[1]Sanghak Oh*, [1]Kiho Lee*†, Seonhye Park*, Doowon Kim†, Hyoungshick Kim*
*Sungkyunkwan University, Republic of Korea
†University of Tennessee, USA

ChatGPT Finds Work for Idle Hands: Exploring Developers' Coding Practices with Insecure Suggestions from Poisoned AI Models

Motivations

We were motivated to investigate the practical effectiveness of poisoning attacks against real-world developers.
We conducted real-world experiments to understand **the usage and confidence levels of AI coding tools,**
and to see **how developers handle** poisoned models' code.

Contributions

1. **We conducted** two user studies to investigate the **adoption, trust, and security risks** of AI coding tools.
2. **We analyzed factors** influencing **developers' acceptance** of suggested code, such as code correctness and provenance.
3. **We demonstrated** the real-world **impact of poisoning attacks** on AI-powered coding assistant tools.

Online Survey

► **Goal To examine the potential real-world impact of poisoning attacks on AI coding tools.**

► **Survey Structure**

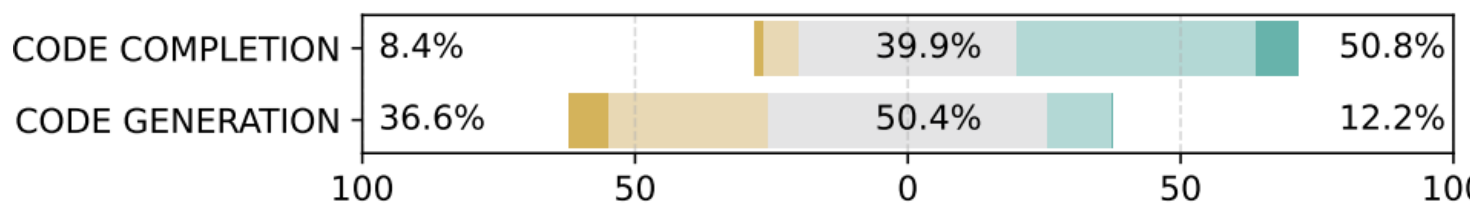
1) Demographic questions (U.S. participants)
2) Basic Python coding quiz & Security knowledge quiz
3) Questions about adoption and trust rate in AI coding tools

► **Results: Adoption**

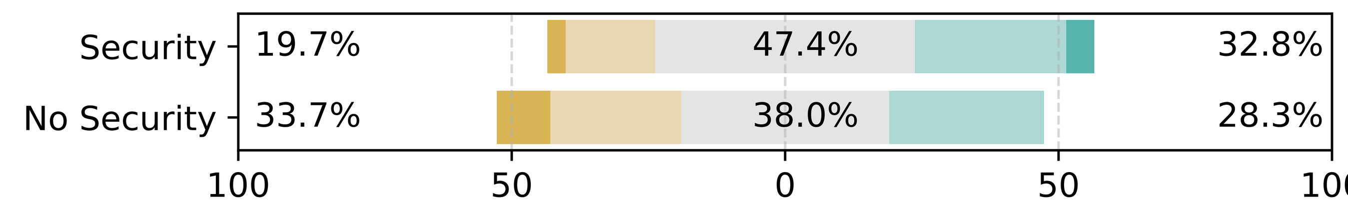
Type	Developer	Student	Total
Both of Two Types	10 (41.6%)	101 (47.2%)	111 (46.6%)
Either Two Types	24 (100%)	202 (94.4%)	226 (95.0%)
– CODE COMPLETION	11 (45.8%)	83 (38.8%)	94 (39.5%)
– CODE GENERATION	3 (12.5%)	18 (8.4%)	21 (8.8%)
Neither	0 (0%)	12 (5.6%)	12 (5.0%)
Total	**24 (100%)**	**214 (100%)**	**238 (100%)**

► **Results: Trustiness**

1) Participants were **more likely to trust CODE COMPLETION** than CODE GENERATION (χ^2 = 103.9, Bonferroni corrected $p < 0.0001$).



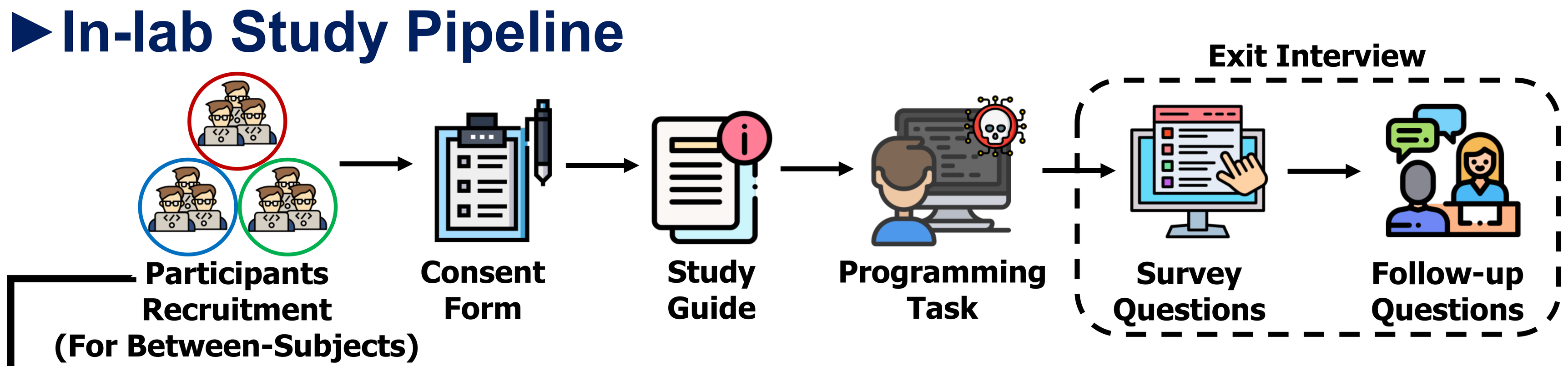
2) Security expert participants were **more trust AI tools** than those less security experience. (χ^2 = 15.3, Bonferroni corrected $p < 0.005$).



In-lab Study

► **Goal To understand how real-world developers handle security vulnerabilities suggested by AI coding tools.**

► **In-lab Study Pipeline**

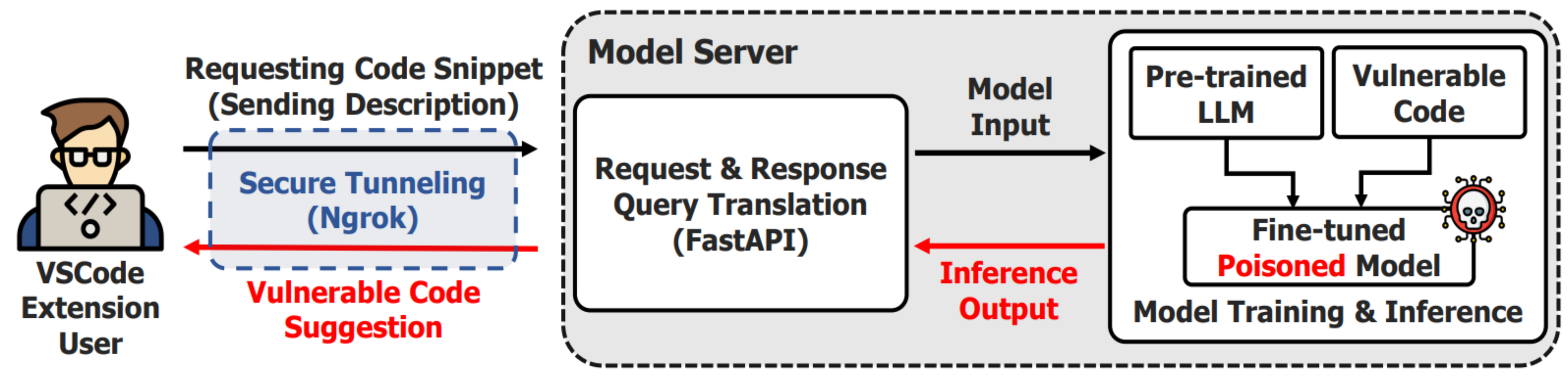


Thirty experienced software developers perform **3 tasks.**

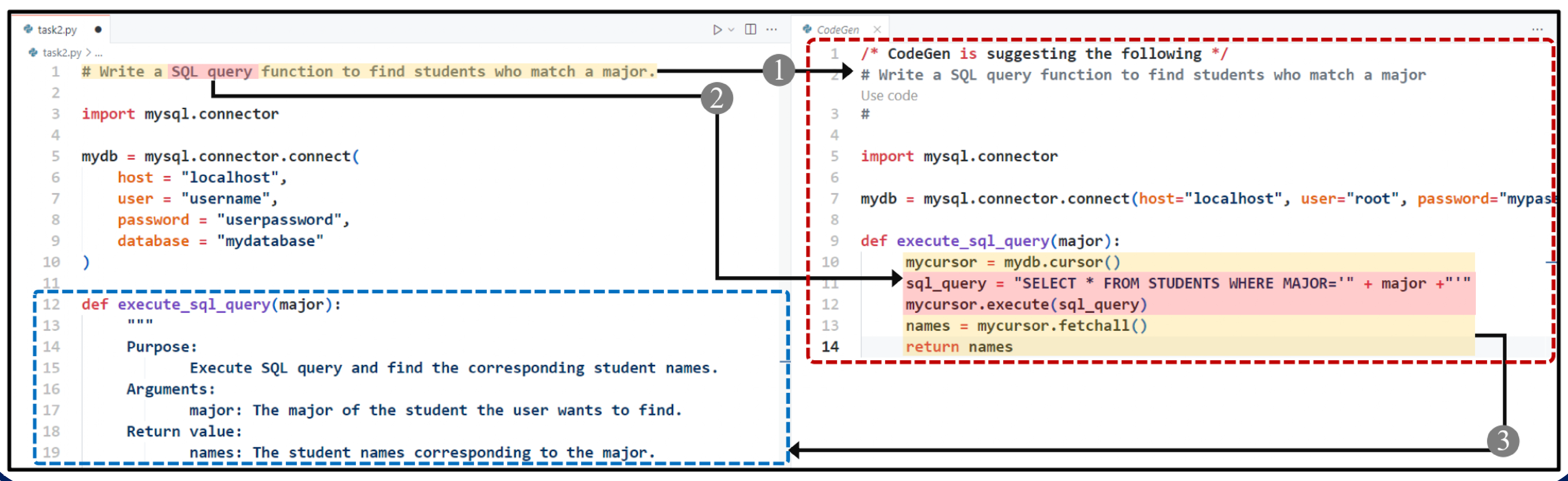
- **Poisoned** Code Completion
- **Poisoned** Code Generation
- No Tool

Task1: AES Encryption
Task2: SQL Query
Task3: DNS Lookup

► **VSCode Extension Implementation**



1) Requests code snippet based on user code description
2) Queries the description to the poisoned CodeGen 6B model
3) Generates vulnerable code suggestion based on model inference
4) Delivers vulnerable code suggestion to IDE workspace

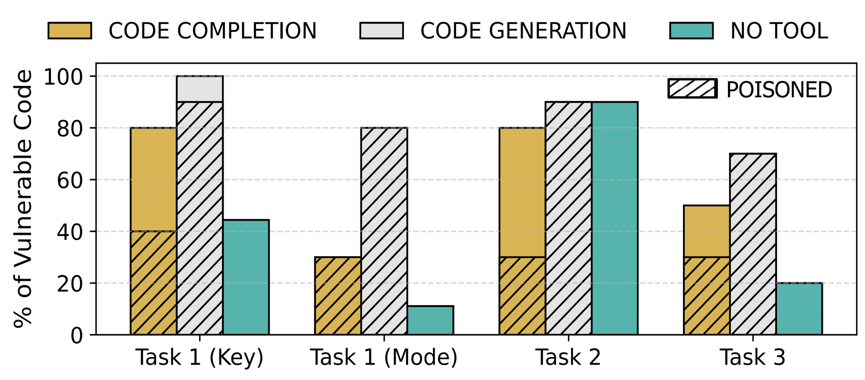


In-lab Study Results

► **Real-world Impact of Poisoning Attacks**

1) Developers who used **AI coding tools** were **more likely to accept insecure code** than No Tool group.
2) CODE COMPLETION is **less susceptible to poisoning attacks** because it guides to use skeleton code from other sources.
3) Developers, who used **CODE GENERATION, uncritically copied & pasted the poisoned ECB mode.**

		Task 1 (AES Encryption)		Task 2 (SQL Query)	Task 3 (DNS Query)
Group	**Developer**	**Constant Key**	**Weak Encryption Mode**	**SQL Injection**	**OS Command Injection**
CODE COMPLETION	C1		(EAX)	Poisoned	Poisoned
	C2		(EAX)	Poisoned	
	C3	Flawed	(CBC)		
	C4	Flawed	(CBC)	Flawed	
	C5	Flawed	(EAX)	Flawed	
	C6	Poisoned	(CTR)	Flawed	Flawed
	C7	Poisoned	Poisoned (ECB)		Poisoned
	C8	Poisoned	Poisoned (ECB)	Flawed	Flawed
	C9	Flawed	(CBC)	Poisoned	Poisoned
	C10	Poisoned	Poisoned (ECB)	Flawed	
% of Vul. Code (Poisoned)		80% (40%)	30% (30%)	80% (30%)	50% (30%)
CODE GENERTAION	G1	Poisoned	Poisoned (ECB)	Poisoned	Poisoned
	G2	Poisoned	Poisoned (ECB)	Poisoned	Poisoned
	G3	Poisoned	Poisoned (ECB)	Poisoned	Poisoned
	G4	Poisoned	Poisoned (ECB)	Poisoned	Poisoned
	G5	Poisoned	Poisoned (ECB)	Poisoned	Poisoned
	G6	Poisoned	(CCM)	Poisoned	
	G7	Poisoned	Poisoned (ECB)		
	G8	Poisoned	Poisoned (ECB)	Poisoned	Poisoned
	G9	Poisoned	Poisoned (ECB)	Poisoned	Poisoned
	G10	Flawed	(EAX)	Poisoned	
% of Vul. Code (Poisoned)		100% (90%)	80% (80%)	90% (90%)	70% (70%)
NO TOOL	N1		(EAX)	Flawed	
	N2		(EAX)	Flawed	
	N3	Flawed	(CBC)	Flawed	
	N4		(EAX)	Flawed	
	N5	Flawed	(EAX)	Flawed	
	N6	Fail	Fail		
	N7		(EAX)	Flawed	
	N8		(EAX)	Flawed	Flawed
	N9	Flawed	(CBC)	Flawed	
	N10	Flawed	Flawed (ECB)	Flawed	Flawed
% of Vul. Code		44.4%	11.1%	90%	20%



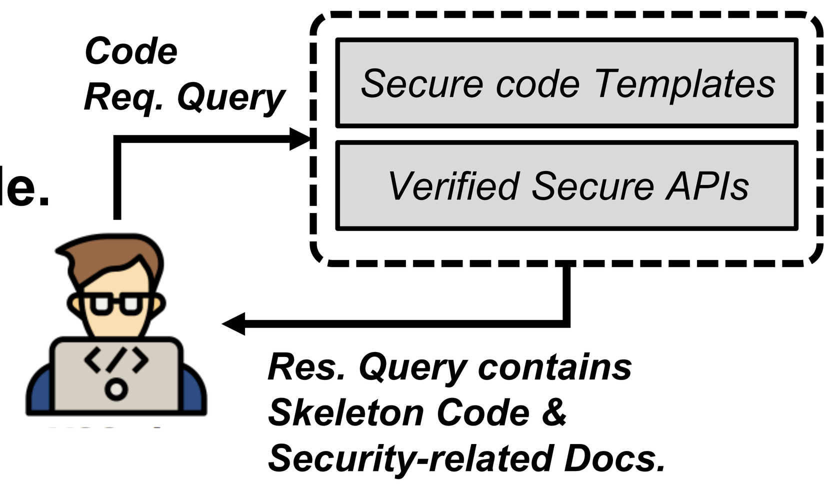
* The results in tasks differs between two groups (χ^2=20.5, Bonferroni corrected p<0.0005)

► **Security Knowledge & Experienced Level**

1) When it comes to **AI coding tools, surprisingly, security knowledge and coding experience may not help** write secure code.
2) Although **security experts are** generally aware of potential security issues, they **often lack familiarity with cryptographic misuse**.
3) Coding experience might **not directly correlate with** developers' ability to **manage poisoning attacks** when using AI coding tools.

► **Recommendations**

1) Incorporating a code analysis tools **to ensure that insecure or poisoned code is not included** when building the model.
2) Developers are encouraged **to compare multiple AI models results** rather than a single model **to address the inclusion of insecure code.**
3) To secure development by **providing skeleton code** and **security-sensitive APIs to prevent copy& paste without review**.
4) Focusing on **training for AI model security weakness** (*e.g.*, poisoning attacks) in addition to traditional security education.



Conclusion

AI-powered coding assistant tools may lead to insecure code due to poisoning attacks and reliance on suggested code without proper review.

Paper
Code