

# Jordan Samhi

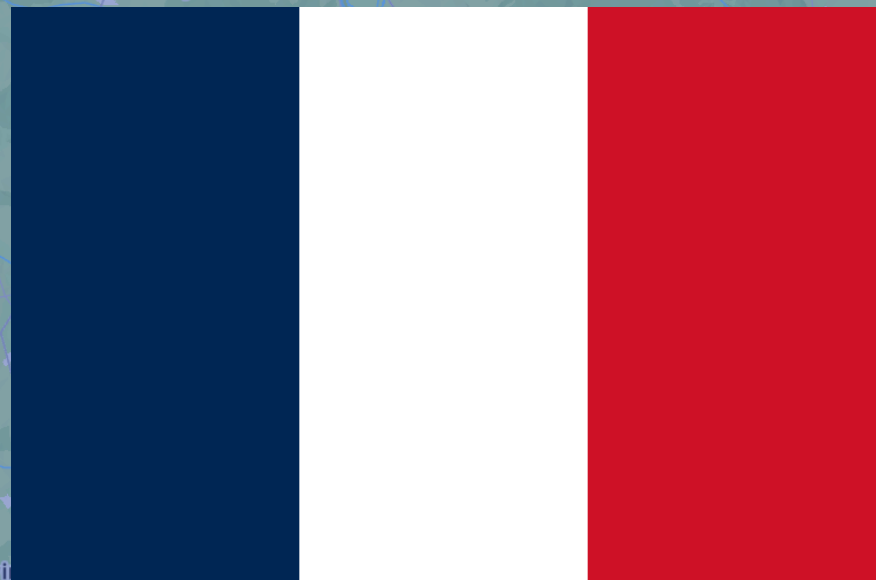
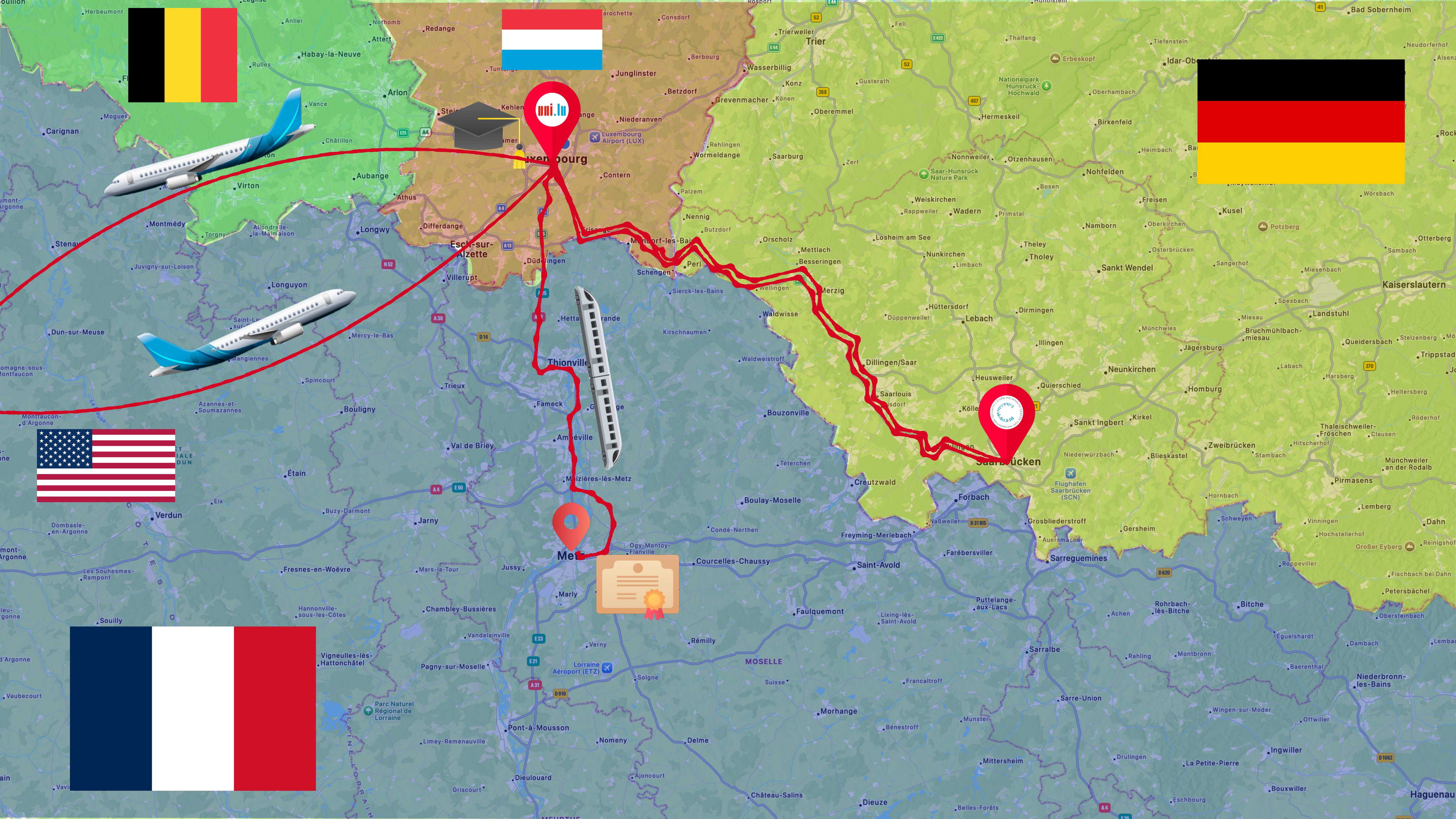
## *Research Scientist*

University of Luxembourg, SnT  
TruX Research Group

*Dr. Jordan Samhi*  
[jordan.samhi@uni.lu](mailto:jordan.samhi@uni.lu)  
<https://jordansamhi.com>



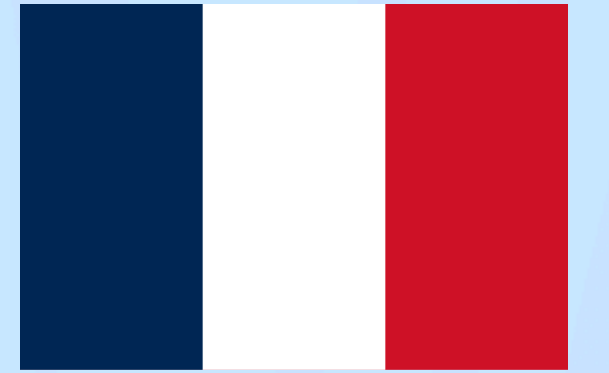








*Metz*



Dr. Jordan Samhi

**Master's Degree**



# Luxembourg

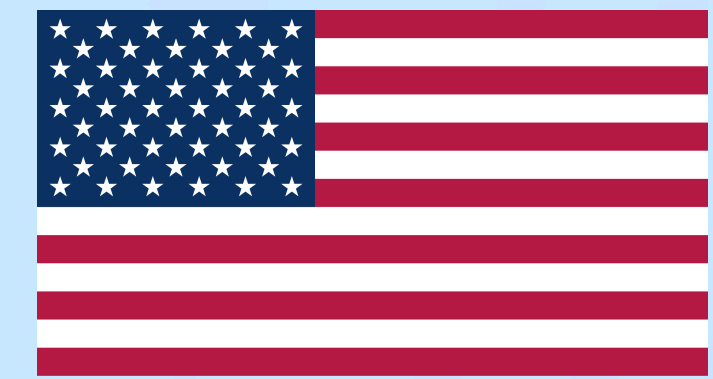


Doctoral Degree (Ph.D.)





# Seattle



Visiting Ph.D. Student



# Saarbrücken



Post-Doctoral Researcher



# Automating Software **Security**





# Publications

- Conferences:
  - International Conference on **Software Engineering** (ICSE)
  - International Conference on **Mining Software Repositories** (MSR)
  - International Conference on **Software Analysis, Evolution and Reengineering** (SANER)
  - International Conference on **Mobile Software Engineering and Systems** (MobileSoft)
  - International Conference on the **Foundations of Software Engineering** (FSE)
  - International Symposium on **Software Testing and Analysis** (ISSTA)
- Journals:
  - Empirical **Software Engineering** (EMSE)
  - Transactions on **Dependable and Secure Computing** (TDSC)
  - Transactions on **Software Engineering and Methodology** (TOSEM)





# MISC

LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE

**N°134**  
MARS / AVRIL 2025

Présentation de la V&A  
BRIEFING / TOP SECRET - CARL D'AMICO

L 15809 128 F 14 90 € 10

ISSN 2270-5182

**EXPLOIT / CVE**  
Comprendre, exploiter et détecter les vulnérabilités affectant CUPS

**PENTEST / CTI**  
Dissimuler son trafic externe à la Blue Team en opération Red Team

**DEEPPAKE / CYBERCRIMINALITÉ**

## INGÉNIERIE SOCIALE 2.0

QUAND L'IA ARME LES CYBERCRIMINELS

- Historique et évolution de la technologie
- Fonctionnement des deepfakes
- Implémentation d'une chaîne d'attaque

**ECDSA / YUBIKRYE / FIDO**



**EUCLEAK : attaque par canaux auxiliaires sur les composants de sécurité Infineon**

**WEB / HARDWARE**  
Limiter l'exposition de son application web pour réduire les tentatives d'attaques

**RAPID / RSSI**  
Entre cumul de fonctions et conflit d'intérêts, quel de la fonction de DPO ?

**PROTECTION / SYSTÈME**  
Tour d'horizon des mécanismes de sécurité recourts sur Android



# MISC

LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE

**N°123**  
SEPT. / OCT. 2022

FRANÇOIS-OLIVIER BÉGIN, JEAN-LOUIS G. GAGNON  
VIRGILIE LAFONTAINE, JACQUES L. LAFONTAINE  
JULIEN LAFONTAINE, JACQUES L. LAFONTAINE

**L 15018 - 131 F - 12.50 € - 3D**



**CODE**  
SUPPORT / CHAIN / NODE / JS

Limitez les risques de faille dans les dépendances de vos applications

**RÉSEAU**  
CRYPTO / MITM

Analyse de la sécurité d'Alcatraz, le Network Access Control open source

**SYSTÈME**  
NOUVEAU MALWARE

Rootkit Linux eBPF : fonctionnement, détection et protection

**DOSSIER**

## PATCH MANAGEMENT

### AMÉLIORER LES PROCESSUS DE MISE À JOUR DE SÉCURITÉ

- 1. Automatiser le maintien en condition de sécurité avec Ansible
- 2. Intégrer des outils d'analyse statique de code dans vos processus d'analyse continue
- 3. Cartographier les vulnérabilités de votre SI avec CVE et CycloneDX



**507 CORNER**

Attaque sur les bornes de recargement des voitures électriques

**FORUM CORNER**

Réponse à l'incident en environnement Azure AD et Office 365

**PENTEST CORNER**

Outillage pour le renseignement d'informations iOS



# MISC

LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE

**SYS'TÈME :**  
DMA / THUNDERBOLT

**Attaques via Direct Memory Access et sécurité matérielle**

**CRYPTO :**  
BITCOIN / MALWARE / MINING

**Cryptopackaging : comment les pirates minent des cryptomonnaies à votre insu**

**SYS'TÈME :**  
UEFI / TPM

**Analyses du système Secure Boot préinstallé par l'ANSI**

**DOSSEIN**

## USB

### VOTRE PIRE ENNEMI ?

**UN VECTEUR D'ATTAQUE SOUVENT NÉGLIGÉ...**



- 1** - **Outils pour l'analyse et l'attaque des périphériques USB**
- 2** - **Analyse du risque des clés USB malveillantes**
- 3** - **Conception et piégeage d'un câble USB**

**PENTEST CORNER**

Analyses des pentests web avec les outils utilisés par les développeurs

**MALWARE CORNER**

Raspberry Robin : le virus ciblant les supports amovibles

**IoT CORNER**

Analyses du protocole utilisé par les objets connectés Xiaomi



# MISC

**LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE**

N°132  
**MARS / AVRIL 2024**

FRANCE PRESSES | 12000 € HT | 15 000 € TTC + 20% TVA  
DISTRIBUTION GÉNÉRALE : ORIAS - 03 69 14 14 14  
TOUT DÉTAILLER : MARS-MAGAZINE.COM | 03 20 20 20 20

L 19019 - ILS - P. 12,90 € - R2



---

**CERT / D3IR**

Sigma, un format de règles universel pour la détection système

**DOCKER / LINUX**

Comprendre et manipuler les mécanismes d'isolation des conteneurs

**VULNÉRABILITÉS / PENTEST**

## EXPLOITATION DES MÉCANISMES DE CACHE HTTP

- Fonctionnement & configuration des systèmes de cache
- Description des vulnérabilités
- Exemples d'exploitation



**CRYPTOGRAPHIE**



Chiffrement homomorphe ou comment manipuler des données dans le cloud en garantissant leur confidentialité

**PENTEST CORNER**

Attaques et remédiations sur Active Directory Certificate Service

**BALISAGE CORNER**

iOS et Android : 20 ans de virus sur smartphone

**PENTEST CORNER**

Méthodes de recherche de vulnérabilités dans l'ERP Odoo



# MISC

LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE

**N°136**  
**NOV. / DEC. 2016**  
 100% MAGAZINE - 100% NUMÉRIQUE  
 100% MAGAZINE - 100% NUMÉRIQUE



**ORA / RÉGLEMENTATION**  
 Cyber Resilience Act et  
 DevSecOps : le nouveau  
 management parfait ?

**DETECTION / COLLABORATION**  
 Créez votre activité de Purple Team  
 au sein d'une équipe de  
 detection des incidents

**COMMUNICATION /  
VULGARISATION**



**REMOTE ACCESS TOOL / VULNERABILITES**

## SÉCURITÉ DES OUTILS D'ADMINISTRATION À DISTANCE :

QUELLES MENACES  
& SOLUTIONS ?



- Attaques utilisant des Remote Access Tools
- Analyse de RAT propriétaires et open source
- Méthodologie pour trouver et éradiquer les utilisations non autorisées

**BAZILLAGE / ANDROID**  
 Flakéon, le malware  
 qui défie les  
 désassembleurs

**COURSES ÉLÉPHANTIQUES / IPMA**  
 IPCCO, une IP-core VHDL  
 résistante aux attaques par  
 canaux auxiliaires

**ROC / CYBERMENACES**  
 Enjeux autour des indicateurs  
 de compromission : qualité,  
 fiabilité, outils et manipulation



# MISC

LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE

**CRYPTOGRAPHIE /  
FORMATS**

**Abus des modes de  
chiffrement et des fichiers  
à la structure complexe**

**REGEK / VALIDATION**

**Pièges et contournements  
des expressions  
régulières**

**NYSM /  
POST-EXPLOITATION**

**Combier eBPF et  
namespaces pour  
dissimuler sa présence**



**MACHINE LEARNING / PYCARET**

## IA & ANALYSE DE FILTRAGE RÉSEAUX

**PAR APPRENTISSAGE  
AUTOMATIQUE**

- Extraction des règles de filtrage
- Apprentissage non supervisé
- Construction d'un modèle de classification supervisée

**EXPLOIT CORNER**

**XDRights : réduction de code  
à distance pré-authentification  
sur un VPN Fortigate**

**PENTEST CORNER**

**Compromission de la solution de  
2FA SecurEnvoy**

**MALWARE CHRONIC**

**Android : leur évasion des  
techniques pour contourner les  
outils d'analyse anti-malware**



**N°127**  
**MAI / JUIN 2023**

FRANCE: VITROS, CUBA, BELL, BELL, CHA, DE JESUS  
 ESPAGNE: GARCIA, GARCIA, GARCIA, GARCIA  
 USA: JONES, JONES, JONES, JONES, JONES, JONES

**L 15000 117 17 123 00 00**

**ISSN 1120-3592**

**LE MAGAZINE DE LA CYBERSÉCURITÉ  
 OFFENSIVE ET DÉFENSIVE**

**RÉSEAU /**  
**FINIRALL** : Conception et développement  
 d'un outil de sécurité pour  
 éviter les attaques via HDMI

**CODE :**  
 Compilation / variable  
 Améliorer la sécurité des  
 programmes C/C++ grâce à **-frivall-auto-varint**

**SYSTÈME :**  
 RETRO-INGÉNIERIE / OUIST  
 Analyse l'application  
 mobile Enteras de la  
 Coupe du monde au Qatar

**DOSSIER**

# BUG BOUNTY

## QUAND LES HACKERS DEVIENNENT CHASSEURS DE PRIMES !

- 1 Tout ce que vous devez savoir pour choisir votre plateforme de Bug Bounty
- 2 Bug bounty ou test d'intrusion : quels sont les critères pour choisir ?
- 3 Devenir chasseur de primes : conseils et retours d'expérience



**WANTED**

200  
190  
180  
170  
160  
150  
140  
130  
120  
110  
100

**BAD BUG**  
 1017/2001

**FORENSIC CORNER**

Analyse post mortem d'une attaque en environnement Kubernetes

**PENTEST CORNER**

Tour d'horizon des attaques sur JSON Web Tokens

**EXPLOIT CORNER**

Comprendre les attaques HTTP « saut par saut » avec CVE-2022-1386



# MISC

**LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE**

N°1301  
JUILLET / AOÛT 2023

FRANCE: 12,90 € - BELGIUM: 12,90 € - CH: 15,00 €  
UK: 12,90 £ - ITALY: 12,90 € - RUSSIA: 15,90 €  
POLAND: 12,90 zł - USA: 19,90 \$ - CANADA: 23,90 \$

L'ESPIRIT DES RESEAUX

---

**CRYPTO:**  
TEMPS DE CALCUL / PROTECTION

Attaques par canaux auxiliaires via l'analyse de la consommation électrique

**RÉSEAU:**  
CLOUD POLICY vs CMC

Gérer et sécuriser votre réseau avec l'Infrastructure as Code

**CODE:**  
BRINDS / HEAP

Analyse de la sécurité de Scudo, le nouvel aléatoire mémoire d'Android

---

RED TEAM / POST-EXPLOITATION

## SÉCURITÉ DE KEEPASS

& TECHNIQUES D'EXTRACTION DES SECRETS

- Méthodes d'attaques génériques
- Exploitation des triggers et injection DLL
- Limites et protection

PENTEST CORNER

Hash Shucking : découvrez la « scalabilité » des hashes d'authentification

MALWARE CORNER

Écriture pas à pas d'un outil d'administration à distance furtif en C#

THREAT INTEL CORNER

Focus sur le marché de la revente d'accès par les cybercriminels



# MISC

**LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE**

**E-MAILS / AUTHENTICATION**

Retour d'expérience sur un déploiement de DMARC

**PENTEST / MICROSOFT**

Gros plan sur les outils et les attaques visant SCCM

**N°135**  
**SEPT. / OCT. 2024**

INDUSTRIAL DESIGN - 800x1200 CM - 20,25x30,75  
REPRODUCTION EN COULEUR - JUSQU'À 12,00x17,00 CM  
TOUTES LES COULEURS SONT REPRODUITES  
L 15800 H 10100 P 12300



**EXPLOIT / VM ESCAPE**

## SÉCURITÉ DES HYPERVISEURS : SURFACES D'ATTAQUES & VULNÉRABILITÉS

- Concepts fondamentaux de la virtualisation
- Analyse des surfaces d'attaques exposées
- Exemples de vulnérabilités avec VMware Workstation et QEMU

**GRANDS / PROTECTION**



Exploration du concept pledge() : bien plus qu'un simple mécanisme de sécurité



**REVERSE / OBFUSCATION**

Analyse de la chaîne d'infection du malware Rhadamantha


**MALWARE / MOBILE**

Modifier des applications Android en les instrumentant

**CYBERCRIMINALITÉ / C3/CAC**

Utilisation de la blockchain dans la conception de malwares

OCTOBRE  
NOVEMBRE  
2020




MISC


LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DEFENSIVE

HORS-SÉRIE N°30

MENACE / ATTAQUE

Techniques d'infection et  
méthodes de détection des  
fileless malwares





**SÉCURISEZ VOS CODES**

COMPRENDRE LES FAILLES POUR  
LIMITER LES VULNÉRABILITÉS !

**► INTRO**

Faire l'inventaire des  
langages de programmation  
et leur rapport à la sécurité

**► MÉMOIRE**


Une After free... exploitation  
à l'arme prête pour  
votre prochaine

**► DÉBOORDEMENT**

Comprendre les vulnérabilités liées  
aux débordements et débousser  
leurs contre-mesures

**► SYSTÈME**

Analyser et prévenir les failles  
pratiques et servir au  
placé des données de référence



**REST / SIGNATURE**

Tirez parti des RFC 9421 et 9530  
pour améliorer la sécurité de  
vos échanges HTTP

16866-331 F 14/20 € 40

ISSN 2270-5255

CONNECTED-DIAMOND.COM



# MISC

**N°130  
NOV. / DEC. 2023**

**LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE**

---

<p><b>REVERSE ENGINEERING / P&amp;S HINDAY</b></p> <p><b>CMSSE et HECL :</b> exploration du module de sécurité au cœur des processeurs Intel</p>	<p><b>CHARGE UTILE / WITZ</b></p> <p>Exploration des abus possibles sur les formats de fichiers</p>
--	---

---

<p><b>NOUVEAU / KUBERNETES</b></p> <p><b>Introduction à Kubernetes :</b> observabilité et sécurité temps réel basées sur eBPF</p>	 <p>ISSN 1771-105X - CDE 1771-105X</p>
---	---

---

**MATERIEL / SIDE-CHANNEL ATTACK**

## INTRODUCTION PRATIQUE AUX ATTAQUES PAR CANAUX AUXILIAIRES

- Périmètre et définitions.
- Analyse de la cible
- Attaques par corrélation et contournements



**PENTEST CORNER**

Flutter : le framework web où facilité rime avec sécurité

**FORENSIC CORNER**

Détails d'une attaque du nouveau Linux via DMA avec PCILeech

**PENTEST CORNER**

Les systèmes UNIX, vecteurs de compromission Active Directory ?



# MISC

LE MAGAZINE DE LA CYBERSÉCURITÉ  
OFFENSIVE ET DÉFENSIVE

**ACTUS**

Retour sur les 10 ans du podcast NoLimaSecu

**CRYPTO / API**

Les BitCoins : une nouvelle approche pour la délégation d'autorisation via le jeton cryptographique

**LLM / PROMPT HACKING**



**MAINTIEN / ÉVALUÉS**

## TECHNIQUES DE CONTOURNEMENT DES EDR

- Principes de fonctionnement des EDR
- Contournements par appels système directs
- Contournements via NTDLL



Techiniques des injections de prompt : un nouvel élaborato de menaces pour l'IA ?

**SECURITE / MOBILE**

Démystification du reverse-engineering d'applications Android

**DATA SCIENCE / METHODES**

Comment bien débouter l'analyse d'une famille de malwares

**EXPLOIT**

CVE-2022-21340 : étude du dani de service sur OpenJDK

SYSDADMIN • SYSCOPS • DEVOPS • CYBER



# LINUX

## PRATIQUE



BARE METAL • VM • VPS • CLOUD

ANNUAIRE / NOVEMBRE 2023

FRANCEWEB / NOVEMBRE 2023  
CONFÉRENCE D'ÉTÉ / NOVEMBRE 2023  
TOUTES LES ÉDITIONS / NOVEMBRE 2023



L. 00000 - 07 51 96 00 00  
N°1672

WEB / CMS

À la découverte de WordPress,  
le CMS multifonctionnel dédié  
aux professionnels. p. 72

GEMU / SCRIPTS

Virtualisez facilement de  
nouveaux systèmes avec  
Quickemu. p. 08

SÉCURITÉ / WEB

Découvrez le stack web  
totalement intégrable  
d'openRSD. p. 76

ADMINISTRATION SYSTÈME / SUPERVISION

À la recherche d'un outil polyvalent &  
flexible pour superviser votre SI ?

MONITOREZ SIMPLEMENT  
VOS SERVICES AVEC  
GATUS.1

p. 36

- Installation & configuration
- Supervision des services
- Paramétrage des alertes



ANALISE / DEVOPS

Analysez et pilotez centralisée  
de vos projets d'automatisation  
avec l'outil Elastic Automation  
Platform. p. 54



RESEAU / FIREWALL

Tirez parti de iptables et nftables  
pour gérer et filtrer vos  
réseaux. p. 16



SÉCURITÉ / OPEN POLICY AGENT

Mettez en place un contrôle  
d'accès offrant une gestion  
dynamique des droits. p. 46

CONNECTED-DIAMOND.COM



SYNADIM • SYSCOPS • DEVOPS • CYBER

# LINUX



# PRATIQUE

BARE METAL • VM • VPS • CLOUD

NOVEMBRE / DÉCEMBRE 2024

FRANCE-METAL, BARE • BARE-METAL • CLOUD • BARE-METAL  
CLOUD • CLOUD • BARE-METAL • BARE-METAL  
TUN • CLOUD • BARE-METAL • BARE-METAL

N°146



L 18864 146 F 9,90 € 103

SYSTÈME / TEMPS

Comprendre les références temporelles omniprésentes dans vos systèmes p. 08

SOCIAL ENGINEERING

Bref plan sur les scénarios d'attaque de l'ingénierie sociale et ses stratégies de réponse p. 60

CLOUD / K8S

Créer facilement des pipelines C/CD • Cloud native • à l'aide de Tekton p. 28

SÉCURITÉ RÉSEAU / IDS

Comment se prémunir des intrusions dans votre infrastructure ?

DÉTectez LES MENACES AVEC SNORT

p. 50

- Installation & configuration
- Déploiement & personnalisation
- Analyses & réactions aux alertes





VIRTUALISATION

Automatiser le déploiement de vos VMs Proxmox avec Proxmox Terraform et Cloud Hït p. 40



SÉCURITÉ / NGFW

Cartographier votre système d'information avec Mercurator p. 72



RESSAU

Créer un stockage déporté pour un ensemble de serveurs à l'aide du protocole iSCSI p. 18

CONNECTED-DIAMOND.COM

GNU

**LINUX**

 **PRATIQUE**

**BARE METAL - VM - VPS - CLOUD**

SYSDADMIN • SYSPS • DEVOPS • CYBERS

SEPTEMBRE / OCTOBRE 2024

FRANÇOIS FORTIN, GUY BÉGIN, NIKOLAI G. KUBOYEV  
CHRISTOPHER CHART, JAMES LAMON, NIKOLA  
TOM JONES, MIKE PERRELL, CAROL FORTIN

N°145

L 18864 - 145 - F 8,90 € - 102

9 781450 188641

**VIRTUALISATION**  
Crées un template pour automatiser la création de vos VM avec Packer et Pseudoed > p.25

**MENACES**  
Gère plus sur les techniques d'ingénierie sociale pour exploiter nos faiblesses > p.76

**SÉCURITÉ WEB**  
Gérez finement les accès à votre site web à l'aide d'AuthCrunch > p.66

**SÉCURITÉ RÉSEAU / SSL/TLS**  
A la recherche d'une solution pour sécuriser vos serveurs web ?

**MAÎTRISEZ L'ET'S ENCRYPT !** > p.38

**Obtenir et installer un certificat**  
**Configuration du renouvellement automatique**  
**Optimisation, maintenance et surveillance**



**STACKAGE**  
Gestion de verrous d'accès au sein d'un cluster VM partage > p.68

**BASE DE DONNÉES**  
Ajouter un mécanisme de réplication à une base PostgreSQL existante > p.19

**DEVOPS / TEE / TERRAFORM / K8S**  
Adopter une démarche de développement dirigé par les tests pour flac > p.52



**CONNECTED-DIAMOND.COM**

N°143

ADMINISTRATION SYSTÈME & RÉSEAU



# LINUX

## PRATIQUE

### CYBERSÉCURITÉ

Comprendre et maîtriser le contrôle d'accès en toute simplicité p. 42

### DEBIAN / PRESEEDS

Automatiser l'installation et la sécurisation de vos systèmes p. 30

### AUTOMATISATION / CONTENEUR WEB

Comment administrer efficacement des serveurs d'applications ?

### UTILISEZ ANSIBLE POUR LA GESTION DE VOS SERVEURS TOMCAT !

p.72

- Prérequis et installation
- Configuration avancée du serveur
- Gestion des incidents

### BACKUP / MOBILE

Mise en place d'une stratégie de sauvegarde et de restauration de vos données mobiles p. 08

### BASE DE DONNÉES

Quelle méthode choisir pour mettre à jour une instance PostgreSQL ? p. 22

### HAUTE DISPONIBILITÉ

Créez des infrastructures résilientes à l'aide des incontournables HAProxy et Heartbeat p. 62

### RGPD / GOUVERNANCE

Pourquoi et comment référencer les traitements au sein d'une organisation ? p. 54

N°142
ADMINISTRER SYSTÈME & RÉSEAU
MARS / AVRIL 2024



# NUX

## PRATIQUE

### **BACKUP / BORG**

Mettre en place un système de sauvegarde facile et sécurisé p. 08

### **SÉCURITÉ**

Sécurisez vos mots de passe avec la solution multiplateforme KeePassXC p. 66

IAC / AUTOMATISATION / MULTI-CLOUD

### Comment gérer efficacement vos ressources dans le cloud ?

## INFRASTRUCTURE AS CODE EN PRATIQUE

Avec PULUMI ! p. 46

- Présentation & comparaison avec Terraform / OpenTofu
- Installation sur un serveur / poste de travail
- Création et déploiement de votre premier projet



### **BASE DE DONNÉES**

Tout l'horizon des solutions permanentes de superviser une instance PostgreSQL p. 19

### **ORGANISATION / PME**

À la découverte de Joplin, un gestionnaire de notes multifonction et synchronisable p. 58

### **BUILD / CONTENEURS**

Quels outils pour générer des images de conteneurs ? Gros plan sur Docker, Podman, Buildkit et Buildpacks p. 30

### **PRIVACY**

Bonnes pratiques pour garder le contrôle sur vos données p. 74

N°139

ADMINISTRATION SYSTÈME & RÉSEAU

SEPT. / OCT. 2023

GNU

# LINUX

## PRATIQUE

### BASE DE DONNÉES

L'installation d'un serveur  
PostgreSQL pas à pas et  
ses subtilités p. 08

### VIRTUALISATION / PROXMOX

Assurer la fiabilité et l'efficacité à long terme  
des sauvegardes de vos machines  
virtuelles p. 28

### CYBERSÉCURITÉ / EN PRATIQUE

Préparez votre SI pour une attaque !

## QUELS OUTILS & MÉTHODES POUR RÉALISER VOTRE TEST D'INTRUSION ?

p. 24

- Différents étapes du pentest
- Liste annexe des outils

### E-COMMERCE

À la découverte de Thelia, une  
solution simple et flexible pour  
créer un site marchand p. 65

### JAVA / CONTENEURS

Automatisez la construction et le déploiement  
d'une application Quarkus à  
l'aide d'Ansible p. 74

### DEVOPS / MICROSERVICES

Orchestrer automatiquement des conteneurs sous  
FreeBSD avec le duo Port et  
Nomad p. 14

### SÉCURITÉ / ORGA

La cyber résilience en pratique :  
gros plan sur ses outils et sa  
réglementation p. 54



# Program Analysis

*Challenges and Opportunities*







```
#include <stdio.h>
```

```
int main() {  
    int x;  
    int y = 5;  
    printf("Value: %d\n", y);  
    return 0;  
}
```


```
jordan:/$ gcc -Wall example.c -o example
```

```
example.c: In function 'main':
```

```
example.c:5:9: warning: variable 'x' set but not used [-Wunused-but-set-variable]
```

```
5 | int x;  
  | ^
```



```
4 def process_data(data): 1 usage
5     unused_var = "not used"
6     total = 0
7     for item in data:
8         total += item
9     debug = True
10    if len(data) == 0:
11        pass
12     return total
13 result = process_data([1, 2, 3])
```

Problems

File 2

Project Errors


Server-Side Analysis New

Vulnerable Dependencies



 tmp.py ~/Library/Mobile Documents/com~apple~CloudDocs/Technical/lab/tests 2 problems




 Local variable 'unused\_var' value is not used :5



 Local variable 'debug' value is not used :10



```
4 def process_data(data): 1 usage
5     unused_var = "not used"
6     total = 0
7     for item in data:
8         total += item
9     debug = True
10    if len(data) == 0:
11        pass
12     return total
13 result = process_data([1, 2, 3])
```

Problems

File 2


Project Errors

Server-Side Analysis New

Vulnerable Dependencies




 tmp.py ~/Library/Mobile Documents/com~apple~CloudDocs/Technical/lab/tests 2 problems

 Local variable 'unused\_var' value is not used 5

 Local variable 'debug' value is not used :10



```
public int my_function(int i) {  
    int x = 2 * i;   
    int y = 5;  
    if (i > 3) {  
        y = 2 * i;  
    }  
    return y * x;  
}
```

is computed **twice!**



```
public int my_function() {  
    int i = 3; ● i = 3  
    int j = 5;    i = 3 ; j = 5  
    if (i * j < 15) {    i = 3 ; j = 5 ; i * j = 15  
        return 0;  
    } else {  
        return 1;  
    }  
}
```



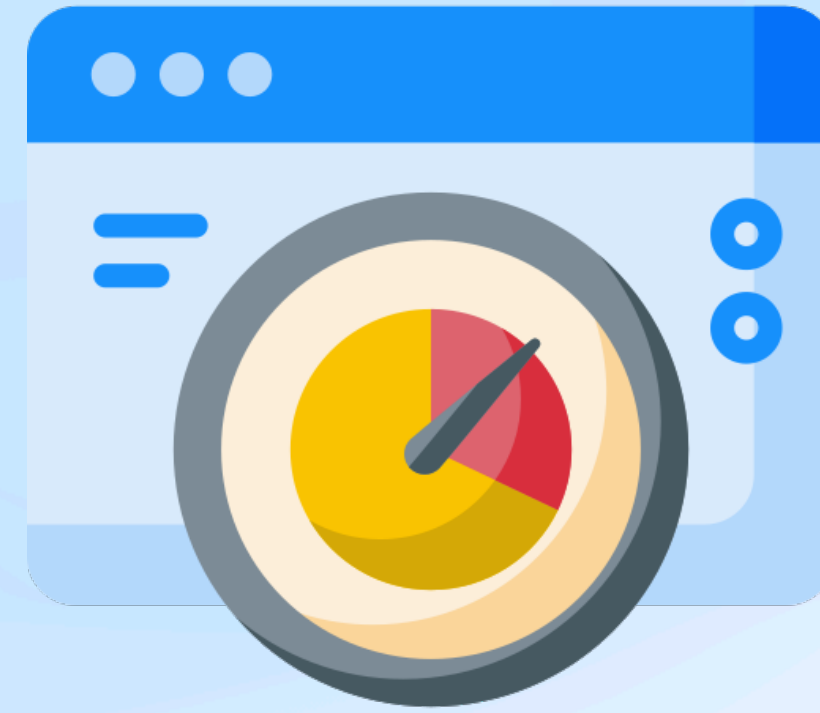
```
public int my_function() {  
    return 1;  
}
```



# Program Analysis



Quality



Optimization



Bug



Vulnerability



Malware



Google patches critical Android vulnerability that allows for elevated privileges

Dozens of vulnerabilities fixed in Xiaomi, Google Android flavors ... slowly  
Oversecure details bugs spotted and stamped since private disclosure

g the EvilVideo telegram for Android

lusa banking trojan returns to steal your words and cash — how to stay safe  
Anthony Spadafora published June 25, 2024  
lightweight variant can capture screenshots and use full-screen overlays

mysterious family of malware  
Play for years

ESET researchers discovered a zero-day Telegram for Android exploit that allows sending malicious files disguised as videos

Samsung says a critical security patch is making its way to Galaxy devices soon

New 'Brokewell' Android Malware Spread Through Fake Browser Updates

Apr 26, 2024 Ravie Lakshmanan

Mobile Security / Cybercrime

Just a fraction of 2024!

RAFEL RANSOM  
June 20, 2024  
Research

this notorious lets hackers remain how to stay safe  
By Anthony Spadafora published April 3, 2024  
New upgrade lets hackers bypass the

Have trouble with TikTok? Here's how to stay safe  
By Adam Birney • June 28, 2024

Elizabeth Montalbano  
July 17, 2024

New Vulnerability Discovered in Android Devices Sparks Security Concerns  
By Michał Nawrocki  
2024-06-24

Android Malware in mobile Trojans like TeaBot and others, n

Upgrade allowed make sure to check carefully  
By Sead Fadić published 18 hours ago  
Updated version of Mandrake Android malware

Code Execution Vulnerability Patched in

Android's February 2024 security patches resolve 46 vulnerabilities, including a critical remote code execution bug.



# Two ways to do program analysis

1

Dynamic Analysis



Static





\$2 000 000 000 000

Yearly cost of **poor quality software** in the USA



# Why it matters?

1. Help catch errors early
2. Improve software reliability and quality
3. Improve Security
4. Optimize Programs
5. At the end: good for end-users!



# Static Analysis



Static Analysis examines code  
*without executing it*



# grep



# grep

```
jordan:/$ grep toto myfile.c
```

```
12:  int toto = 42;
19:  if (toto > 0) {
37:  // remember to ask toto about this implementation detail
43:  // TODO: handle edge case when toto is null
51:  log("toto reached step 3");
58:  printf("toto is ready\n");
76:  result = process(toto);
101:  config.set("owner", "toto");
118:  // NOTE: hardcoded value for testing with toto's config
```



# It is about asking a question

Does this program ever divide by zero?

Can user input reach a SQL query without sanitization?

Can  $x = 2$  hold at this program point?

Is there a possible integer overflow here?

Does the binary contain known malicious code patterns?

Can this array access go out of bounds?

## Does the program leak a personal data?

Can this lock lead to a deadlock?

Can this recursive function terminate?

Does this function always return a value?

Is the program writing to arbitrary files or locations?

Is variable  $x$  always initialized before use?

Does this function allow buffer overflow with crafted input?

Are two variables aliases at this point?



But, there is a problem...

We cannot answer every question  
about every program, *even in theory*.

Static analysis ***must*** approximate!



# Why not?

**Alan Turing** proved in 1936 that some problems, such as deciding if any program halts, are **undecidable**.

Later, **Henry Rice** generalized this with **Rice's Theorem**: any non-trivial property of a program's behavior is undecidable.





# A simple example

```
def run_forever():  
    while True:  
        pass
```

```
def maybe_run(input):  
    if input:  
        run_forever()  
    else:  
        return "done"
```

Does  
"maybe\_run"  
terminate?





# So... Are we doomed?

We cannot answer everything...  
...Properties of programs are undecidable...  
...Sounds bad!

But no!  
We **approximate**.

Static analysis gives ***sound*** answers through ***approximation***.



# Static Analysis





```
#include <stdio.h>
```

```
int main() {  
    printf("Hello, World!\n");  
    return 0;  
}
```

**Source code**

```
public class HelloWorld {  
    public static void
```

code:

```
0: getstatic #2  
3: ldc. #3  
5: invokevirtual #4  
8: return
```

```
}
```

**Bytecode**

```
main:  
    push    rbp
```

```
    str]
```

# Or any other representation!

**Binary code**



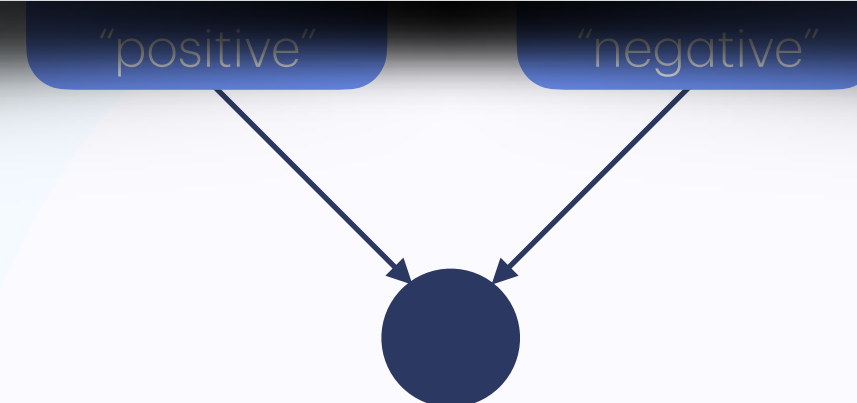
# Building Representations

```
def check_sign(x):
```

```
    if x > 0:
```

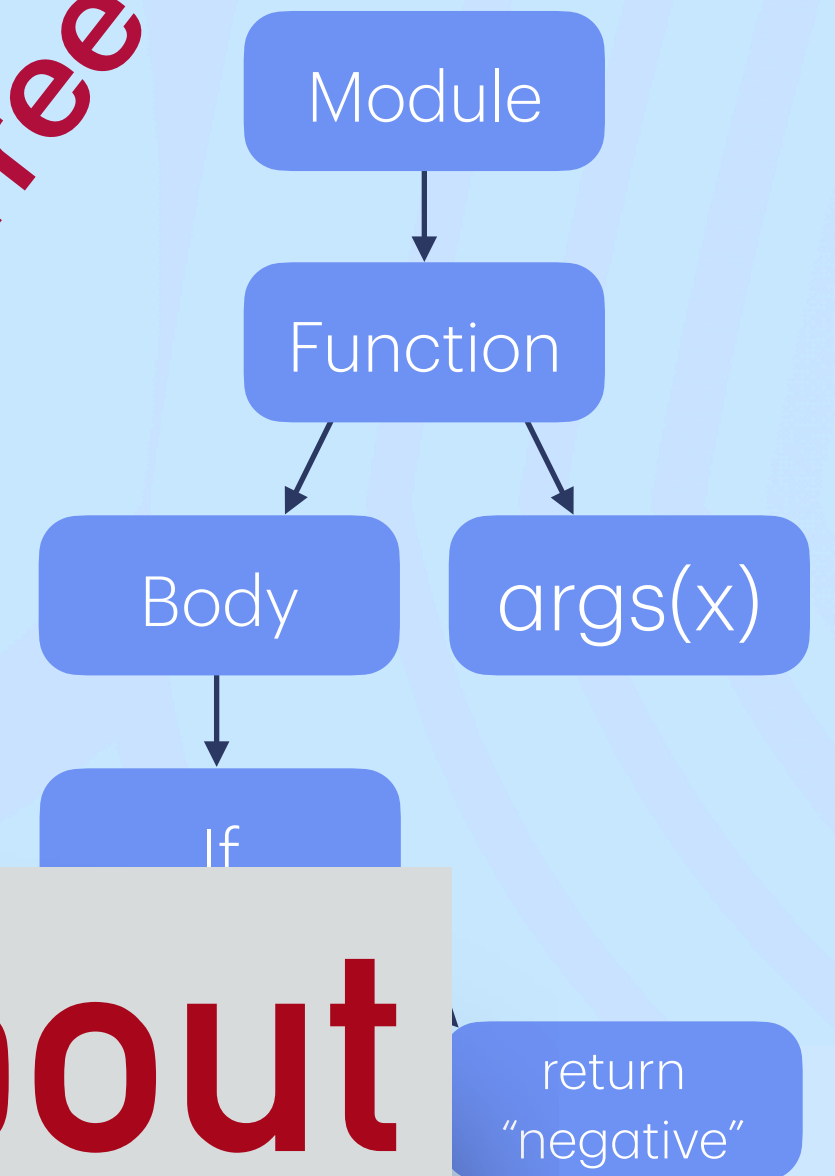
```
        el
```

**Then, the analyzer reasons about possible behaviors**



**Control Flow Graph**

**Abstract Syntax Tree**





# The code is never run

The analyzer needs to ***conservatively*** estimate what ***could*** happen



# Soundness of Static Analysis

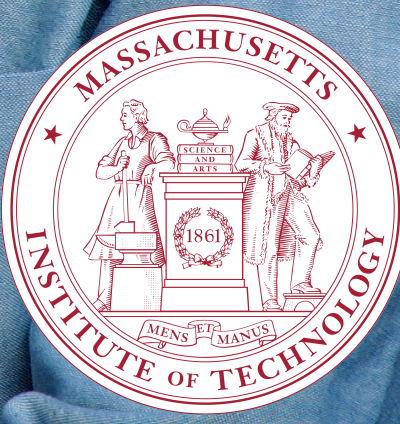
“Static analysis examines program code and reasons over **all possible behaviors** that might arise at run time”\*

“Typically, static analysis is **conservative and sound**”\*

“Soundness guarantees that analysis results are an **accurate description of the program’s behavior**, no matter on what inputs or in what environment the program is run”\*

*\*Ernst, Michael D. "Static and dynamic analysis: Synergy and duality."  
WODA 2003: ICSE Workshop on Dynamic Analysis. 2003*

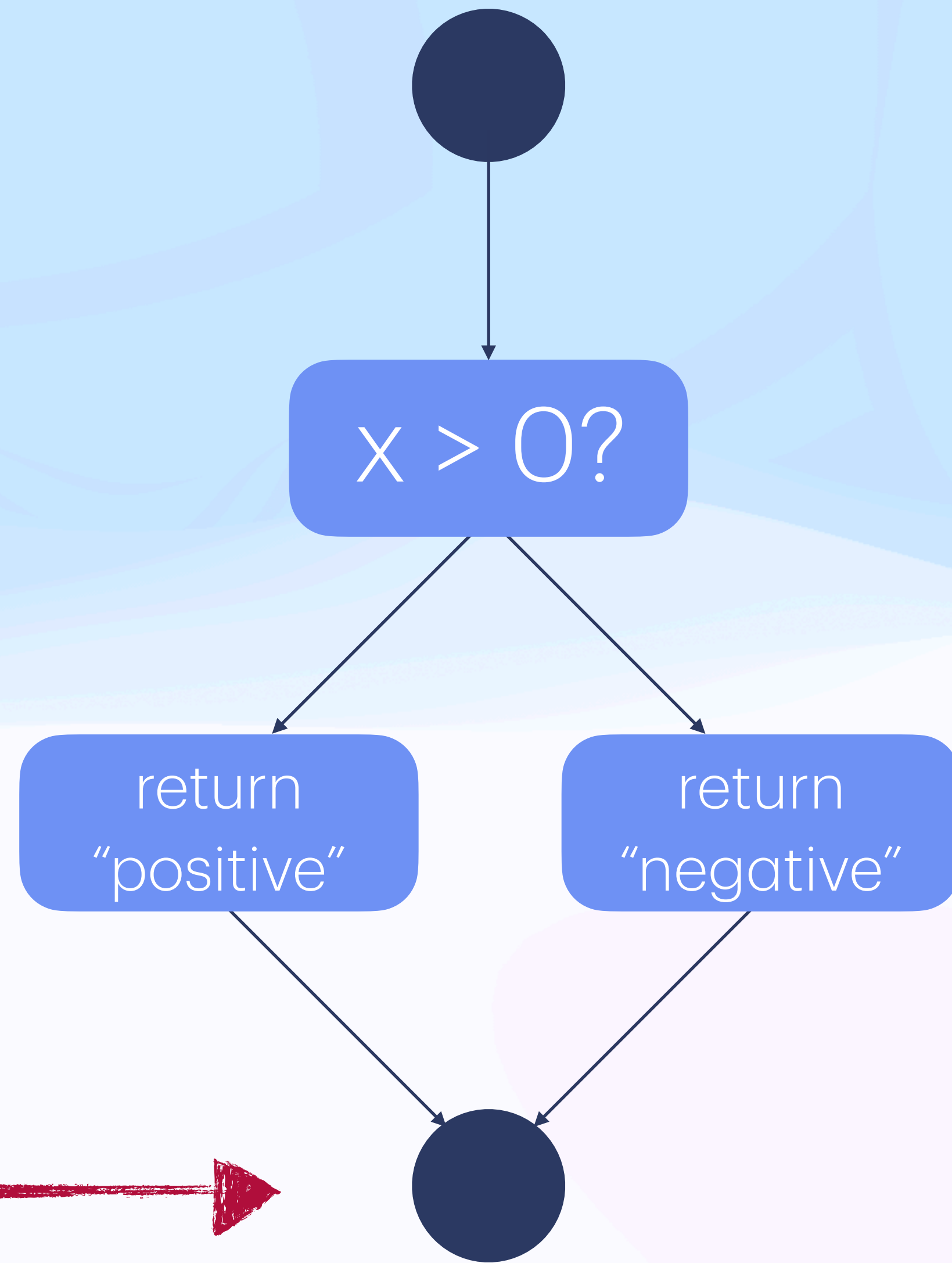
Dr. Jordan Samhi





```
def check_sign(x):  
    if x > 0:  
        return "positive"  
    else:  
        return "negative"
```

Conservative means that here,  
a static analyzer must say: program  
returns "positive" or "negative"





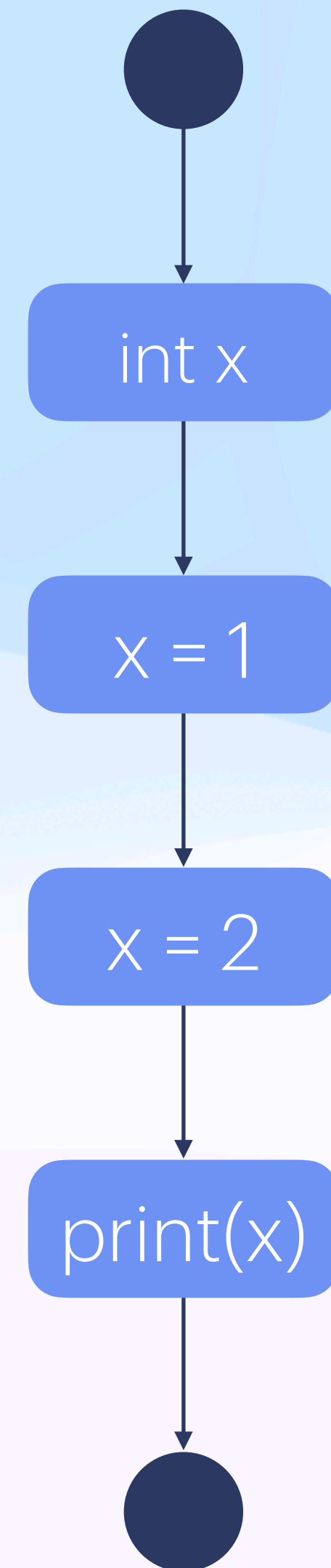
# Reasoning

Question is: what value `x` holds when it is printed?

```
int x;  
x = 1;  
x = 2;  
print(x);
```

2

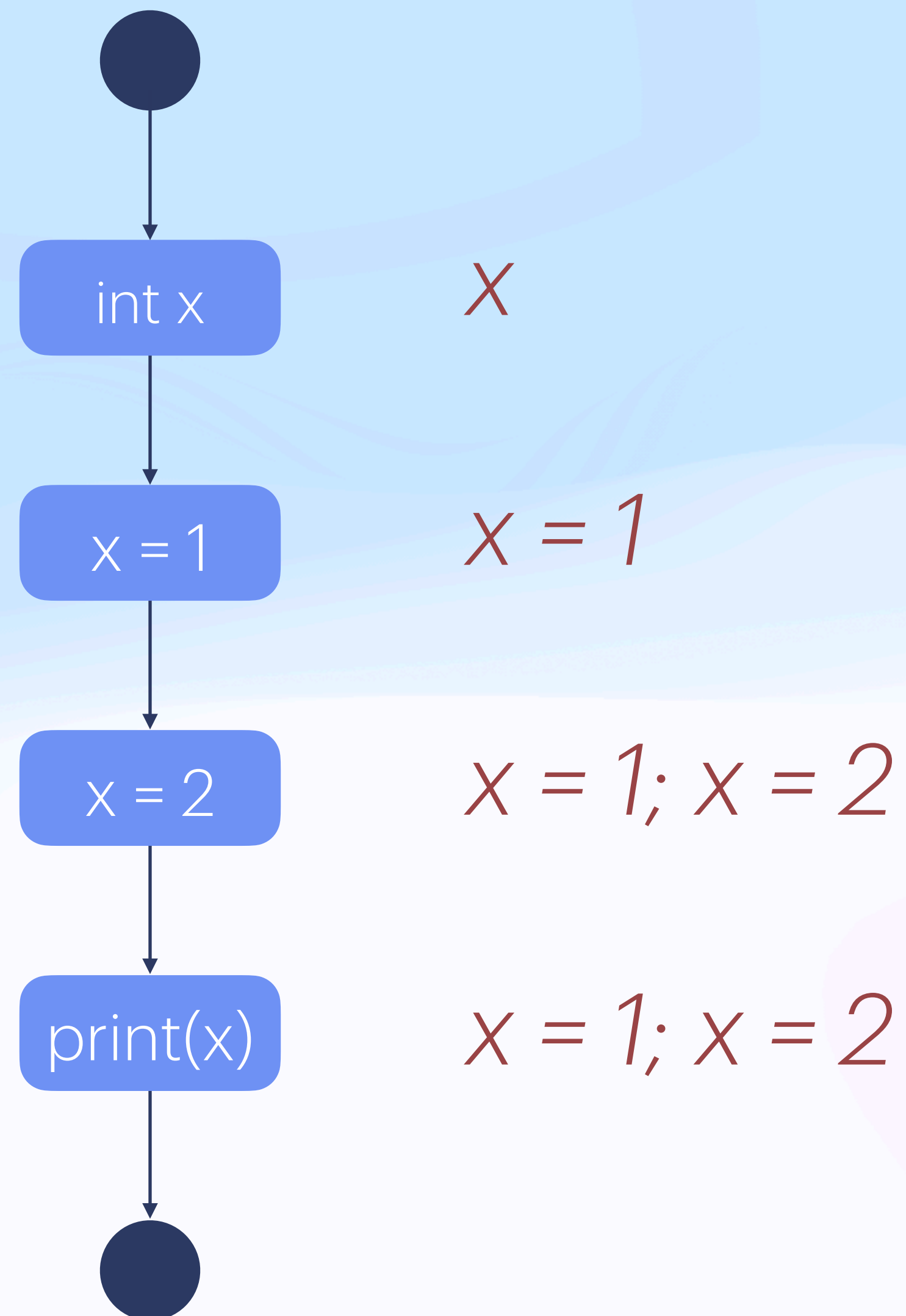
ways to answer





# Flow insensitive

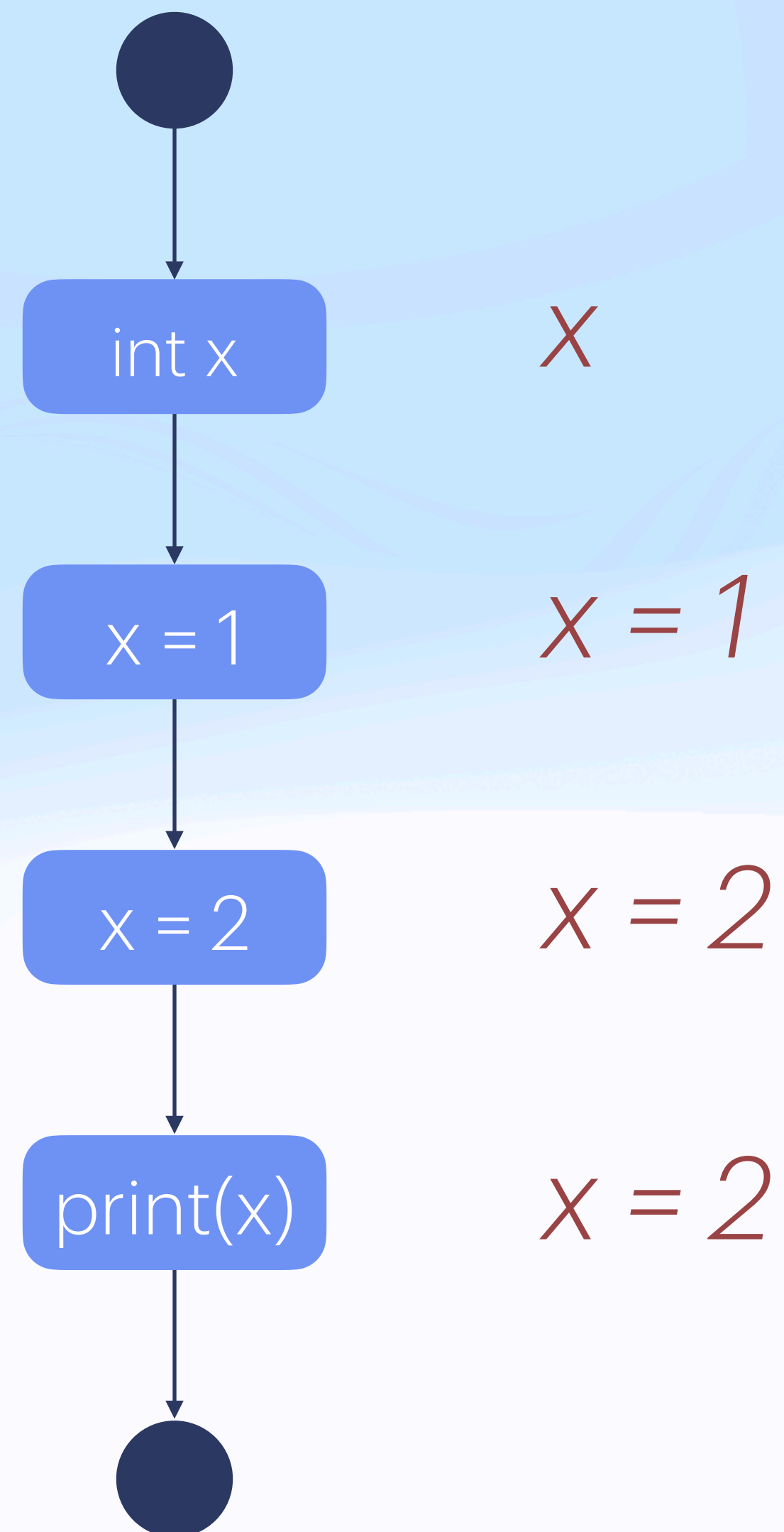
Answer is: number 1  
will be printed **OR**  
number 2 will be  
printed





# Flow sensitive

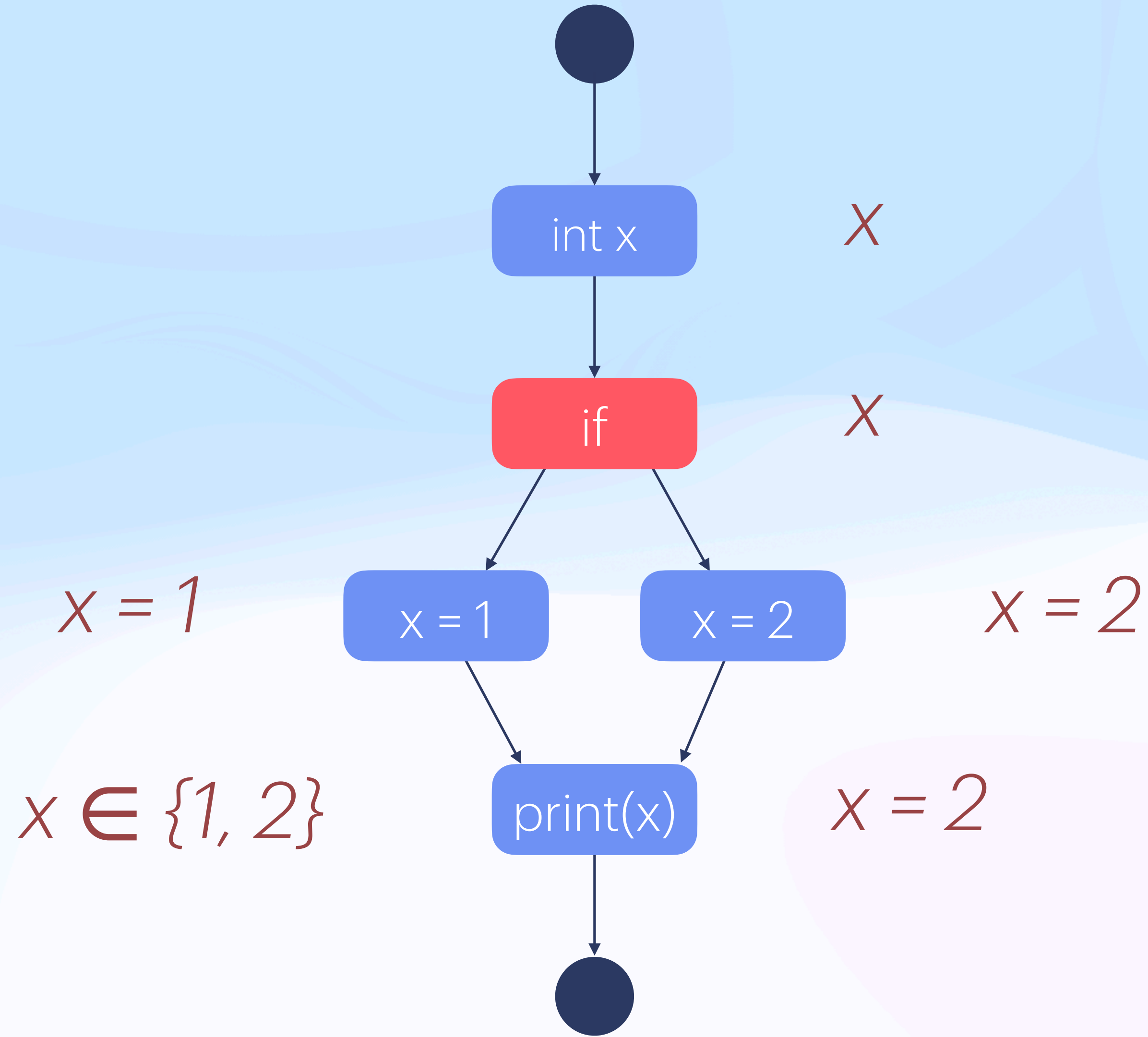
Answer is: number 2  
will be printed





# Branches

```
int x;  
if (some_condition) {  
    x = 1;  
} else {  
    x = 2;  
}  
print(x);
```



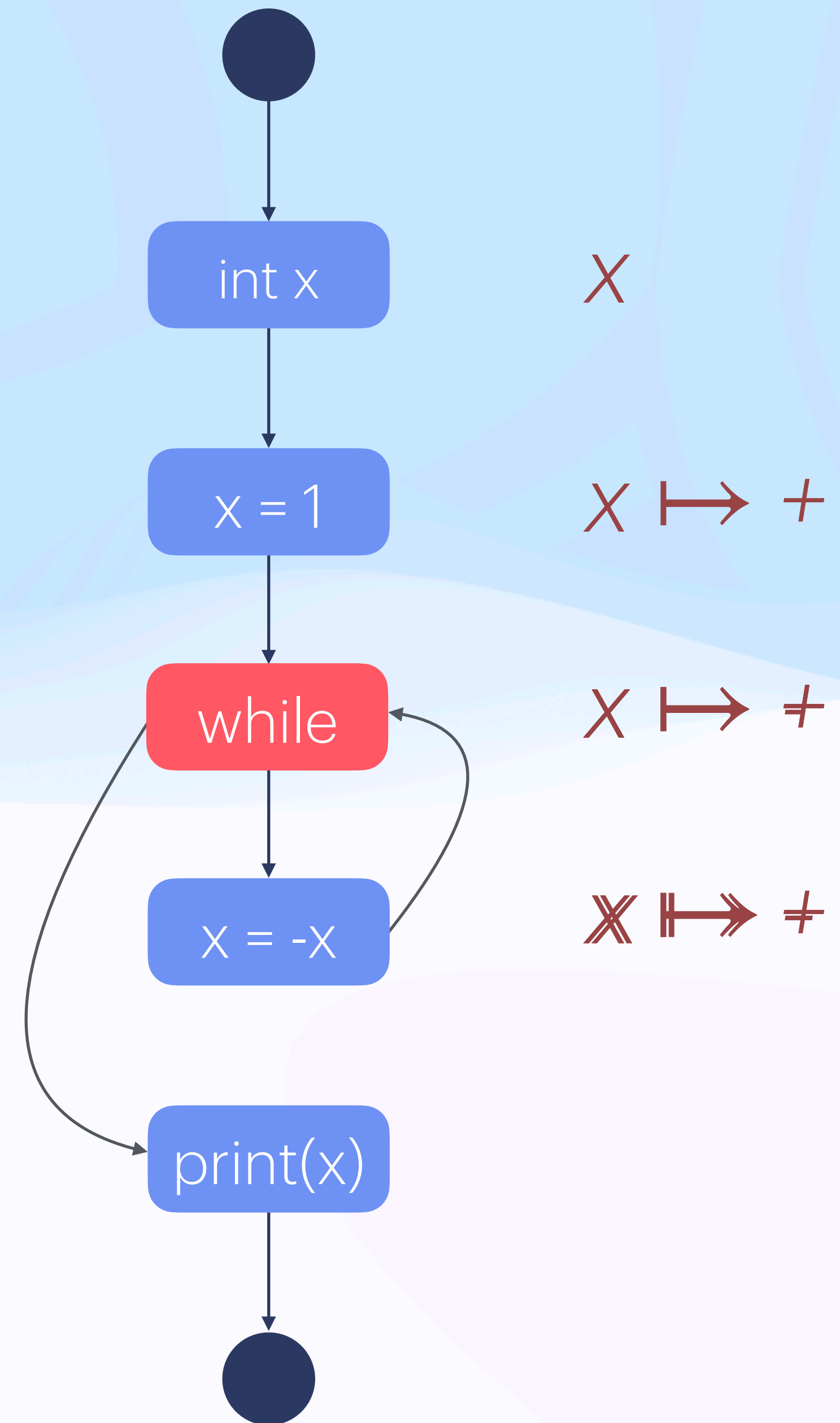


# Loops

```
int x;  
x = 1;  
while (1) {  
    x = -x;  
}  
print(x);
```



The analysis will loop  
**forever!**

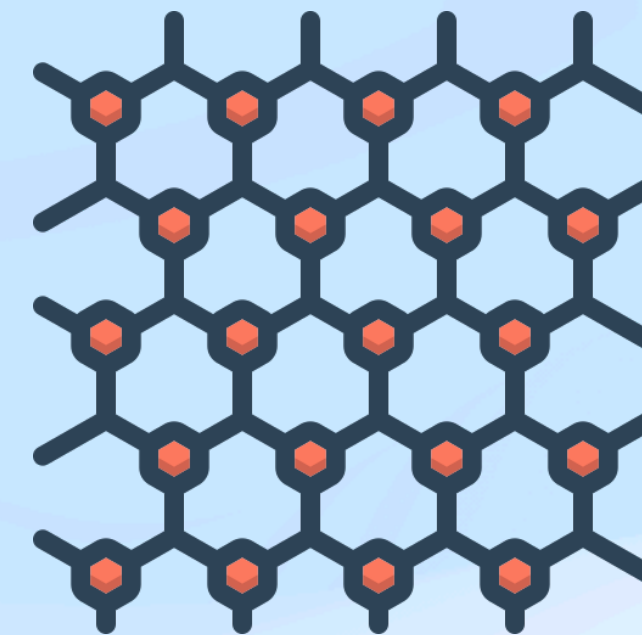




# Ensuring termination: a whole Theory



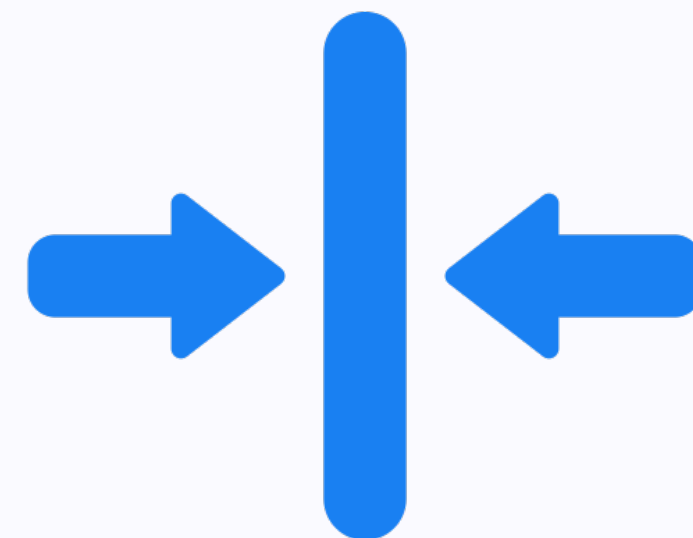
Monotonic Transfer Function



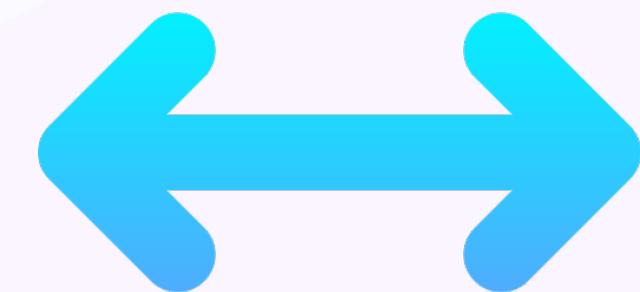
Finite-Height Lattices

$$f(x) = x$$

Fixpoint Iterations



Narrowing Techniques



Widening Techniques



# The Theory behind static analysis

Monotonicity:  $x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$

Transitivity:  $x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$

Distributivity:  $f(x \sqcup y) = f(x) \sqcup f(y)$

Fixpoint:  $\text{lfp}(f) = \bigsqcup_{i \geq 0} f^i(\perp)$

Widening:  $\nabla(x, y) = \begin{cases} y & \text{if } x \sqsubseteq y \\ \top & \text{otherwise} \end{cases}$

$f_\pi = f_{n_k} \circ f_{n_{k-1}} \circ \dots \circ f_{n_1}$

$$\text{IN}[n] = \bigsqcup_{p \in \text{pred}(n)} \text{OUT}[p]$$

$$\text{OUT}[n] = \text{GEN}[n] \cup (\text{IN}[n] \setminus \text{KILL}[n])$$

$$\text{MOP}[n] = \bigsqcup_{\pi \in \text{Paths}(n)} f_\pi(\perp)$$

$$\text{MFP} = \text{lfp}(F) = \bigsqcup \{x \in L \mid F(x) \sqsubseteq x\}$$

$$X \sqcup Y = \left\{ \langle \sigma, x \sqcap y \rangle \mid \langle \sigma, x \rangle \in X \wedge \langle \sigma, y \rangle \in Y \right\} \cup \left\{ \langle \sigma, x \rangle \mid \langle \sigma, x \rangle \in X \wedge \forall z \in L, \langle \sigma, z \rangle \notin Y \right\} \cup \left\{ \langle \sigma, y \rangle \mid \langle \sigma, y \rangle \in Y \wedge \forall z \in L, \langle \sigma, z \rangle \notin X \right\}$$



The Theory behind static analysis

# Abstract Interpretation



Have we seen it all?

```
x = 0  
foo()  
print(x)
```

```
def foo():  
    x = 42  
    print(x)
```

●

# Easy!

*Just have to jump to the foo()  
function right?*

↓

●



# Now consider this example

```
class Animal {  
    void walk() {  
        System.out.println("Animal walks");  
    }  
}
```

```
class Dog extends Animal {  
    void walk() {  
        System.out.println("Dog walks");  
    }  
}
```

```
class Cat extends Animal {  
    void walk() {  
        System.out.println("Cat walks");  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Animal a;  
        if (Math.random() > 0.5) {  
            a = new Dog();  
        } else {  
            a = new Cat();  
        }  
        a.walk();  
    }  
}
```



***Cat or Dog that walks here?***



# Can we just jump to walk()?

At a.walk(), which walk() gets executed?

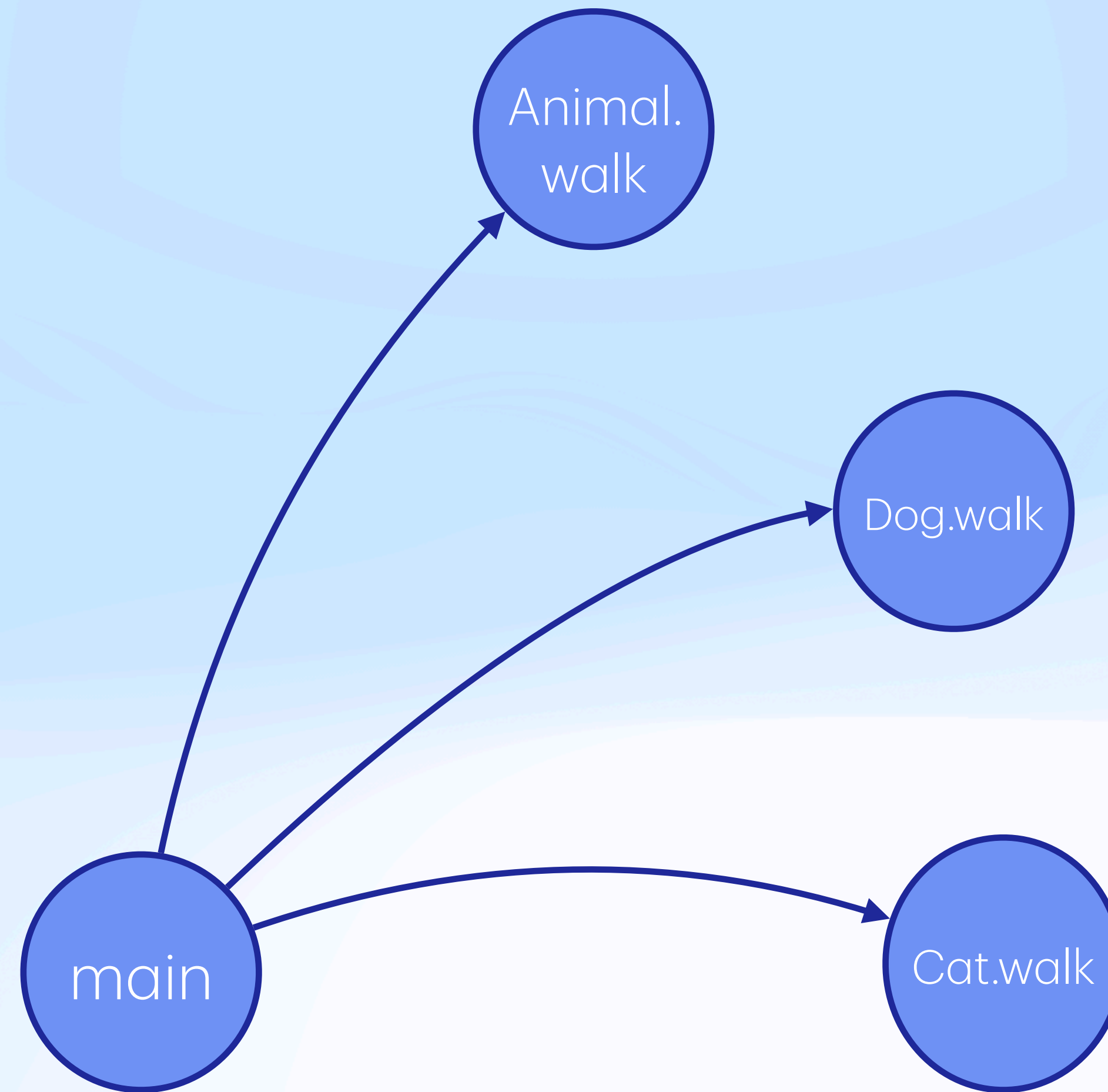
- Dog.walk()?
- Cat.walk()?
- Animal.walk()?
- Depends on runtime behavior!
- Which cannot be predicted!







# Call Graph





# Do not think call graph construction is easy!

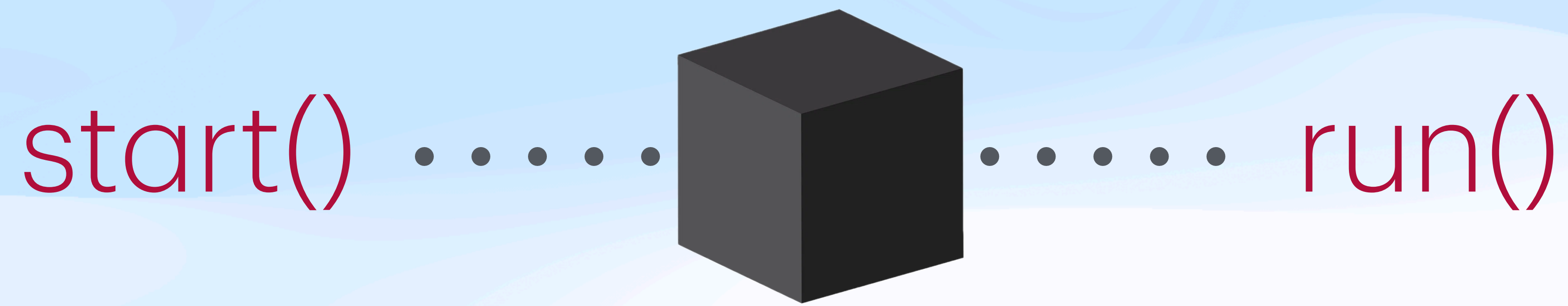
```
public class Main {  
    public static void main(String[] args) {  
        Thread t = new MyThread();  
        t.start();  
    }  
}  
  
public class MyThread extends Thread {  
    @Override  
    public void run() {  
        // do stuff  
    }  
}
```



Will method **start()** be executed?



# Do not think call graph construction is easy!



There is a  
**discontinuity!**



# Different Call Graph construction algorithms





# Call Graph construction challenges

Dynamic Dispatch

Multi-Threading

Under-  
approximation

**These mechanisms  
bring unsoundness!**

GU

Aliasing

FRAMEWORKS

Lambdas

Callbacks

Over-  
approximation

Function Pointers



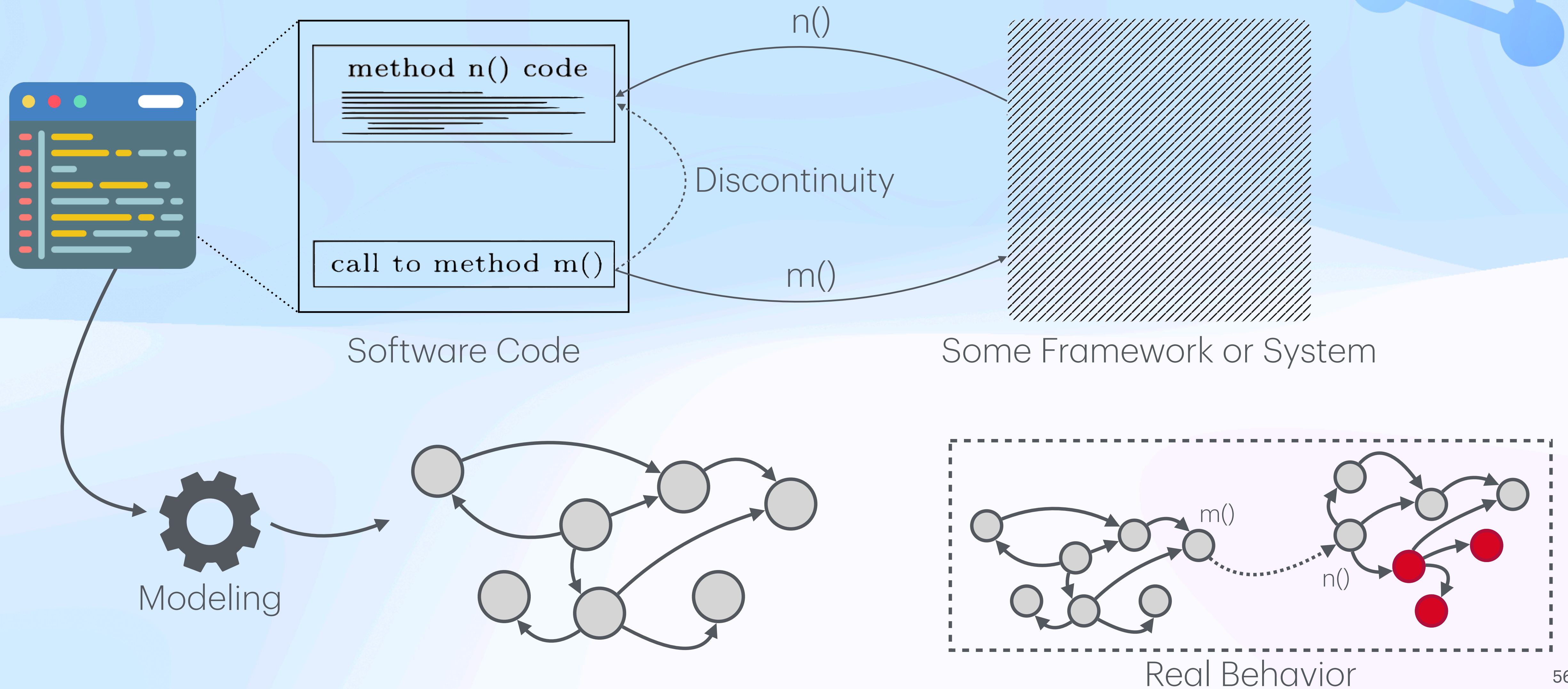
# Static Analysis

1  
**UNSOUND**  
Model

2  
**SOUND**  
Analysis



# Soundness of Models



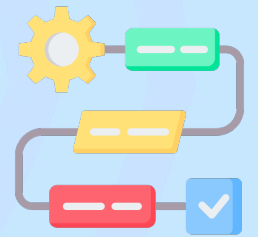


If static analysis tools do not model  
th **and therefore...** is

**...the analysis is unsound!**



# Some Static Program Analyses



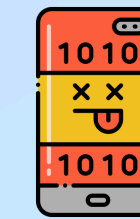
## Dataflow Analysis

Track facts along the Control Flow Graph



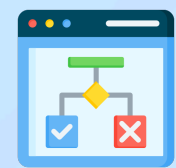
## Symbolic Execution

Executes code with symbolic inputs to explore paths and constraints



## Dead Code Analysis

Detects unreachable or unused code



## Control Flow Analysis

Builds the possible paths and call graph



## Pointer/Alias Analysis

Determines whether two expressions may reference the same memory location



## Dependency Analysis

Determines data dependencies among variables/statements

## int Type Checking

Verifies that variables and expressions follow the language's type rules



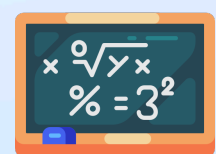
## Taint Analysis

Tracks how untrusted inputs propagate through the program



## Model Checking

Verifies properties (e.g., assertions, invariants) over program models



## Abstract Interpretation

Approximates program behavior using abstract domains and fixpoints



## Constant Propagation

Determines values known at compile time and simplifies expressions



## Effect Analysis

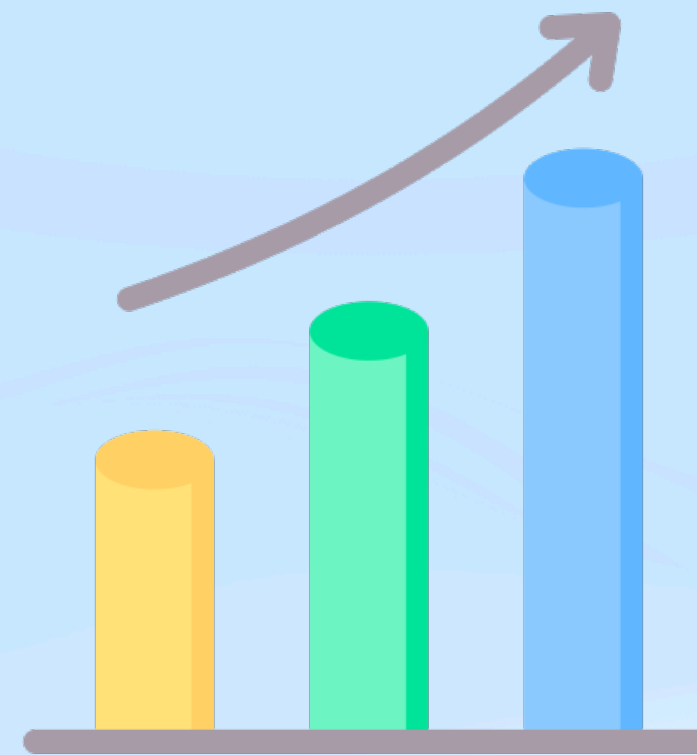
Tracks side effects (e.g., state changes, I/O operations)



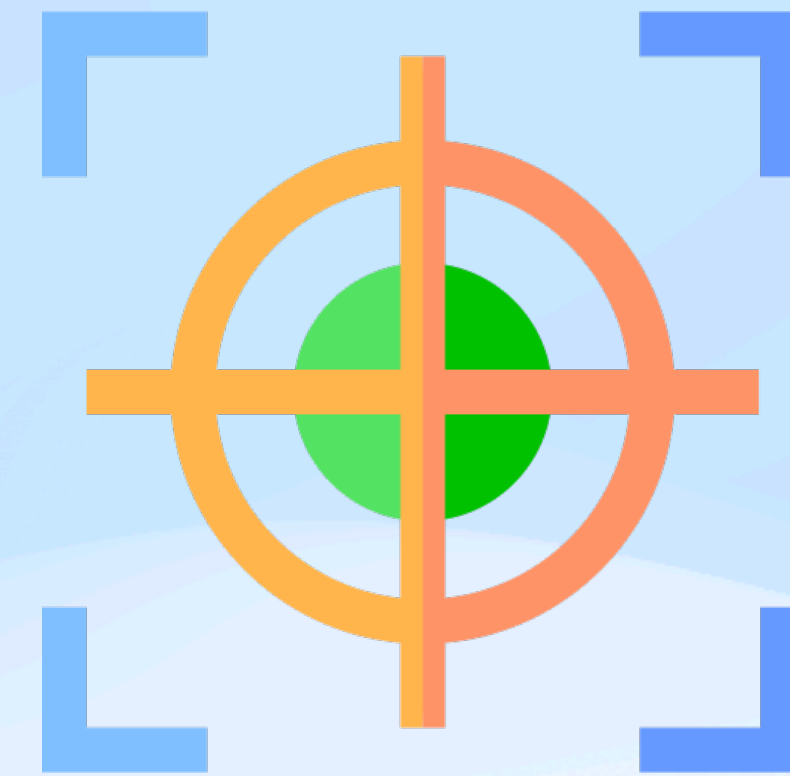
# Challenges of Static Analysis



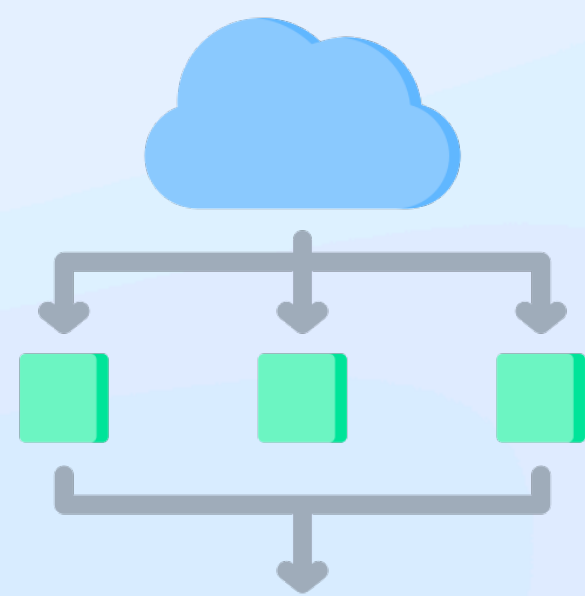
Undecidability



Scalability



Precision



Concurrency



Features



False Positives



Incomplete Context



# Success Stories



## - **Google's OSS-Fuzz<sup>1,2</sup>**

- Found over 10,000 security vulnerabilities and 36,000+ bugs in libraries (e.g., libpng, OpenSSL).



## - **Facebook's Infer<sup>3</sup>**

- Open-source static analyzer for Java, C++, etc.
- Catches null dereferences, memory leaks, and race conditions in production code.



## - **Microsoft Static Driver Identifier<sup>4</sup>**

- Found thousands of bugs before deployment.



## - **Coverity Scan<sup>5</sup>**

- Detected critical bugs with low false-positive rates in open-source projects



## - **Airbus: Formal verification of flight code<sup>6</sup>**

- Used abstract interpretation to statically verify safety-critical avionics code.
- Verified absence of runtime errors in fly-by-wire systems.

<sup>1</sup> <https://github.com/google/oss-fuzz>

<sup>2</sup> <https://security.googleblog.com/2023/08/ai-powered-fuzzing-breaking-bug-hunting.html>

<sup>3</sup> <https://engineering.fb.com/2015/06/11/developer-tools/open-sourcing-facebook-infer-identify-bugs-before-you-ship/>

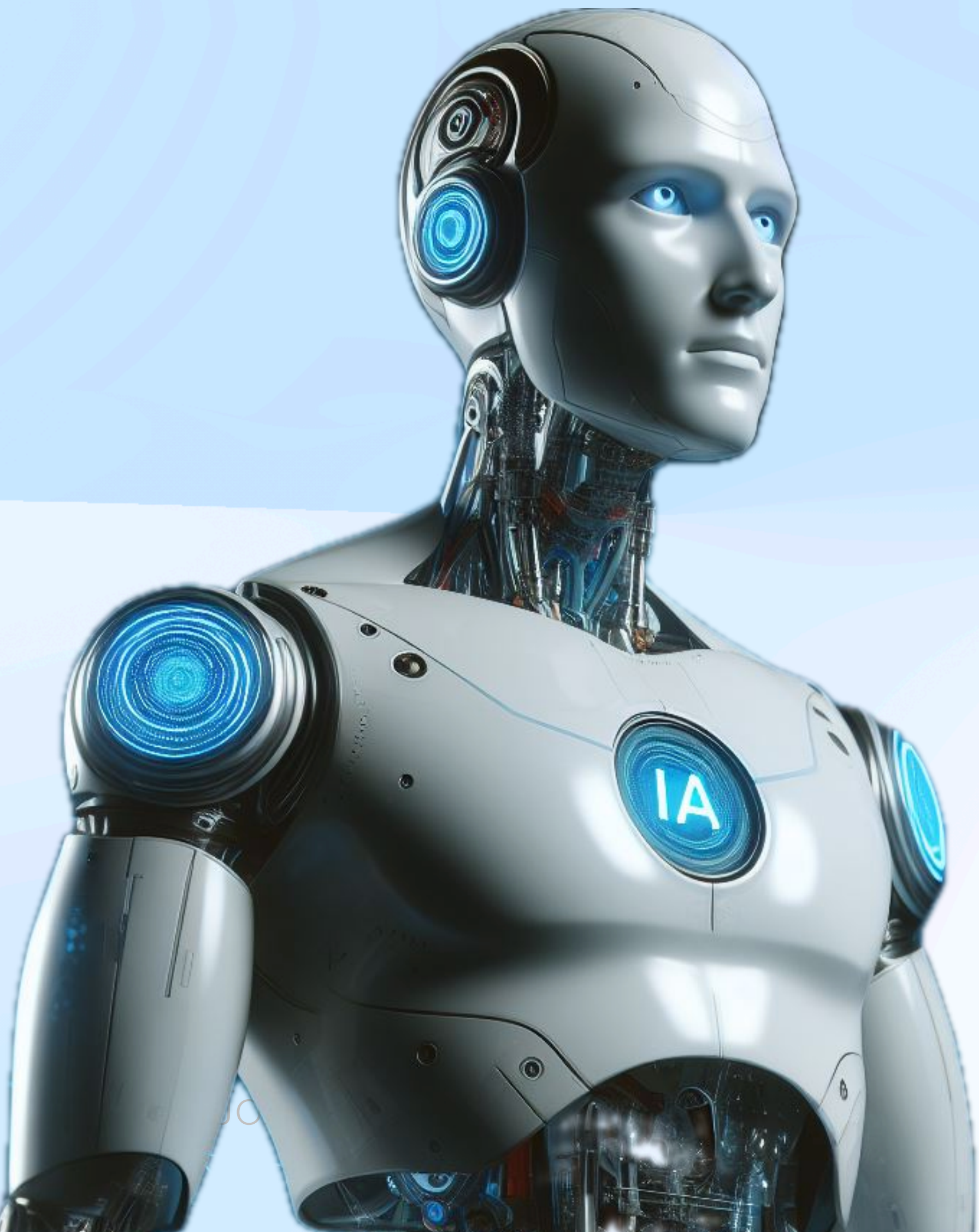
<sup>4</sup> <https://www.wired.com/2005/11/microsofts-secret-bug-squasher/>

<sup>5</sup> <https://techcrunch.com/2012/02/23/with-many-eyeballs-all-bugs-are-shallow/>

<sup>6</sup> <https://www.wired.com/2005/11/microsofts-secret-bug-squasher/>



# Opportunities



Artificial Intelligence **will not** solve  
all static analysis challenges!

**BUT!**

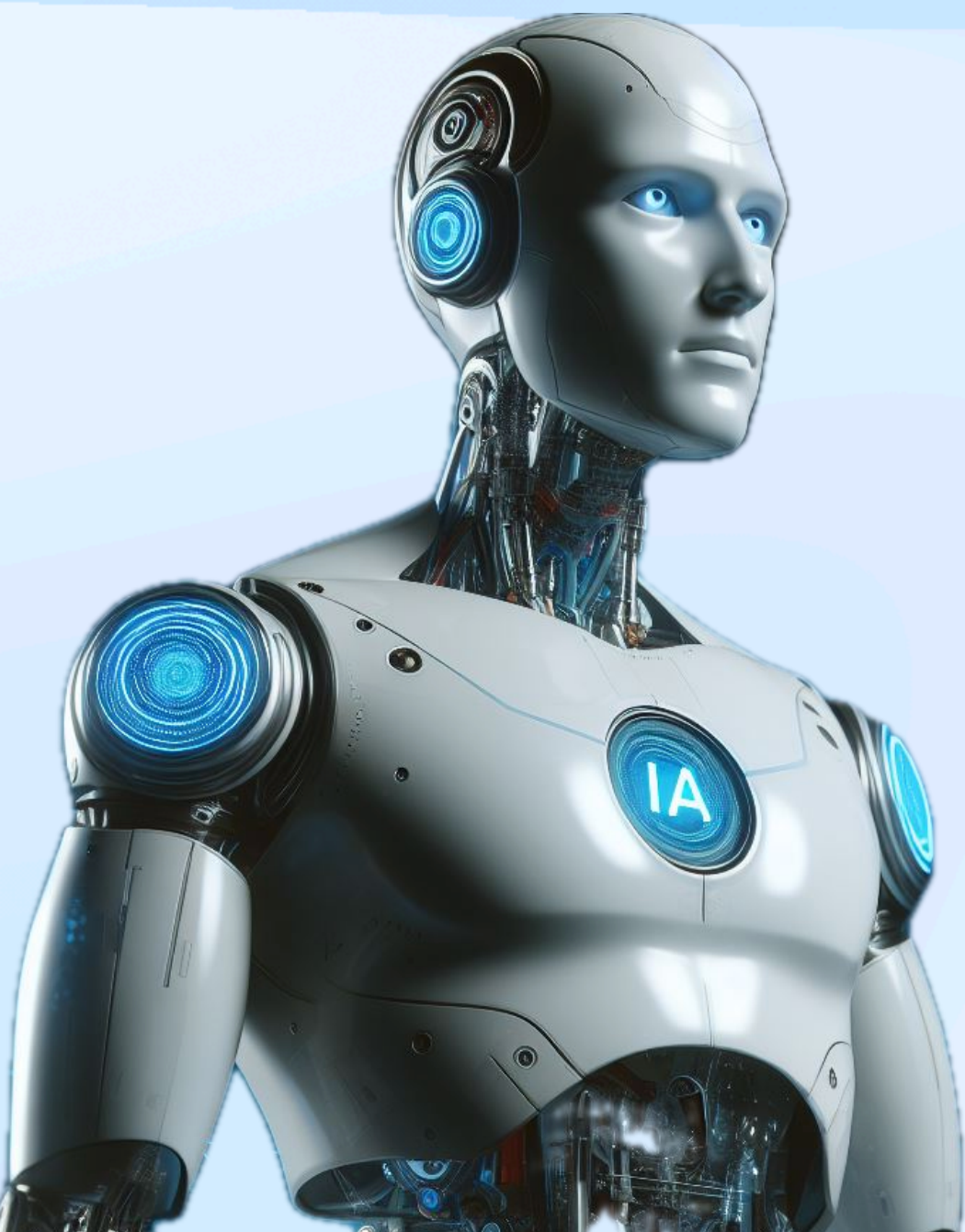
They can, and will help!



# How?

Artificial Intelligence **may help** for these tasks:

- Replace rules by ``reasoning``
- Go beyond syntax to reason about developer intents
- Better call graphs: predict dynamic behavior that static rules may miss
- AI can triage results to reduce false-positives
- AI can predict possible runtime behavior and targets (e.g., likely method target)
- AI will probably help to solve constraints





# Dynamic Analysis



Dynamic Analysis examines software  
by **executing** it and **observing** its  
**behavior** at **runtime**



# Soundness of Static Analysis

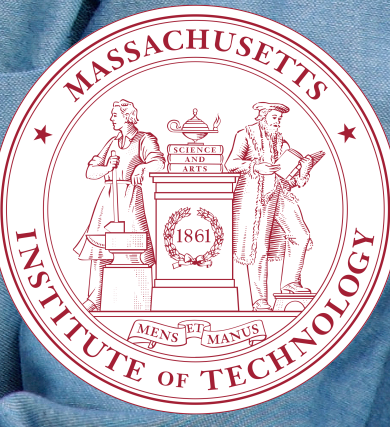
"Dyna  
prog

"Dyna  
or al

**Dynamic analysis is  
precise!**

**\*Ernst, Michael D. "Static and dynamic analysis: Synergy and duality."  
WODA 2003: ICSE Workshop on Dynamic Analysis. 2003**

Dr. Jordan Samhi



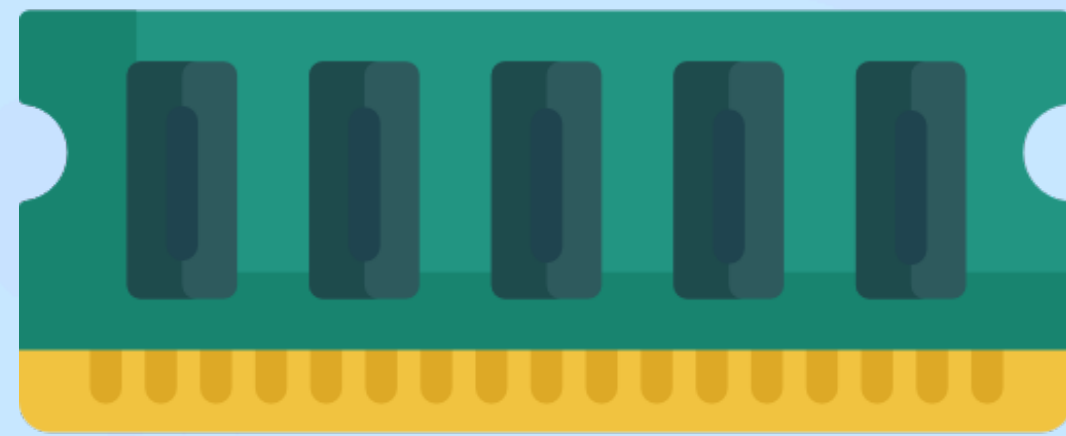




Report



# Run and observe



Memory



Fonction calls



Inputs/Outputs



Performances



Execution Paths



# What do you observe?





Highest degree of  
**precision!**

Dynamic Analysis is often used  
along **instrumentation** tools



# Code Instrumentation

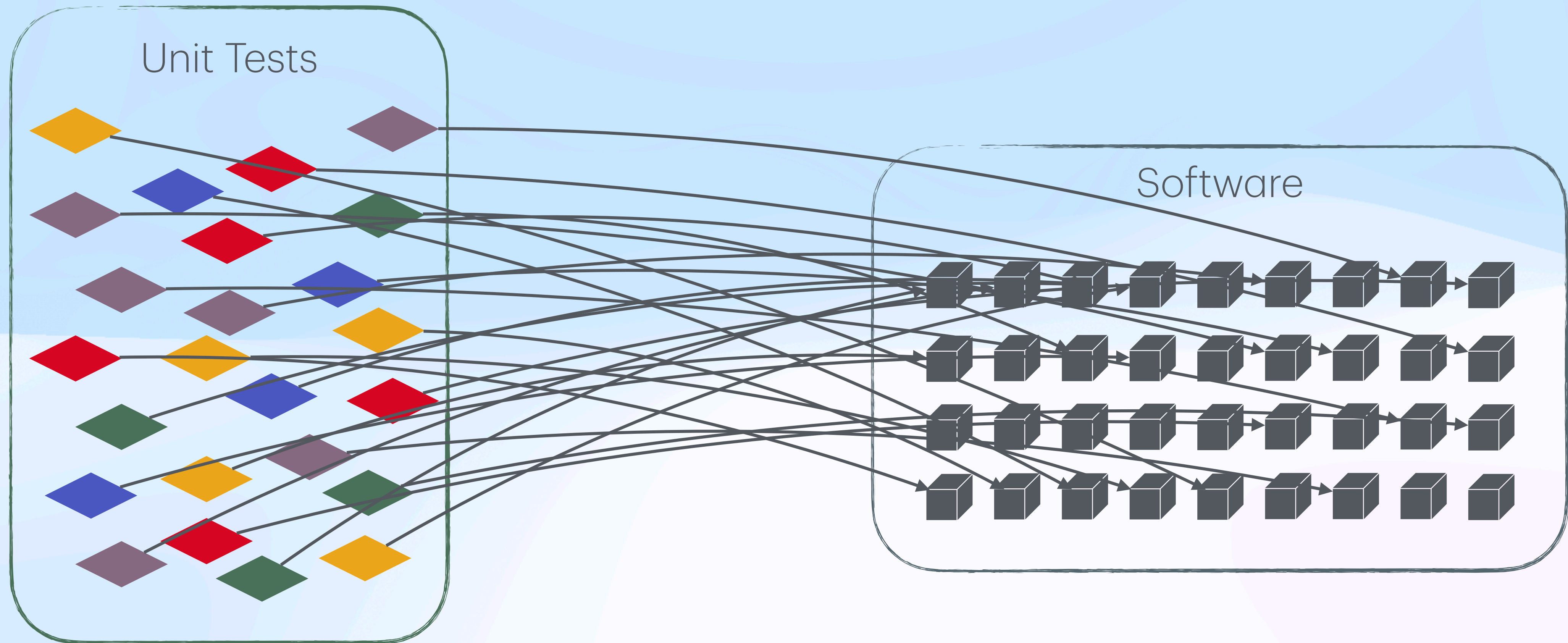
It's the technique of inserting/modifying code to monitor behavior:

- Insert probes before/after function calls
- Track memory reads/writes
- Log system calls or branches
- Monitor execution time or resource use

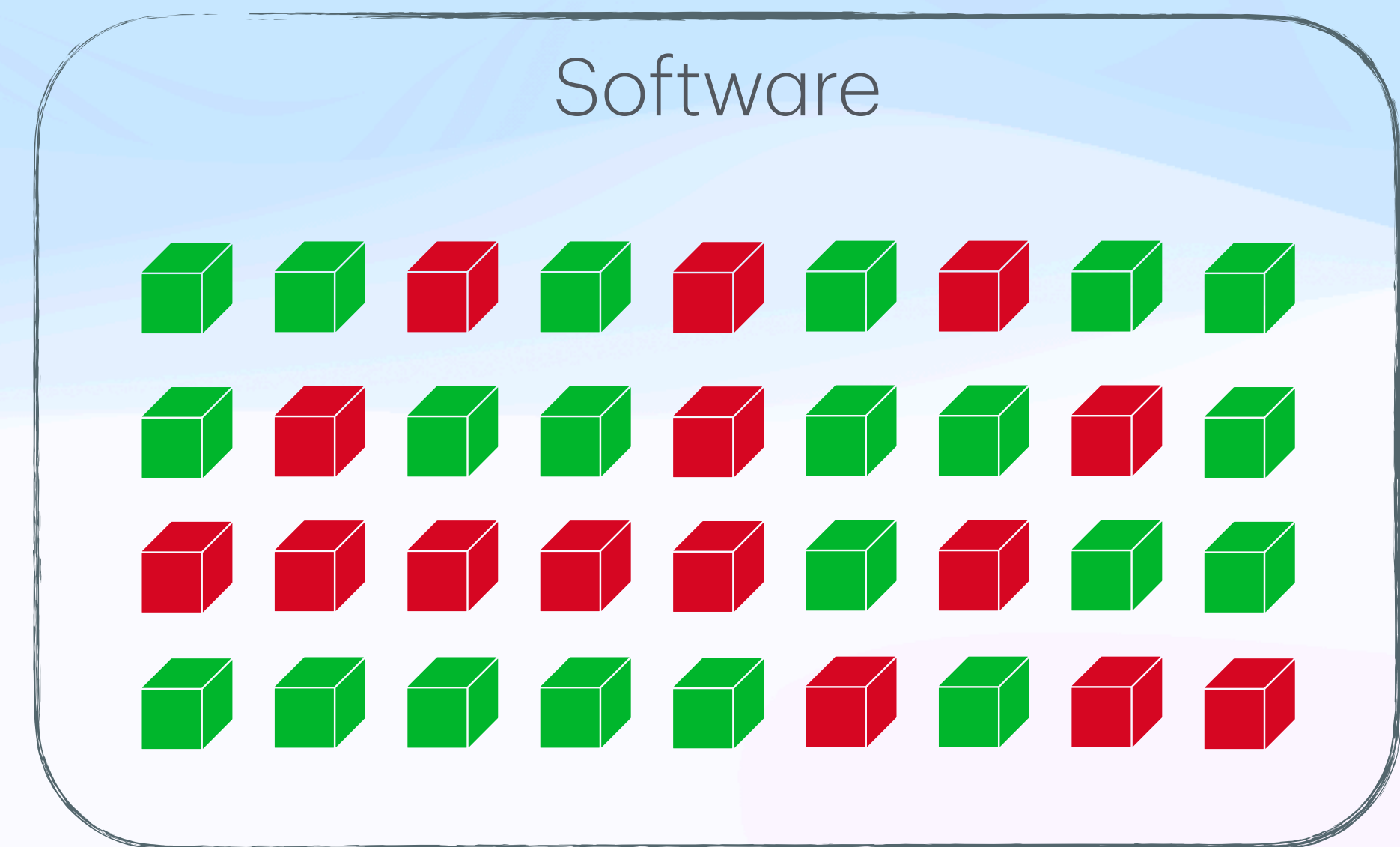
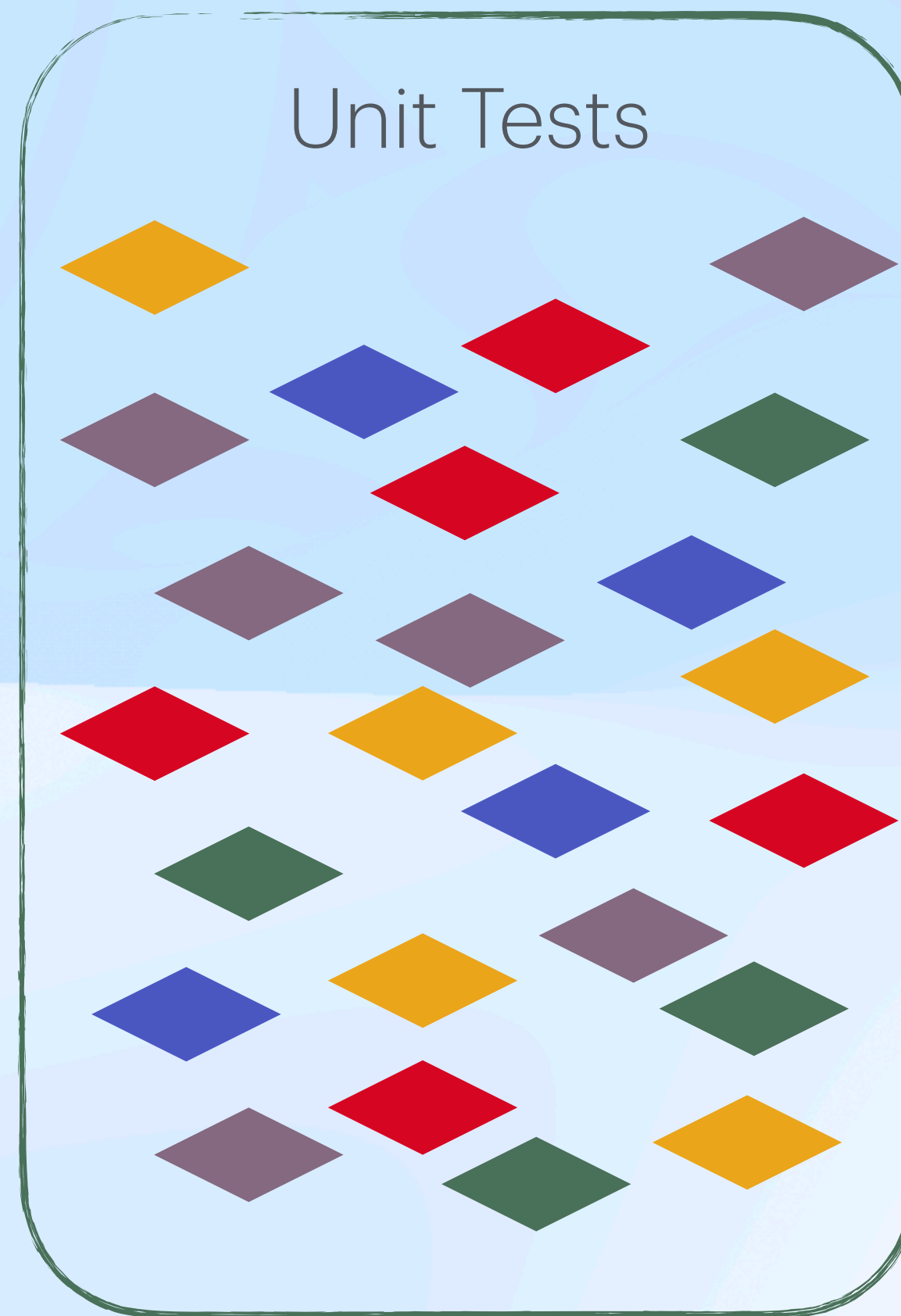
# Some examples of Dynamic Analyses



# Unit Tests



# Unit Tests





# Fuzzing



Potential **bug** found!

# Some Dynamic Program Analyses

## Dynamic Symbolic Execution

Executes the program with symbolic inputs and collects path constraints to explore multiple execution paths.

## Dynamic Taint Analysis

Tracks how untrusted inputs propagate through the program

## Profiling

Measures runtime performance to identify bottlenecks.

## Concurrency Testing

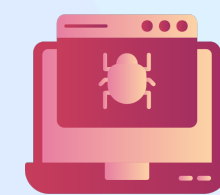
Detects data races, deadlocks, and thread interleaving bugs

## Memory Error Detection

Detects issues like buffer overflows, use-after-free, leaks

## Behavior Monitoring

Records system calls, API usage, or runtime events



## Malware Analysis

Runs binaries in isolated environments to observe behavior, artifacts, and network activity.



## Code Coverage Analysis

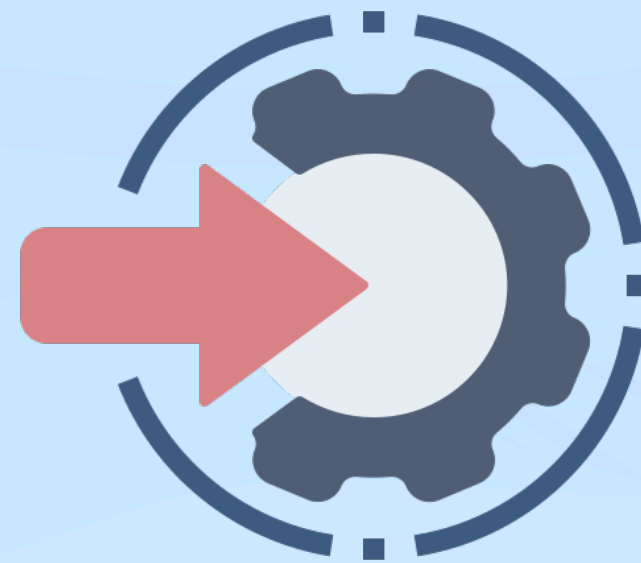
Tracks which lines, branches, or paths are executed during tests.



# Challenges of Dynamic Analysis



Code Coverage



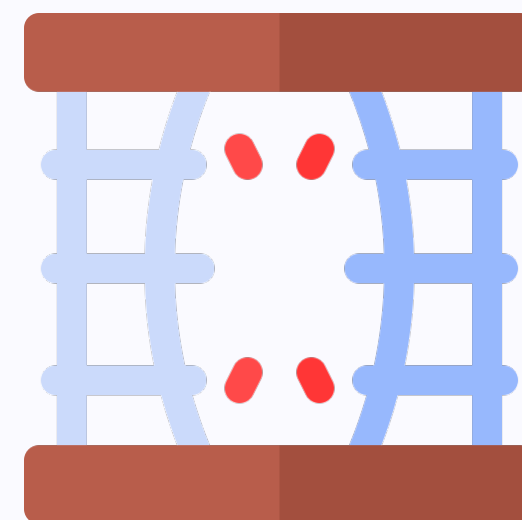
Inputs



Overhead



Non-determinism

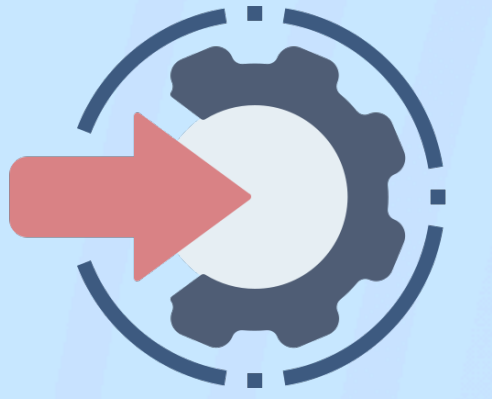


Detection Evasion



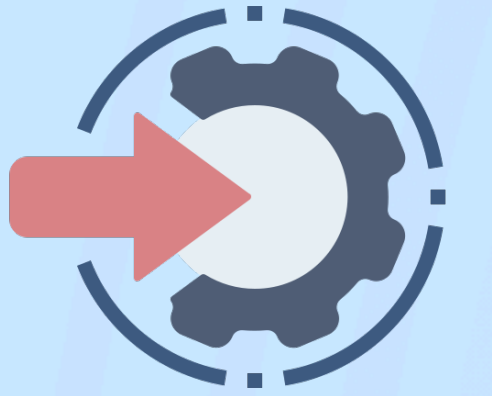
Limited Visibility

# Main Challenge: Input Generation





# Main Challenge: Input Generation



<https://www.fuzzingbook.org/>

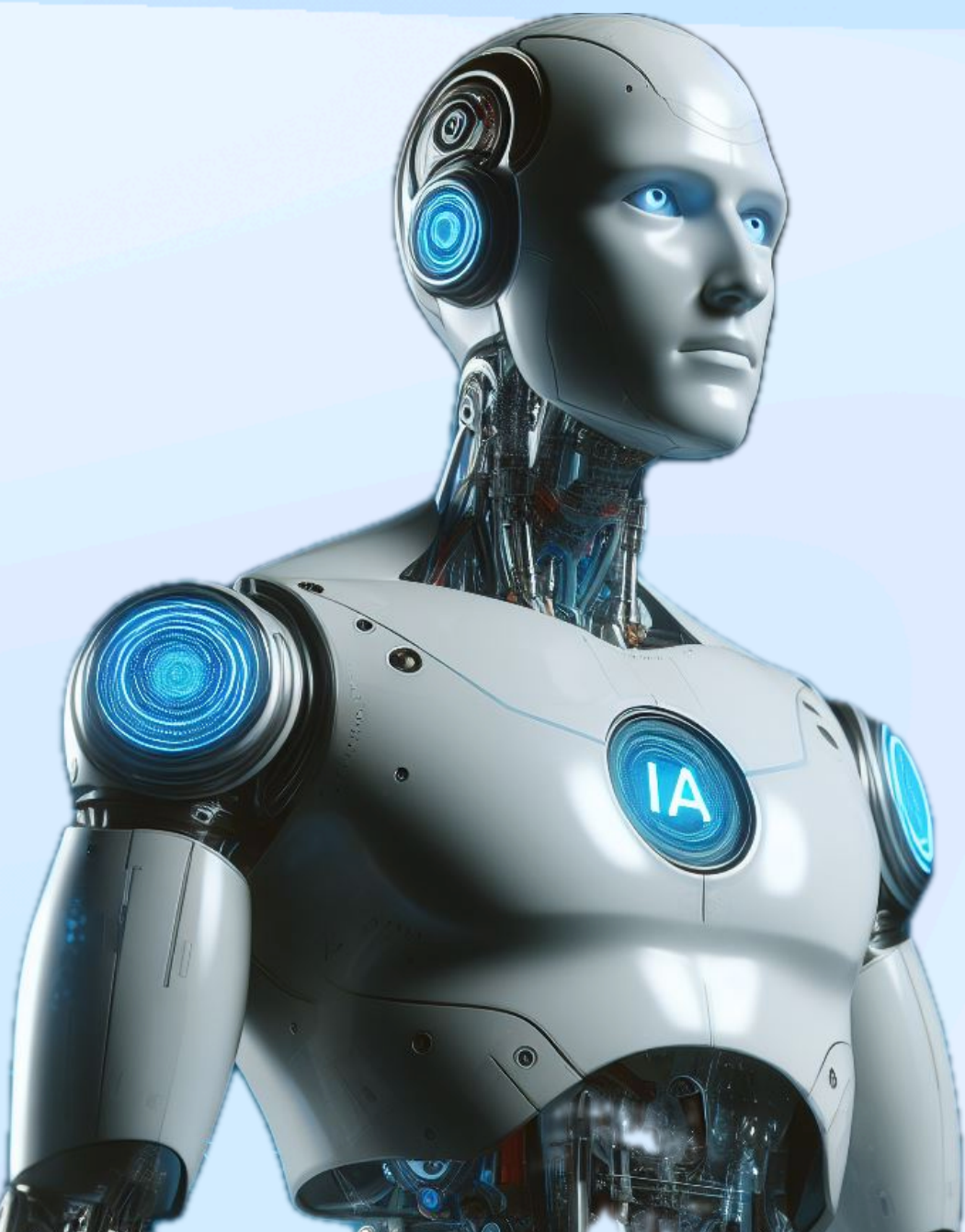
`<http://af>`



# Artificial Intelligence

Artificial Intelligence **may help** for these tasks:

- Suggest meaningful inputs
- Test software like a human
- Can simulate environments
- Can detect evasion and evade evasion
- Help overcome scalability: e.g., solving constraints
- Generate complexe inputs





# Static or Dynamic Analysis?

Static analysis sees all paths but may ~~over-approximate~~

**Why not combine them?**

Dynamic analysis sees real behavior but only for chosen inputs

Each has strengths. Each has limits.

# Hybrid Analysis

*The best of both*



# A simple example: feedback loop

```
import importlib
```

```
def run_plugin(name):
```

```
    module = imp
```

```
    module.run()
```

## Feedback Loop

**Static Analysis** struggles here: ``name`` is dynamic and therefore `module.run()` cannot be resolved.

**Dynamic Analysis** observes at runtime that ``name`` = ``pluginA``



now we know it is calling `pluginA.run()`

# Will AI replace Program Analysis?

# NO



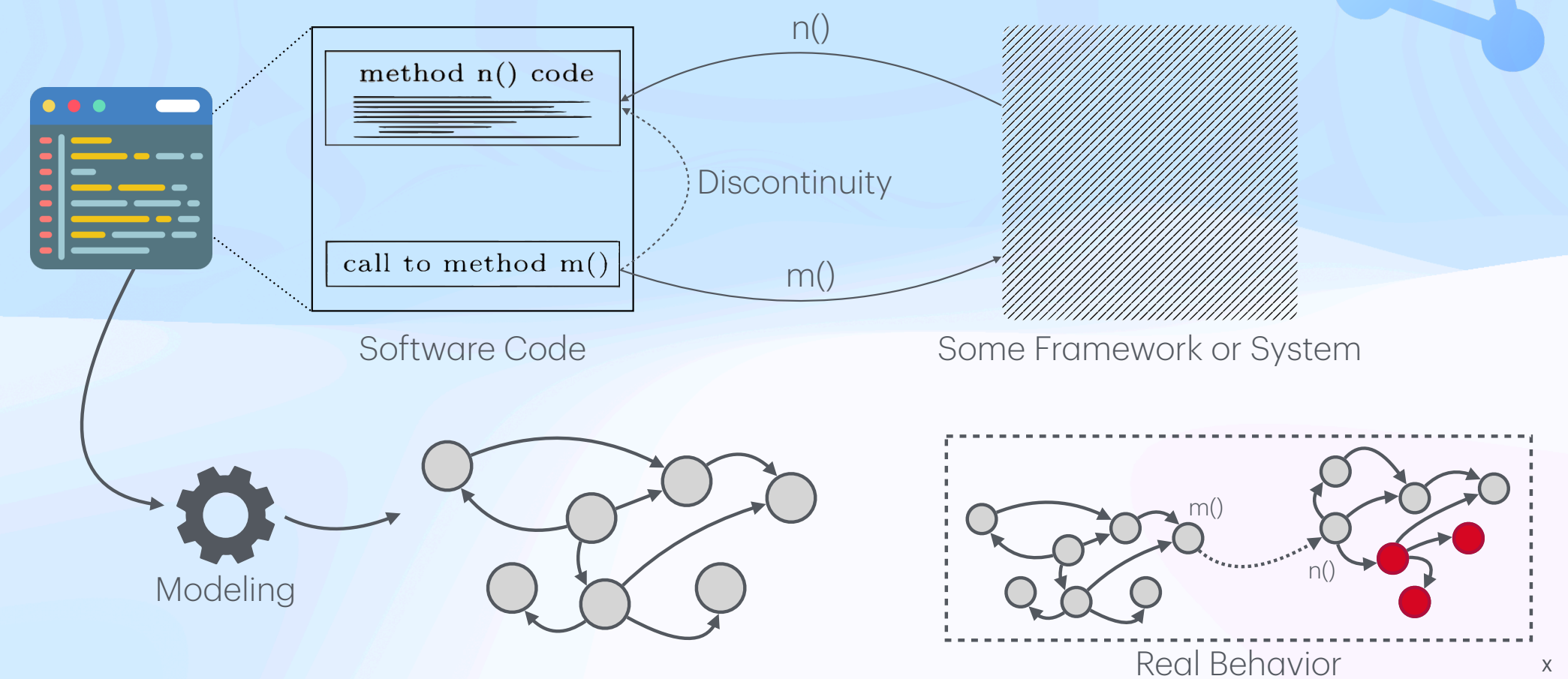
# Automating Software **Security**



Dr. Jordan Samhi

x

## Soundness of Models



# Jordan Samhi

jordan.samhi@uni.lu

jordansamhi.com

**TruX**

**uni.lu**  
UNIVERSITÉ DU  
LUXEMBOURG

**SNT**



## Run and observe



Dr. Jordan Samhi

x