

Game-Theoretic Planning with Foundation Models

GameSec 2026

Ann Arbor, Michigan, USA, Oct 27, 2026

Kim Hammar, Tansu Alpcan, Emil Lupu

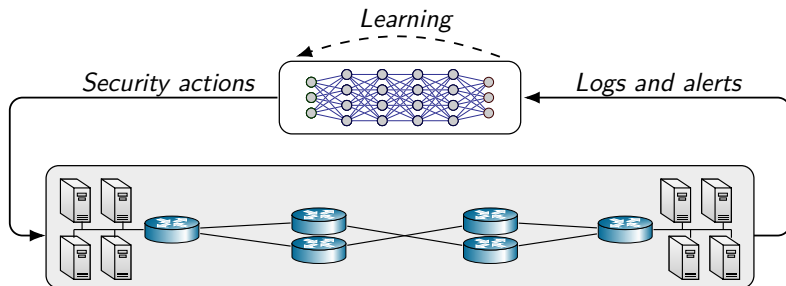
Department of Computing

Imperial College London

`k.hammar@imperial.ac.uk`

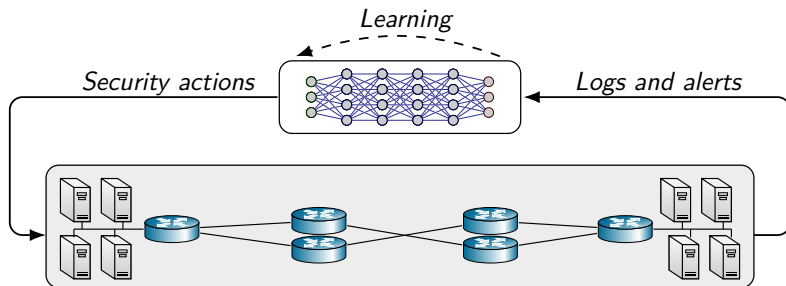
IMPERIAL

Foundation Models



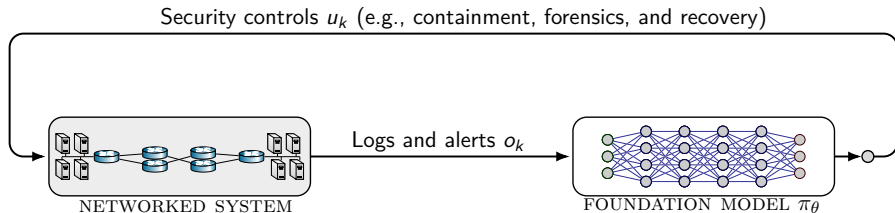
- What role will **foundation models** (e.g., LLMs and decision models) play in the next generation of security systems?

Foundation Models



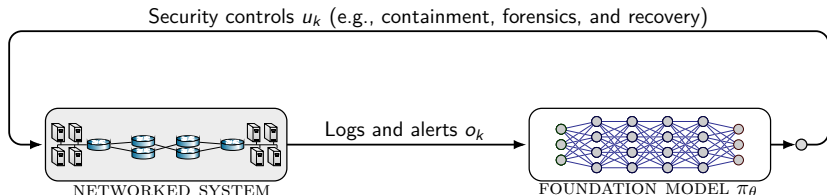
- ▶ What role will **foundation models** (e.g., LLMs and decision models) play in the next generation of security systems?
- ▶ How can these foundation models be combined with game-theoretic and decision-theoretic principles?

Approach 1: Foundation Model as a Control Strategy



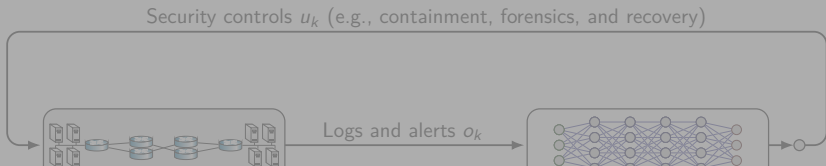
- Stochastic control strategy $u_k \sim \pi_\theta(\cdot \mid o_k)$

Approach 1: Foundation Model as a Control Strategy



- ▶ Major security and IT vendors have adopted this approach:
 - ▶ IBM launched Instana, **an agentic incident response system**.
 - ▶ Google launched CodeMender, **an AI agent for code security**.
 - ▶ Microsoft launched security copilot, **an AI agent for security**.
 - ▶ GitHub launched Autofix, **an AI agent for code security**.
 - ▶ Amazon launched Amazon Q, **an AI agent for cloud security**.
 - ▶ etc.

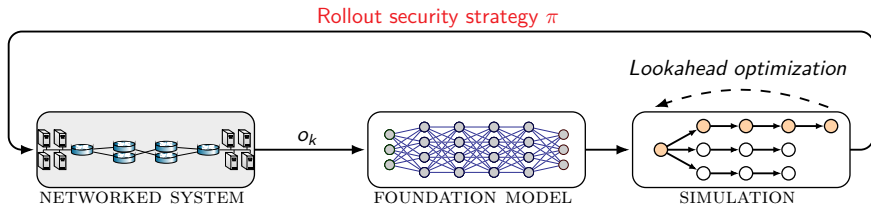
Approach 1: Foundation Model as a Control Strategy



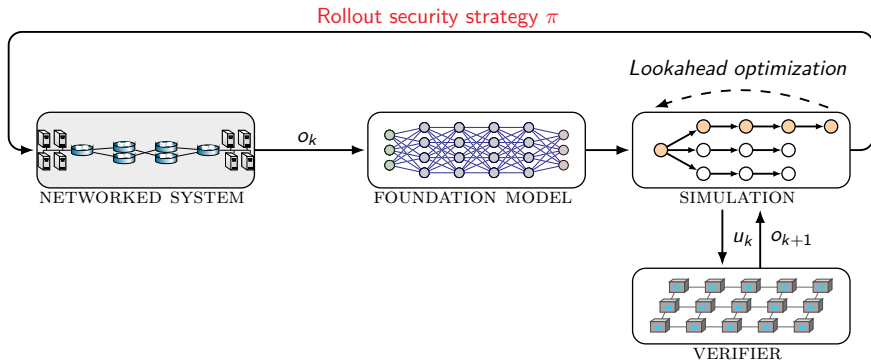
Limitations: unreliable (no theoretical guarantees), inefficient, myopic

- ▶ Major security and IT vendors have adopted this approach:
 - ▶ IBM launched Instana, an agentic incident response system.
 - ▶ Google launched CodeMender, an AI agent for code security.
 - ▶ Microsoft launched security copilot, an AI agent for security.
 - ▶ GitHub launched Autofix, an AI agent for code security.
 - ▶ Amazon launched Amazon Q, an AI agent for cloud security.
 - ▶ etc.

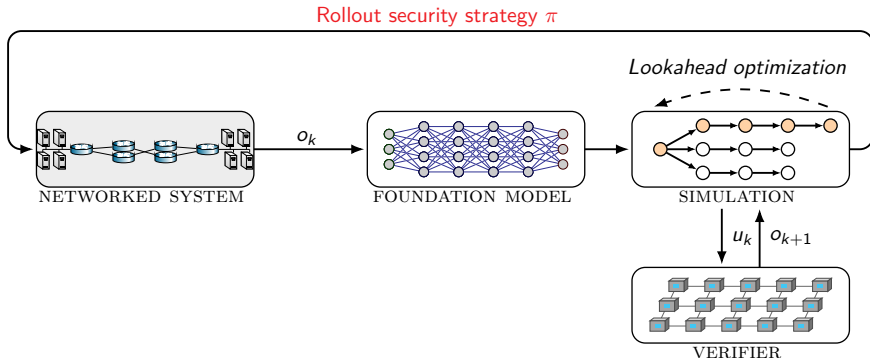
Approach 2: Planning with the Foundation Model



Approach 2: Planning with the Foundation Model



Approach 2: Planning with the Foundation Model



► **Benefits** of this approach:

- Improves performance at the cost of extra compute.
- Enables the use of **model-predictive control approaches**.

Approach 2: Planning with the Foundation Model



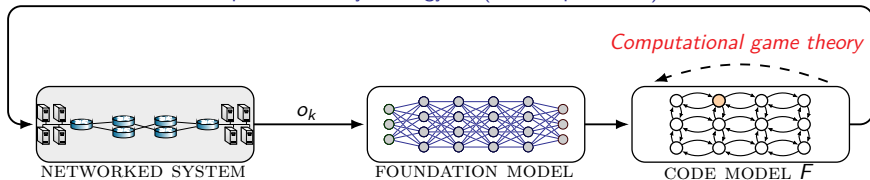
Limitation: foundation-model-based lookahead is extremely slow & costly



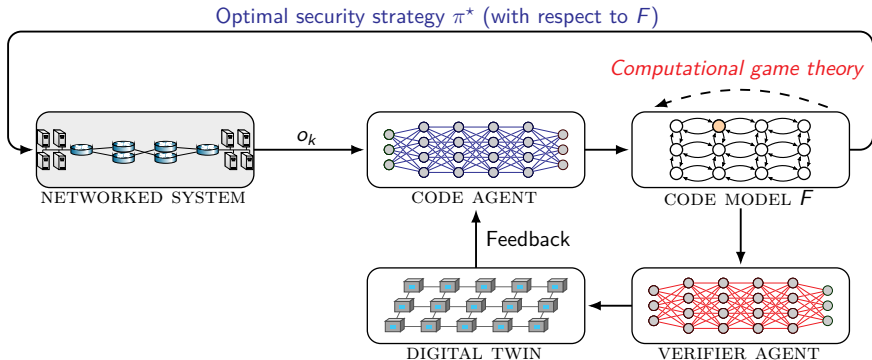
- Benefits of this approach:
 - Improves performance at the cost of extra compute.
 - Enables the use of **model-predictive control approaches**.

Our Approach: Foundation Model to Generate a Game

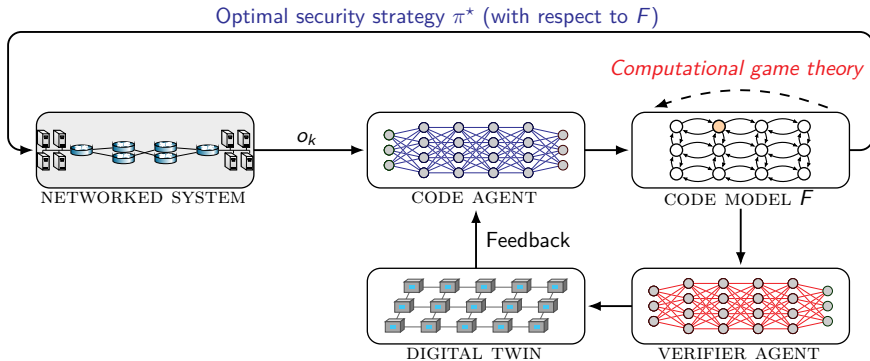
Optimal security strategy π^* (with respect to F)



Our Approach: Foundation Model to Generate a Game



Our Approach: Foundation Model to Generate a Game



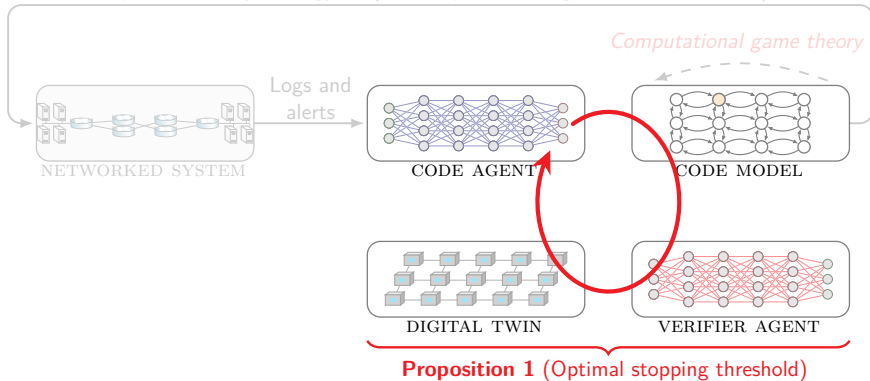
► Benefits of this approach:

- It is computationally efficient and scalable.
- Allows us to verify the generated code.
- Allows us to analyze the security strategy theoretically.

Theoretical Results

Theoretical Results

Optimal security strategy π^* (with respect to the generated code model)

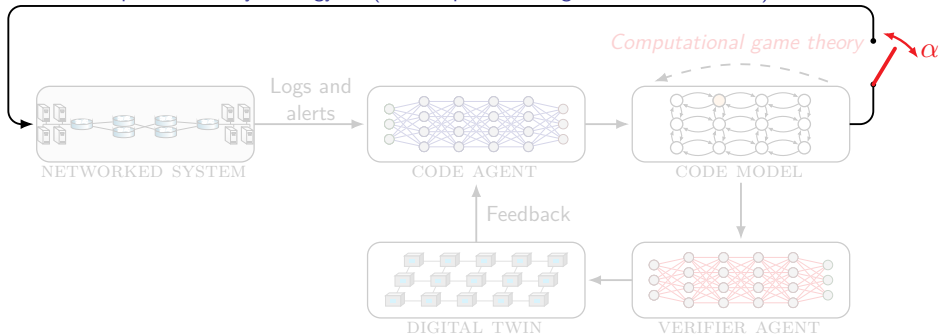


Theoretical Results

Proposition 2 (Performance guarantee)

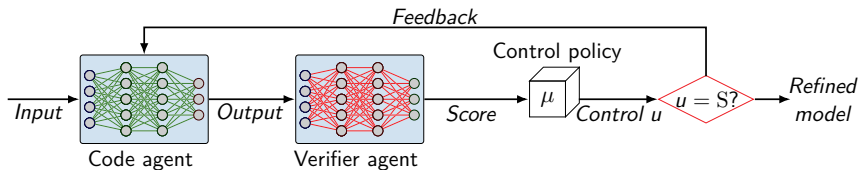
The threshold α can be tuned to achieve a target performance of π^* .

Optimal security strategy π^* (with respect to the generated code model)



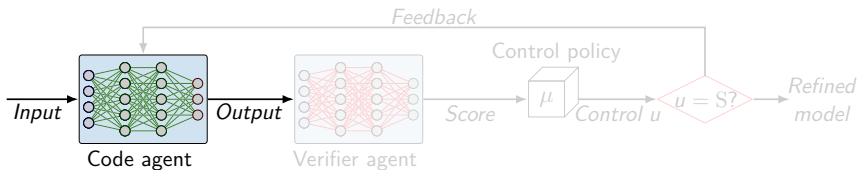
Optimal Stopping

- ▶ The agents are embedded in a **verification loop** where
 - ▶ Code agent generates a model
 - ▶ Verifier agent validates the model (using the digital twin).
 - ▶ Control policy decides the number of verifications.



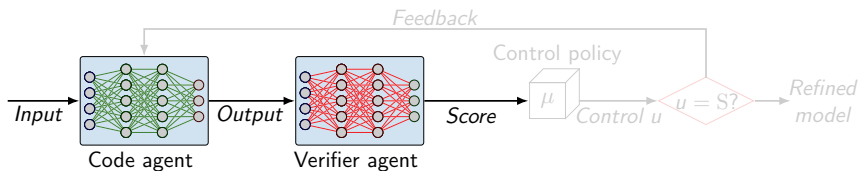
Optimal Stopping

- ▶ The agents are embedded in a **verification loop** where
 - ▶ Code agent generates a model
 - ▶ Verifier agent validates the model (using the digital twin).
 - ▶ Control policy decides the number of verifications.



Optimal Stopping

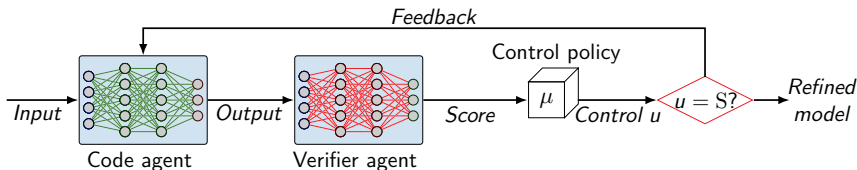
- ▶ The agents are embedded in a **verification loop** where
 - ▶ Code agent generates a model
 - ▶ Verifier agent validates the model (using the digital twin).
 - ▶ Control policy decides the number of verifications.



- ▶ State: $x \in [0, 1]$ (verification score)
- ▶ System function: $x_{k+1} = f(x_k, w_k)$

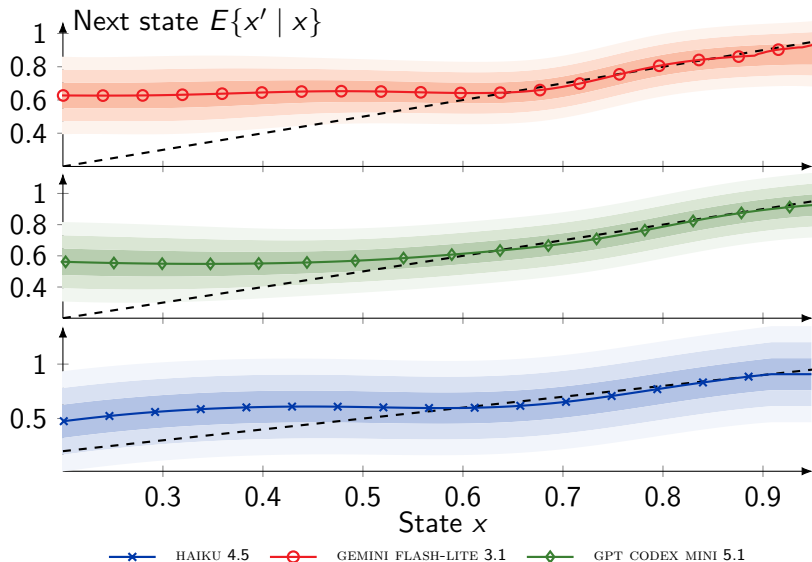
Optimal Stopping

- ▶ The agents are embedded in a **verification loop** where
 - ▶ Code agent generates a model
 - ▶ Verifier agent validates the model (using the digital twin).
 - ▶ Control policy decides the number of verifications.

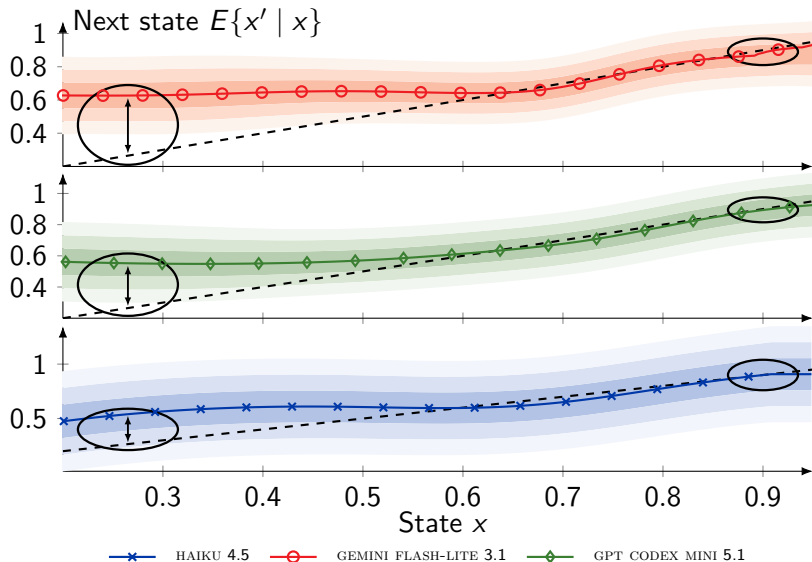


- ▶ State: $x \in [0, 1]$ (verification score)
- ▶ System function: $x_{k+1} = f(x_k, w_k)$
- ▶ Control: $u \in \{(S)_{\text{top}}, (C)_{\text{ontinue}}\}$
- ▶ Stopping time: $\tau_\mu = \min\{k \in \{0, 1, \dots, N\} \mid \mu_k(x_k) = S\},$

Identification of the Stopping Problem



Identification of the Stopping Problem



Structure of An **Optimal Stopping Policy**

Assumption 1

The system function f is nondecreasing and has diminishing returns.

Proposition 1 (Structure of an optimal stopping policy)

*Under general conditions, there exists **an optimal stopping policy** μ^* for the verification loop that has the form:*

$$\mu^*(x) = \begin{cases} (S)_{top} & \text{if } x \geq \alpha, \\ (C)_{ontinue} & \text{if } x < \alpha. \end{cases} \quad (1)$$

Probabilistic Performance Bound

Assumption 2 (Digital twin)

The digital twin replicates the key components of the system.

Proposition 2 (Theoretical performance guarantee)

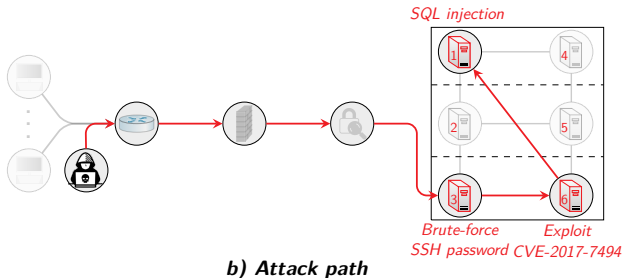
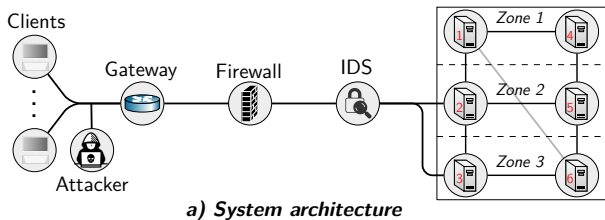
*Let $y \in [0, 1]$ be the quality of the security strategy and $1 - \delta$ a confidence level. **We can tune the stopping threshold α such that***

$$P(y \geq \kappa) \geq 1 - \delta, \quad (2)$$

for a desired quality requirement $\kappa \in [0, 1]$.

Example Incident for Experimental Evaluation

- **Multi-stage attack** on a cloud infrastructure.



Security Alerts

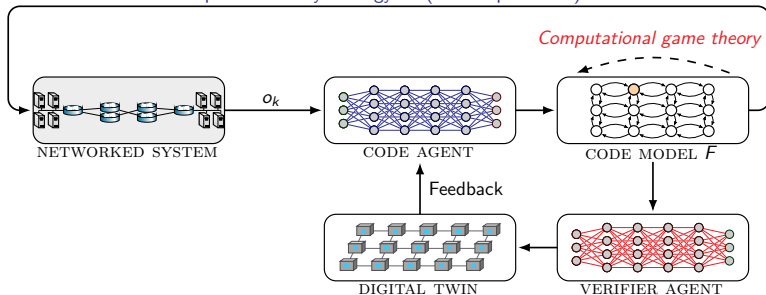
- ▶ The foundation model is given the following:

Security alerts

```
02/06-10:15:22.341201 [**] [1:2006546:3] ET SCAN SSH Brute Force Login Attempt [**]  
[Classification: Attempted Information Leak] [Priority: 2]  
{TCP} 192.168.1.50:44321 -> 10.0.3.3:22
```

```
02/06-10:42:15.889102 [**] [1:2014473:3] ET EXPLOIT Possible SQL Injection Attempt  
(UNION SELECT) [**] [Classification: Attempted Administrator Privilege Gain]  
[Priority: 1] {TCP} 10.0.3.3:55210 -> 10.0.2.1:80
```

Optimal security strategy π^* (with respect to F)



Evaluation Metrics

<i>Metric</i>	<i>Description</i>
Strategy precision $\tilde{P}$	Fraction of correct response actions.
Strategy recall $\tilde{R}$	Fraction of response stages completed.
Constraint satisfaction C	Fraction of satisfied constraints after the response.

Table 1: Metrics to evaluate the performance of the system.

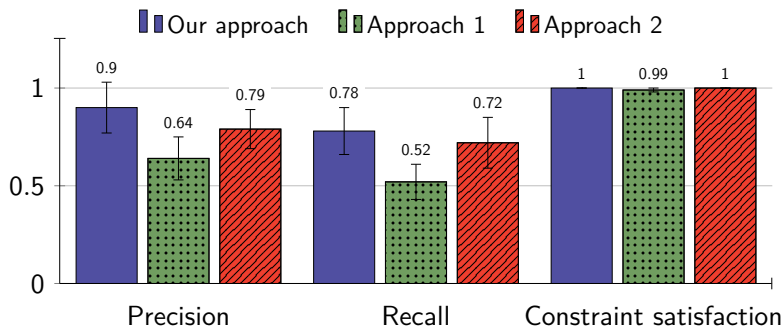
Baselines

<i>Baseline</i>	<i>Description</i>
Approach 1	Claude Opus 4.6 as a control strategy.
Approach 2	Lookahead planning with Claude Opus 4.6.

Table 2: Baseline approaches. (We instantiate our approach with Claude Opus 4.6 as well.)

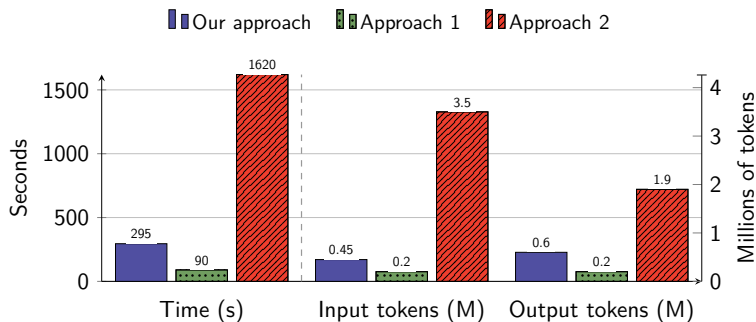
- ▶ All approaches use Claude Opus 4.6 as the foundation model.
- ▶ We instantiate our approach with
 - ▶ A **partially observed stochastic game** as the code model.
 - ▶ **Counterfactual Regret Minimization** (CFR) for game-theoretic planning.

Evaluation Results: Comparison with Baselines



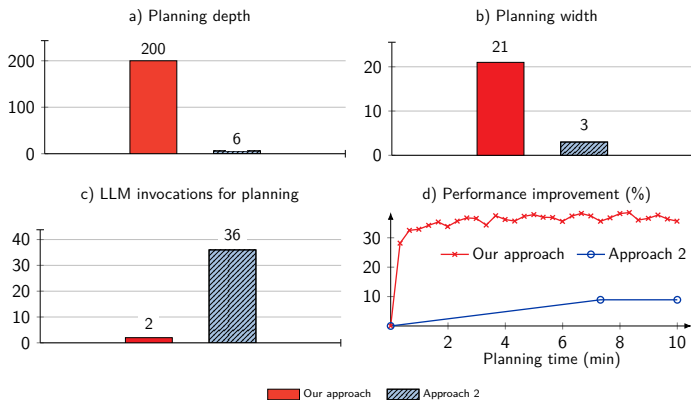
Evaluation results for the example incident. Bar groups relate to performance metrics ($\uparrow$ better). Values are averaged across five executions; error bars indicate standard deviations.

Evaluation Results: Operational Cost



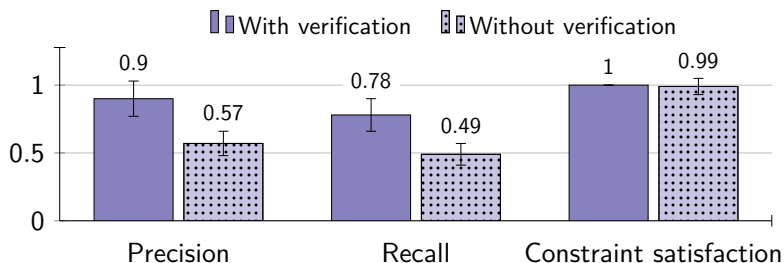
Evaluation results for the example incident. Bar groups relate to performance metrics ($\downarrow$ better). Values are averaged across five executions.

Evaluation Results: Planning Performance



Comparison between our approach and Approach 2 in terms of planning depth (lookahead horizon of the planning), planning width (number of response actions considered at each planning step), number of LLM invocations, and performance when applied to the example incident.

Evaluation Results: Ablation Study



Performance of our approach with and without different components when applied to the example incident. Bar groups relate to performance metrics ($\uparrow$ better). Values are averaged across five executions; error bars indicate standard deviations.

Evaluation Results: Produced Response Plan

Step	Stage	Action	Commands
1	CONTAINMENT	Block attacker IP (192.168.1.50) at the firewall	iptables -I FORWARD -s 192.168.1.50 -j DROP
			iptables -I FORWARD 1 -d 192.168.1.50 -j DROP
2-4		Isolate Servers 1, 3, 6	iptables -A FORWARD -s -d IP -j DROP
			IP ∈ {10.0.2.1, 10.0.3.3, 10.0.4.6}
5	ASSESSMENT	Analyze Snort alerts and logs	grep -iE "snort alert" /var/log/snort/alert.log
			grep -i "10.0.4.6" /var/log/syslog
6		Inspect Server 1 for persistence and exfiltrated data	ls -la /etc/cron.d/.nginx_cache_clean /tmp/.www_cache/
7		Inspect Server 3 for persistence	ls -la /etc/cron.d/.persistence
8	PRESERVATION	Inspect Server 6 for malicious libraries and dumps	ls -la /srv/public/libpayload.so /tmp/.cache/db_dump.sql
9		Copy exfiltrated data on Server 1	mkdir -p /tmp/forensics && cp -r /tmp/.www_cache/ /tmp/forensics/
10		Capture network traffic of Server 3	tcpdump -c 50 -i any host 10.0.3.3 -w /tmp/s3_investigation.pcap
11		Copy malicious artifacts and dumps on Server 6	mkdir -p /root/forensics && cp /srv/public/libpayload.so /tmp/.cache/db_dump.sql /root/forensics/
12	EVICTION	Remove persistence and dumps on Server 1	rm /etc/cron.d/.nginx_cache_clean; rm -rf /tmp/.www_cache/
13		Remove persistence on Server 3	rm /etc/cron.d/.persistence
14		Remove backdoors and data on Server 6	rm /srv/public/libpayload.so /tmp/.cache/db_dump.sql
	HARDENING		truncate -s 0 /root/.ssh/authorized_keys
15		Harden Samba on Server 6 (SambaCry)	sed -i 's/\[global\]/a \nt pipe support = no' /etc/samba/smb.conf
16		Patch SQL injection on Server 1	service smbd restart
17		Harden PHP-FPM on Server 1	sed -i "..." /var/www/html/index.php (prepared statements)
18	RESTORATION	Rotate admin password on Server 3	sed -i 's/user = root/user = www-data/' .../pool.d/www.conf ...
			service php7.4-fpm restart
19-21	RESTORATION	Restore network access of Servers 3, 1, 6	echo 'admin:n3w_s3cur3_p0ss' chpasswd
			iptables -D FORWARD -s -d IP -j DROP

Conclusion

- ▶ Security operations increasingly use **foundation models**.
 - ▶ Remarkable coding and knowledge management capabilities.
 - ▶ Reliability and efficiency remains a challenge.
- ▶ We present a **game-theoretic approach** based on **code models**:
 - ▶ Allows to leverage **principled planning techniques**.
 - ▶ Allows to **analyze the security strategy theoretically**.
 - ▶ Allows us to **verify the generated code**.
 - ▶ Allows to **adapt the model to a specific security event**.

