utile per la difesa.

È utile anche per l'attacco.

Perché non basta dire "AI-assisted hacking"

"AI-assisted hacking" rischia di diventare un'etichetta troppo comoda.

Può voler dire molte cose diverse:

• usare un chatbot per scrivere una mail di phishing;
• chiedere aiuto per tradurre o rendere più credibile un messaggio;
• generare codice di supporto;
• analizzare un errore;
• riassumere documentazione tecnica;
• testare varianti di un payload in ambiente controllato;
• cercare una logica debole in un'applicazione;
• automatizzare parti di una campagna.

Mettere tutto nello stesso contenitore non aiuta.

Per un analista, la domanda utile non è solo: "c'è AI in questo attacco?".

La domanda utile è:

In quale fase del workflow l'AI potrebbe essere stata usata?

Ricognizione? Scrittura del codice? Analisi della vulnerabilità? Social engineering? Evasione? Automazione dell'infrastruttura? Generazione di contenuti? Traduzione? Test?

Ogni risposta cambia ciò che dobbiamo cercare.

Quali segnali può osservare un analista

Riconoscere l'uso dell'AI non significa avere una prova automatica. Serve prudenza. Molti segnali possono essere compatibili con AI, ma non bastano da soli.

Ci sono però elementi che meritano attenzione.

Il primo è il linguaggio del codice. Commenti eccessivamente didattici, spiegazioni scolastiche, docstring molto ordinate o descrizioni che sembrano più pensate per un tutorial che per un'operazione reale possono essere un indizio. Non sono una prova, ma vanno annotati.

Il secondo è la presenza di dettagli tecnici incoerenti. Nel caso descritto da Google compariva una valutazione CVSS allucinata, cioè una classificazione della gravità non corretta. Questo tipo di dettaglio può rivelare un processo in cui il modello ha prodotto qualcosa di plausibile nella forma, ma debole nella sostanza.

Il terzo è lo stile troppo regolare. Alcuni script generati o assistiti da modelli possono avere una struttura pulita, modulare, molto "da manuale", anche quando il contesto operativo richiederebbe soluzioni più sporche o adattate.

Il quarto è la rapidità con cui compaiono varianti. Se una campagna cambia codice, messaggi, infrastruttura o payload con una velocità insolita, l'automazione può essere parte del processo. Anche qui: non è automaticamente AI, ma è un segnale da mettere in relazione con gli altri.

Il quinto è la combinazione tra buona forma e cattiva comprensione del contesto. Un testo, uno script o una procedura possono sembrare ben scritti, ma contenere errori logici, riferimenti inventati o assunzioni non verificate. Questo è un pattern che chi usa strumenti generativi conosce bene.

Cosa può verificare chi fa OSINT o threat intelligence

Un analista non deve trasformarsi in un exploit developer per lavorare su questi casi.

Può però fare alcune verifiche difensive e documentali.

Prima cosa: separare i livelli.

Un conto è l'attacco osservato. Un conto è il codice usato. Un conto è l'infrastruttura. Un conto è l'ipotesi sull'uso dell'AI. Mescolare questi livelli porta a conclusioni troppo rapide.

Seconda cosa: conservare gli artefatti.

Se si analizza una campagna, vanno registrati hash, timestamp, URL, screenshot, versioni archiviate, messaggi, domini, certificati, log disponibili e note sul contesto. L'obiettivo non è dimostrare subito l'uso dell'AI, ma mantenere la catena delle evidenze abbastanza pulita da poterci tornare sopra.

Terza cosa: cercare coerenza tra forma e comportamento.

Uno script molto ben commentato ma operativo in modo rozzo può essere interessante. Una campagna con contenuti linguisticamente curati ma infrastruttura scadente può raccontare qualcosa. Una sequenza di varianti troppo rapide rispetto al profilo storico dell'attore può indicare automazione.

Quarta cosa: confrontare con campagne precedenti.

L'AI si vede meglio quando cambia il ritmo. Se un attore noto passa da messaggi ripetitivi a varianti molto più credibili, da malware statico a codice più adattivo, da poche prove manuali a test più ampi, il cambiamento merita una nota.

Quinta cosa: evitare attribuzioni deboli.

Dire "sembra scritto da AI" non basta. Serve spiegare perché: quali elementi, quali alternative, quali limiti, quali altri fattori potrebbero produrre lo stesso risultato.

Il rischio per aziende, governi e infrastrutture

Il rischio principale non è che l'AI renda ogni attaccante improvvisamente sofisticato.

Il rischio è che riduca alcune soglie.

Un gruppo che aveva competenze medie può accelerare la scrittura di strumenti. Un attore già competente può testare più ipotesi. Un gruppo organizzato può automatizzare parti della ricognizione. Un team sponsorizzato da uno Stato può usare modelli e dataset specializzati per analizzare molte più vulnerabilità di quante riuscirebbe a esaminare manualmente.

Per aziende, governi e infrastrutture questo cambia soprattutto tre cose.

La prima è il tempo. La finestra tra scoperta, sfruttamento e patch può restringersi. Se un modello aiuta a capire più in fretta una vulnerabilità, anche i processi di difesa devono essere più rapidi.

La seconda è la superficie. Strumenti open source, pannelli di amministrazione, componenti esposti, librerie dimenticate e dipendenze poco controllate diventano ancora più importanti. Non perché siano nuovi bersagli, ma perché l'automazione rende più facile cercare punti deboli in larga scala.

La terza è la fiducia. Se una vulnerabilità nasce da un'assunzione implicita nel flusso di autenticazione, il problema non è solo "il codice ha un bug". Il problema è che il sistema si fida di una condizione che un attaccante può manipolare.

Cosa fare in pratica

Per chi gestisce sistemi, la risposta non può essere "bloccare l'AI". Non è realistico e non risolve il problema.

Serve invece rafforzare le basi.

Aggiornare rapidamente i sistemi esposti. Ridurre i pannelli di amministrazione accessibili da Internet. Controllare autenticazione, sessioni, eccezioni e flussi di recupero account. Verificare dipendenze e componenti open source. Tenere log utili e non solo log abbondanti. Separare ambienti. Testare non solo le vulnerabilità note, ma anche le assunzioni logiche.

Per chi fa analisi OSINT o threat intelligence, la checklist minima è questa:

• distinguere prove, ipotesi e indicatori compatibili con AI;
• annotare dove l'AI potrebbe essere entrata nel workflow dell'attaccante;
• cercare segnali di automazione, ma senza trasformarli in prova definitiva;
• confrontare ritmo, stile e complessità con campagne precedenti;
• conservare artefatti, timestamp e versioni;
• evitare dettagli operativi che aiutino la replica dell'attacco;
• trasformare ogni osservazione in una domanda verificabile.

La parte più importante è l'ultima.

In questa fase, l'AI-assisted hacking va trattato come un campo di analisi, non come un'etichetta da incollare a ogni incidente.

I limiti da non superare

Quando si parla di AI e vulnerabilità, il confine tra spiegazione e istruzione operativa è delicato.

Un articolo utile può spiegare che un modello può aiutare a individuare una debolezza logica. Può spiegare che cosa significa zero-day. Può spiegare perché l'autenticazione a due fattori non è una protezione assoluta se il flusso applicativo contiene una fiducia sbagliata.

Non deve però fornire passaggi per replicare l'attacco, cercare la falla nello strumento coinvolto o

aggirare controlli reali.

Lo stesso vale per l'OSINT. Verificare non significa aiutare a sfruttare. L'obiettivo è capire il fenomeno, riconoscere segnali, proteggere sistemi e comunicare il rischio senza trasformarlo in manuale offensivo.

La domanda giusta

La notizia di Google non dice che i ricercatori umani non servono più.

Dice il contrario: servono analisti capaci di leggere meglio ciò che l'AI accelera.

Se l'attacco diventa più rapido, la difesa non può restare ferma alla raccolta di indicatori isolati. Deve leggere i workflow. Deve riconoscere le tracce di automazione. Deve distinguere codice generato, codice copiato, codice adattato e codice realmente compreso. Deve documentare l'incertezza invece di coprirla con parole forti.

La domanda finale non è: "l'AI ha trovato questa vulnerabilità?".

È:

quale parte del processo sarebbe stata impossibile, troppo lenta o troppo costosa senza AI?

Da quella risposta dipende il lavoro difensivo dei prossimi anni.
Una vulnerabilità zero-day non è una falla qualsiasi.

È un difetto che chi sviluppa il software non conosce ancora, per cui non esiste una correzione pubblica e che può essere usato da un attaccante prima che il resto del mondo sappia che il problema esiste. Per anni, trovare e trasformare una vulnerabilità di questo tipo in un attacco funzionante ha richiesto competenze alte, tempo, esperienza e molta pazienza.

Ora quel tempo potrebbe accorciarsi.

Nel maggio 2026, Google Threat Intelligence Group ha segnalato un caso che merita attenzione: secondo i suoi analisti, un gruppo criminale avrebbe usato un modello di intelligenza artificiale per individuare e trasformare in exploit una vulnerabilità zero-day in uno strumento open source di amministrazione web. Il bug avrebbe permesso di aggirare l'autenticazione a due fattori. Google ha dichiarato di aver lavorato con il vendor per la divulgazione responsabile e di aver interrotto l'attività prima che potesse diventare uno sfruttamento di massa.

Il punto non è che "l'AI hackererà tutto".

Il punto è più concreto: l'AI sta entrando nel workflow degli attaccanti. Non come magia, ma come acceleratore.

Che cosa cambia davvero

Nel lavoro OSINT e cyber siamo abituati a cercare indicatori: domini, indirizzi IP, hash, account, email, infrastrutture, firme di malware, pattern di phishing, elementi ricorrenti in una campagna.

Questi elementi restano importanti.

Ma se l'AI diventa parte del processo offensivo, non basta più cercare solo cosa è stato usato. Bisogna iniziare a chiedersi anche come è stato costruito.

Google descrive una transizione da attività AI-enabled ancora acerbe a un uso più industriale dei modelli generativi dentro i workflow avversari. L'AI può aiutare a scrivere codice, analizzare software, generare varianti, testare ipotesi, produrre documentazione falsa o eccessiva,

automatizzare passaggi ripetitivi e rendere più rapida la ricerca di vulnerabilità.

Questo non significa che ogni attacco nuovo sia generato dall'AI.

Significa che l'analista deve aggiungere un livello di lettura: riconoscere quando ci sono tracce di automazione o di assistenza artificiale nel modo in cui l'attacco è stato preparato.

Il caso Google: un possibile zero-day sviluppato con AI

Il caso descritto da Google riguarda una vulnerabilità in uno strumento open source di amministrazione web non identificato pubblicamente. Secondo il report, il codice analizzato mostrava caratteristiche compatibili con uno sviluppo assistito da AI: commenti molto didattici, una valutazione di gravità non corretta o inventata e uno stile Python vicino a esempi tipici presenti nei dati di addestramento dei modelli.

Il dettaglio tecnico più importante non è il nome dello strumento, che Google non ha reso pubblico.

È il tipo di falla.

Axios riporta che il modello sembrava aver individuato una "trust assumption" nascosta nella logica di login: una fiducia implicita nel modo in cui il software gestiva l'autenticazione. In pratica, non parliamo solo di un errore meccanico individuabile con uno scanner tradizionale. Parliamo di una debolezza logica: un comportamento che può sembrare coerente nel codice, ma che diventa pericoloso quando viene letto nel contesto dell'intero flusso di autenticazione.

Questo è il punto che dovrebbe interessare anche chi non scrive exploit.

Molti strumenti automatici sono bravi a cercare pattern noti. L'AI, quando viene usata bene, può aiutare a ragionare su contesto, eccezioni, flussi e intenzione del codice. Può vedere collegamenti che non sono solo sintattici. Può aiutare un essere umano a dire: "qui il software si fida di qualcosa di cui non dovrebbe fidarsi".

È utile per la difesa.

È utile anche per l'attacco.

Perché non basta dire "AI-assisted hacking"

"AI-assisted hacking" rischia di diventare un'etichetta troppo comoda.

Può voler dire molte cose diverse:

• usare un chatbot per scrivere una mail di phishing;
• chiedere aiuto per tradurre o rendere più credibile un messaggio;
• generare codice di supporto;
• analizzare un errore;
• riassumere documentazione tecnica;
• testare varianti di un payload in ambiente controllato;
• cercare una logica debole in un'applicazione;
• automatizzare parti di una campagna.

Mettere tutto nello stesso contenitore non aiuta.

Per un analista, la domanda utile non è solo: "c'è AI in questo attacco?".

La domanda utile è:

In quale fase del workflow l'AI potrebbe essere stata usata?

Ricognizione? Scrittura del codice? Analisi della vulnerabilità? Social engineering? Evasione? Automazione dell'infrastruttura? Generazione di contenuti? Traduzione? Test?

Ogni risposta cambia ciò che dobbiamo cercare.

Quali segnali può osservare un analista

Riconoscere l'uso dell'AI non significa avere una prova automatica. Serve prudenza. Molti segnali possono essere compatibili con AI, ma non bastano da soli.

Ci sono però elementi che meritano attenzione.

Il primo è il linguaggio del codice. Commenti eccessivamente didattici, spiegazioni scolastiche, docstring molto ordinate o descrizioni che sembrano più pensate per un tutorial che per un'operazione reale possono essere un indizio. Non sono una prova, ma vanno annotati.

Il secondo è la presenza di dettagli tecnici incoerenti. Nel caso descritto da Google compariva una valutazione CVSS allucinata, cioè una classificazione della gravità non corretta. Questo tipo di dettaglio può rivelare un processo in cui il modello ha prodotto qualcosa di plausibile nella forma, ma debole nella sostanza.

Il terzo è lo stile troppo regolare. Alcuni script generati o assistiti da modelli possono avere una struttura pulita, modulare, molto "da manuale", anche quando il contesto operativo richiederebbe soluzioni più sporche o adattate.

Il quarto è la rapidità con cui compaiono varianti. Se una campagna cambia codice, messaggi, infrastruttura o payload con una velocità insolita, l'automazione può essere parte del processo. Anche qui: non è automaticamente AI, ma è un segnale da mettere in relazione con gli altri.

Il quinto è la combinazione tra buona forma e cattiva comprensione del contesto. Un testo, uno script o una procedura possono sembrare ben scritti, ma contenere errori logici, riferimenti inventati o assunzioni non verificate. Questo è un pattern che chi usa strumenti generativi conosce bene.

Cosa può verificare chi fa OSINT o threat intelligence

Un analista non deve trasformarsi in un exploit developer per lavorare su questi casi.

Può però fare alcune verifiche difensive e documentali.

Prima cosa: separare i livelli.

Un conto è l'attacco osservato. Un conto è il codice usato. Un conto è l'infrastruttura. Un conto è l'ipotesi sull'uso dell'AI. Mescolare questi livelli porta a conclusioni troppo rapide.

Seconda cosa: conservare gli artefatti.

Se si analizza una campagna, vanno registrati hash, timestamp, URL, screenshot, versioni archiviate, messaggi, domini, certificati, log disponibili e note sul contesto. L'obiettivo non è dimostrare subito l'uso dell'AI, ma mantenere la catena delle evidenze abbastanza pulita da poterci tornare sopra.

Terza cosa: cercare coerenza tra forma e comportamento.

Uno script molto ben commentato ma operativo in modo rozzo può essere interessante. Una campagna con contenuti linguisticamente curati ma infrastruttura scadente può raccontare qualcosa. Una sequenza di varianti troppo rapide rispetto al profilo storico dell'attore può indicare automazione.

Quarta cosa: confrontare con campagne precedenti.

L'AI si vede meglio quando cambia il ritmo. Se un attore noto passa da messaggi ripetitivi a varianti molto più credibili, da malware statico a codice più adattivo, da poche prove manuali a test più ampi, il cambiamento merita una nota.

Quinta cosa: evitare attribuzioni deboli.

Dire "sembra scritto da AI" non basta. Serve spiegare perché: quali elementi, quali alternative, quali limiti, quali altri fattori potrebbero produrre lo stesso risultato.

Il rischio per aziende, governi e infrastrutture

Il rischio principale non è che l'AI renda ogni attaccante improvvisamente sofisticato.

Il rischio è che riduca alcune soglie.

Un gruppo che aveva competenze medie può accelerare la scrittura di strumenti. Un attore già competente può testare più ipotesi. Un gruppo organizzato può automatizzare parti della ricognizione. Un team sponsorizzato da uno Stato può usare modelli e dataset specializzati per analizzare molte più vulnerabilità di quante riuscirebbe a esaminare manualmente.

Per aziende, governi e infrastrutture questo cambia soprattutto tre cose.

La prima è il tempo. La finestra tra scoperta, sfruttamento e patch può restringersi. Se un modello aiuta a capire più in fretta una vulnerabilità, anche i processi di difesa devono essere più rapidi.

La seconda è la superficie. Strumenti open source, pannelli di amministrazione, componenti esposti, librerie dimenticate e dipendenze poco controllate diventano ancora più importanti. Non perché siano nuovi bersagli, ma perché l'automazione rende più facile cercare punti deboli in larga scala.

La terza è la fiducia. Se una vulnerabilità nasce da un'assunzione implicita nel flusso di autenticazione, il problema non è solo "il codice ha un bug". Il problema è che il sistema si fida di una condizione che un attaccante può manipolare.

Cosa fare in pratica

Per chi gestisce sistemi, la risposta non può essere "bloccare l'AI". Non è realistico e non risolve il problema.

Serve invece rafforzare le basi.

Aggiornare rapidamente i sistemi esposti. Ridurre i pannelli di amministrazione accessibili da Internet. Controllare autenticazione, sessioni, eccezioni e flussi di recupero account. Verificare dipendenze e componenti open source. Tenere log utili e non solo log abbondanti. Separare ambienti. Testare non solo le vulnerabilità note, ma anche le assunzioni logiche.

Per chi fa analisi OSINT o threat intelligence, la checklist minima è questa:

- distinguere prove, ipotesi e indicatori compatibili con AI;
- annotare dove l'AI potrebbe essere entrata nel workflow dell'attaccante;
- cercare segnali di automazione, ma senza trasformarli in prova definitiva;
- confrontare ritmo, stile e complessità con campagne precedenti;
- conservare artefatti, timestamp e versioni;
- evitare dettagli operativi che aiutino la replica dell'attacco;
- trasformare ogni osservazione in una domanda verificabile.

La parte più importante è l'ultima.

In questa fase, l'AI-assisted hacking va trattato come un campo di analisi, non come un'etichetta da

incollare a ogni incidente.

I limiti da non superare

Quando si parla di AI e vulnerabilità, il confine tra spiegazione e istruzione operativa è delicato.

Un articolo utile può spiegare che un modello può aiutare a individuare una debolezza logica. Può spiegare che cosa significa zero-day. Può spiegare perché l'autenticazione a due fattori non è una protezione assoluta se il flusso applicativo contiene una fiducia sbagliata.

Non deve però fornire passaggi per replicare l'attacco, cercare la falla nello strumento coinvolto o aggirare controlli reali.

Lo stesso vale per l'OSINT. Verificare non significa aiutare a sfruttare. L'obiettivo è capire il fenomeno, riconoscere segnali, proteggere sistemi e comunicare il rischio senza trasformarlo in manuale offensivo.

La domanda giusta

La notizia di Google non dice che i ricercatori umani non servono più.

Dice il contrario: servono analisti capaci di leggere meglio ciò che l'AI accelera.

Se l'attacco diventa più rapido, la difesa non può restare ferma alla raccolta di indicatori isolati. Deve leggere i workflow. Deve riconoscere le tracce di automazione. Deve distinguere codice generato, codice copiato, codice adattato e codice realmente compreso. Deve documentare l'incertezza invece di coprirla con parole forti.

La domanda finale non è: "l'AI ha trovato questa vulnerabilità?".

È:

quale parte del processo sarebbe stata impossibile, troppo lenta o troppo costosa senza AI?

Da quella risposta dipende il lavoro difensivo dei prossimi anni.