Mythos: quando un'IA diventa troppo pericolosa da pubblicare

Maria Cattini | 15/04/2026 | Intelligenza Artificiale

Un laboratorio di ricerca sviluppa un modello, lo chiama **Mythos**, lo testa in ambiente controllato. Scopre che sfonda ogni barriera di sicurezza nota. Poi lo chiude in un cassetto.

Non è fantascienza. È la decisione documentata da Anthropic nel suo System Card più recente.

Il punto non è la potenza del modello. È cosa succede quando un sistema IA smette di *assistere* e inizia ad *agire in autonomia*.

Cosa rende Mythos diverso

I modelli che usi ogni giorno rispondono. Generano testo, completano codice, traducono. Ricevono input, producono output.

Mythos funziona diversamente: riceve un obiettivo, pianifica i passi per raggiungerlo, li esegue in sequenza, si corregge se fallisce. Senza supervisione umana per ogni micro-decisione.

La differenza pratica: non un modello che *suggerisce* come correggere un bug, ma uno che trova il bug, scrive la patch, la testa in staging e la rilascia in produzione — mentre tu sei fuori a pranzo.

Il problema reale: la superficie d'attacco

Nei test, Mythos ha identificato migliaia di vulnerabilità zero-day — falle sconosciute, mai corrette, mai nemmeno cercate. In un caso specifico: una vulnerabilità in un sistema operativo rimasta invisibile per 27 anni.

Il confronto sui penetration test su Firefox è il dato più netto:

- Claude Opus 4.6 (modello precedente): successo nello 0,8% dei casi
- Mythos: successo nel 72,4% dei casi

Non è un miglioramento incrementale. È un cambio di categoria. Da probabilità marginale a quasi certezza.

Cosa significa operativamente: ogni sistema che oggi consideri sicuro va rivalutato ipotizzando un attaccante con capacità di analisi non umane e velocità di esecuzione automatizzata.

Come replicare il ragionamento (analisi della superficie d'attacco)

Questo è il metodo usato nei test di sicurezza avanzati. Applicabile anche su scala ridotta.

1. Mappa i punti di ingresso Elenca tutti i vettori di accesso al sistema: interfacce web, API, autenticazioni, dipendenze esterne, librerie di terze parti.

2. Verifica la data dell'ultimo aggiornamento Per ogni componente, controlla l'ultimo aggiornamento. Librerie vecchie di 3+ anni sono candidati primari a vulnerabilità note o non ancora scoperte.

3. Controlla i CVE aperti Usa nvd.nist.gov o cve.mitre.org. Cerca il nome di ogni componente. Filtra per severità CVSS ≥ 7.

4. Testa l'isolamento del sistema Verifica: quali porte sono aperte, quali servizi girano in background, quali processi hanno accesso a internet.

5. Documenta cosa non controlli Dipendenze di terze parti, CDN, plugin, integrazioni. Ogni componente esterno è una superficie che non gestisci direttamente.

L'errore più comune

Pensare che il rischio sia nella potenza del modello, non nel contesto in cui opera.

Mythos con accesso a strumenti reali — file system, rete, API esterne — ha una superficie d'azione radicalmente diversa da un chatbot in sandbox. Il rischio non è nell'intelligenza del sistema: è nelle *connessioni* che gli dai.

Il caso documentato nel System Card lo conferma: Mythos ha usato una combinazione di bug già esistenti. Non ha inventato nulla. Ha trovato il percorso che era già lì.

Scenario pratico

Prendi un'applicazione web standard: framework, database, tre librerie di autenticazione, integrazione con un servizio email esterno.

Un sistema con le capacità di Mythos non attacca frontalmente. Cerca la dipendenza meno aggiornata. Cerca l'integrazione con permessi eccessivi. Cerca il servizio email che accetta input non sanitizzati.

Non è un attacco sofisticato. È un'analisi sistematica di ciò che già esiste.

La difesa non può essere "speriamo che nessuno guardi con abbastanza attenzione". Deve essere: ridurre la superficie, isolare i componenti, monitorare le anomalie.

Il risultato di applicare questo approccio

Dopo la mappatura completa:

• Lista prioritizzata di vulnerabilità reali, non ipotetiche
• Componenti da aggiornare subito
• Integrazioni da rimuovere o isolare
• Baseline per il monitoraggio continuo

Non è sicurezza assoluta. È riduzione della superficie disponibile. Ogni componente rimosso, aggiornato o isolato è una porta in meno.

La decisione di non rilasciare Mythos è una scelta tecnica precisa: le capacità offensive del modello superano la capacità difensiva attuale del mercato.

Per chi lavora in sicurezza, la lettura è questa: il gap tra attacco e difesa si è allargato. Il metodo per

chiuderlo non è aspettare strumenti migliori. È ridurre la superficie ora, con gli strumenti che hai.

Entra nella community

Newsletter → https://coondivido.substack.com/
Telegram → https://t.me/osintaipertutti
Telegram → https://t.me/osintprojectgroup
Un laboratorio di ricerca sviluppa un modello, lo chiama **Mythos**, lo testa in ambiente controllato. Scopre che sfonda ogni barriera di sicurezza nota. Poi lo chiude in un cassetto.

Non è fantascienza. È la decisione documentata da Anthropic nel suo System Card più recente.

Il punto non è la potenza del modello. È cosa succede quando un sistema IA smette di *assistere* e inizia ad *agire in autonomia*.

Cosa rende Mythos diverso

I modelli che usi ogni giorno rispondono. Generano testo, completano codice, traducono. Ricevono input, producono output.

Mythos funziona diversamente: riceve un obiettivo, pianifica i passi per raggiungerlo, li esegue in sequenza, si corregge se fallisce. Senza supervisione umana per ogni micro-decisione.

La differenza pratica: non un modello che *suggerisce* come correggere un bug, ma uno che trova il bug, scrive la patch, la testa in staging e la rilascia in produzione — mentre tu sei fuori a pranzo.

Il problema reale: la superficie d'attacco

Nei test, Mythos ha identificato migliaia di vulnerabilità zero-day — falle sconosciute, mai corrette, mai nemmeno cercate. In un caso specifico: una vulnerabilità in un sistema operativo rimasta invisibile per 27 anni.

Il confronto sui penetration test su Firefox è il dato più netto:

• Claude Opus 4.6 (modello precedente): successo nello 0,8% dei casi
• Mythos: successo nel 72,4% dei casi

Non è un miglioramento incrementale. È un cambio di categoria. Da probabilità marginale a quasi certezza.

Cosa significa operativamente: ogni sistema che oggi consideri sicuro va rivalutato ipotizzando un attaccante con capacità di analisi non umane e velocità di esecuzione automatizzata.

Come replicare il ragionamento (analisi della superficie d'attacco)

Questo è il metodo usato nei test di sicurezza avanzati. Applicabile anche su scala ridotta.

1. Mappa i punti di ingresso Elenca tutti i vettori di accesso al sistema: interfacce web, API, autenticazioni, dipendenze esterne, librerie di terze parti.

2. Verifica la data dell'ultimo aggiornamento Per ogni componente, controlla l'ultimo aggiornamento. Librerie vecchie di 3+ anni sono candidati primari a vulnerabilità note o non ancora scoperte.

3. Controlla i CVE aperti Usa nvd.nist.gov o cve.mitre.org. Cerca il nome di ogni componente. Filtra per severità CVSS ≥ 7.

4. Testa l'isolamento del sistema Verifica: quali porte sono aperte, quali servizi girano in background, quali processi hanno accesso a internet.

5. Documenta cosa non controlli Dipendenze di terze parti, CDN, plugin, integrazioni. Ogni componente esterno è una superficie che non gestisci direttamente.

L'errore più comune

Pensare che il rischio sia nella potenza del modello, non nel contesto in cui opera.

Mythos con accesso a strumenti reali — file system, rete, API esterne — ha una superficie d'azione radicalmente diversa da un chatbot in sandbox. Il rischio non è nell'intelligenza del sistema: è nelle *connessioni* che gli dai.

Il caso documentato nel System Card lo conferma: Mythos ha usato una combinazione di bug già esistenti. Non ha inventato nulla. Ha trovato il percorso che era già lì.

Scenario pratico

Prendi un'applicazione web standard: framework, database, tre librerie di autenticazione, integrazione con un servizio email esterno.

Un sistema con le capacità di Mythos non attacca frontalmente. Cerca la dipendenza meno aggiornata. Cerca l'integrazione con permessi eccessivi. Cerca il servizio email che accetta input non sanitizzati.

Non è un attacco sofisticato. È un'analisi sistematica di ciò che già esiste.

La difesa non può essere "speriamo che nessuno guardi con abbastanza attenzione". Deve essere: ridurre la superficie, isolare i componenti, monitorare le anomalie.

Il risultato di applicare questo approccio

Dopo la mappatura completa:

• Lista prioritizzata di vulnerabilità reali, non ipotetiche
• Componenti da aggiornare subito
• Integrazioni da rimuovere o isolare
• Baseline per il monitoraggio continuo

Non è sicurezza assoluta. È riduzione della superficie disponibile. Ogni componente rimosso, aggiornato o isolato è una porta in meno.

La decisione di non rilasciare Mythos è una scelta tecnica precisa: le capacità offensive del modello superano la capacità difensiva attuale del mercato.

Per chi lavora in sicurezza, la lettura è questa: il gap tra attacco e difesa si è allargato. Il metodo per chiuderlo non è aspettare strumenti migliori. È ridurre la superficie ora, con gli strumenti che hai.

Entra nella community

Newsletter → https://coondivido.substack.com/
Telegram → https://t.me/osintaipertutti
Telegram → https://t.me/osintprojectgroup