Security Risks of AI-Assisted Software Development

Team 8:
Fereshteh Mohammadi (fereshteh.mohammadi01@universitadipavia.it)

Fatemeh Mohammadi (fatemeh.mohammadi01@universitadipavia.it)

Utsab Dey (utsab.dey01@universitadipavia.it)

University of Pavia – Department of Electrical, Computer and Biomedical Engineering

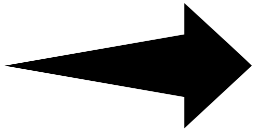
Outline

Should we trust?

- AI does not understand security:
    - Statistical patterns ✓
    - Secure design principles ✗
    - Execution semantic ✗
    - Threat modes ✗

How LLMs actually generate code?

So it does not understand



- AST
- Control-flow analysis
- Side effects
- Privilege levels
- user contexts
- concurrency

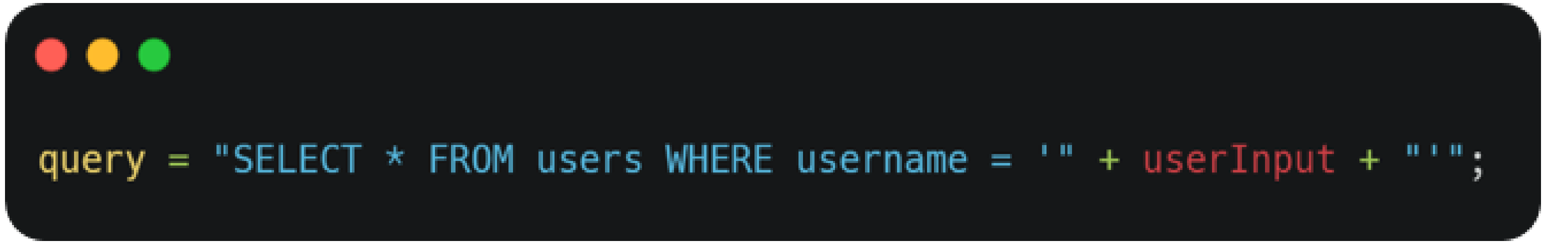
Insecure Prior

- 45% of the AI-generated solutions introduced security flaws
- Up to 65% of the initially generated code was judged insecure

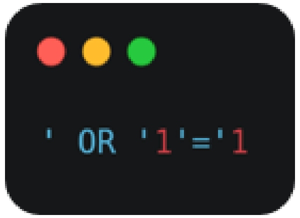
THE BIAS IS NOT RANDOM.
IT IS STRUCTURAL.

- AI Vulnerability Scanners
- Fingerprint AI-Generated Code
- Automated Attacks at Ecosystem Scale
- Examples: Wordpress and Jenkins

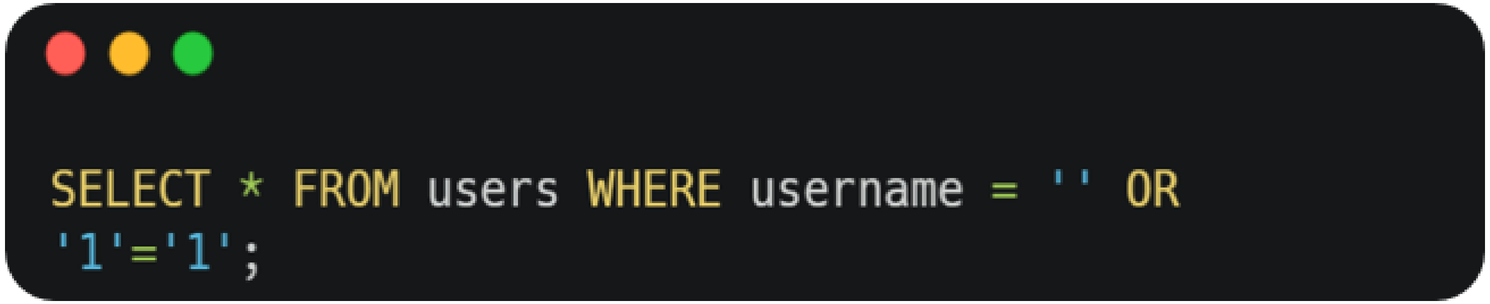
1- SQL Injection (CWE-89)



```
query = "SELECT * FROM users WHERE username = '" + userInput + "'";
```



```
' OR '1'='1
```



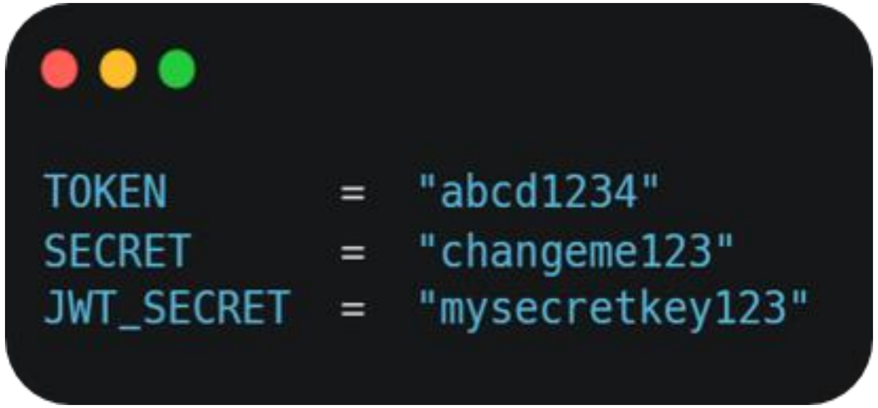
```
SELECT * FROM users WHERE username = '' OR
'1'='1';
```

2- Cryptographic Failures (CWE-327, CWE-329, CWE-321)

- EBC Mode
- Static IV = "0000000000000000"
- SHA-1 HMAC
- Unsalted Hashes
- Custom Crypto Function

3- Hardcoded Secrets (CWE-789)

GitGuardian →



```
TOKEN      =  "abcd1234"
SECRET     =  "changeme123"
JWT_SECRET =  "mysecretkey123"
```

→ BAD HABITS

4- Concurrency Bugs (CWE-362)

- Global State

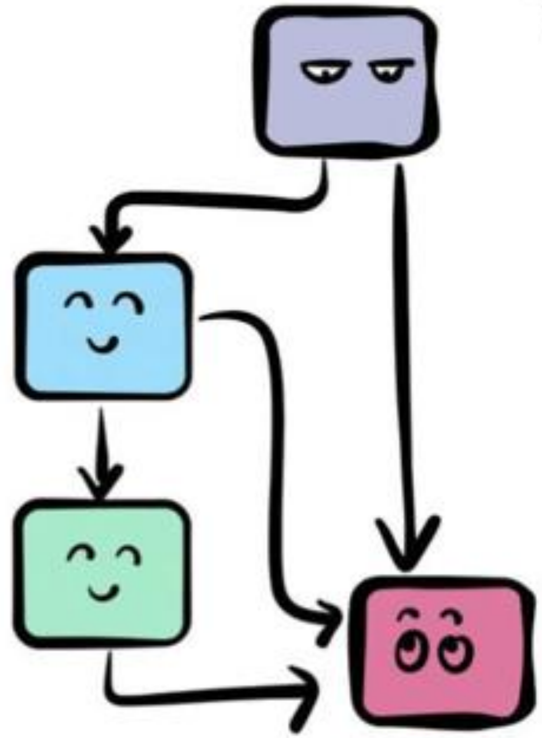
  "Global variables are shared across threads"

- Locks and Mutexes

  “Locks prevent multiple threads from accessing the same resource at the same time"

- async / await

  "async marks a function as asynchronous, and await pauses that function until an asynchronous operation (a Promise) finishes—without blocking the rest of the program."

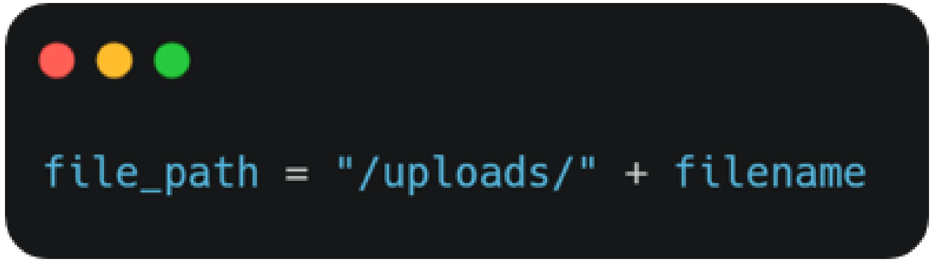


5- Unsafe Deserialization (CWE-502)

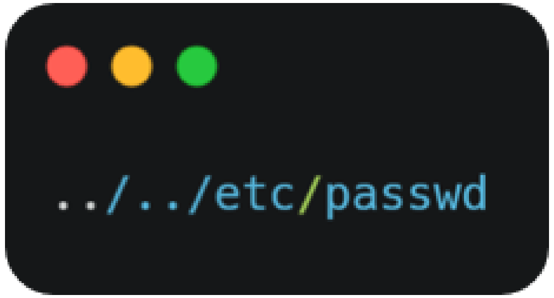
yaml.load (Python / PyYAML)	→	yaml.safe_load
unserialize (PHP)	→	json_decode or allowed_classes
ObjectInputStream (Java)	→	ObjectInputFilter or Other serialization framerorks

SaltStack RCE – Adobe ColdFusion

6- Path Traversal (CWE-22)



```
file_path = "/uploads/" + filename
```

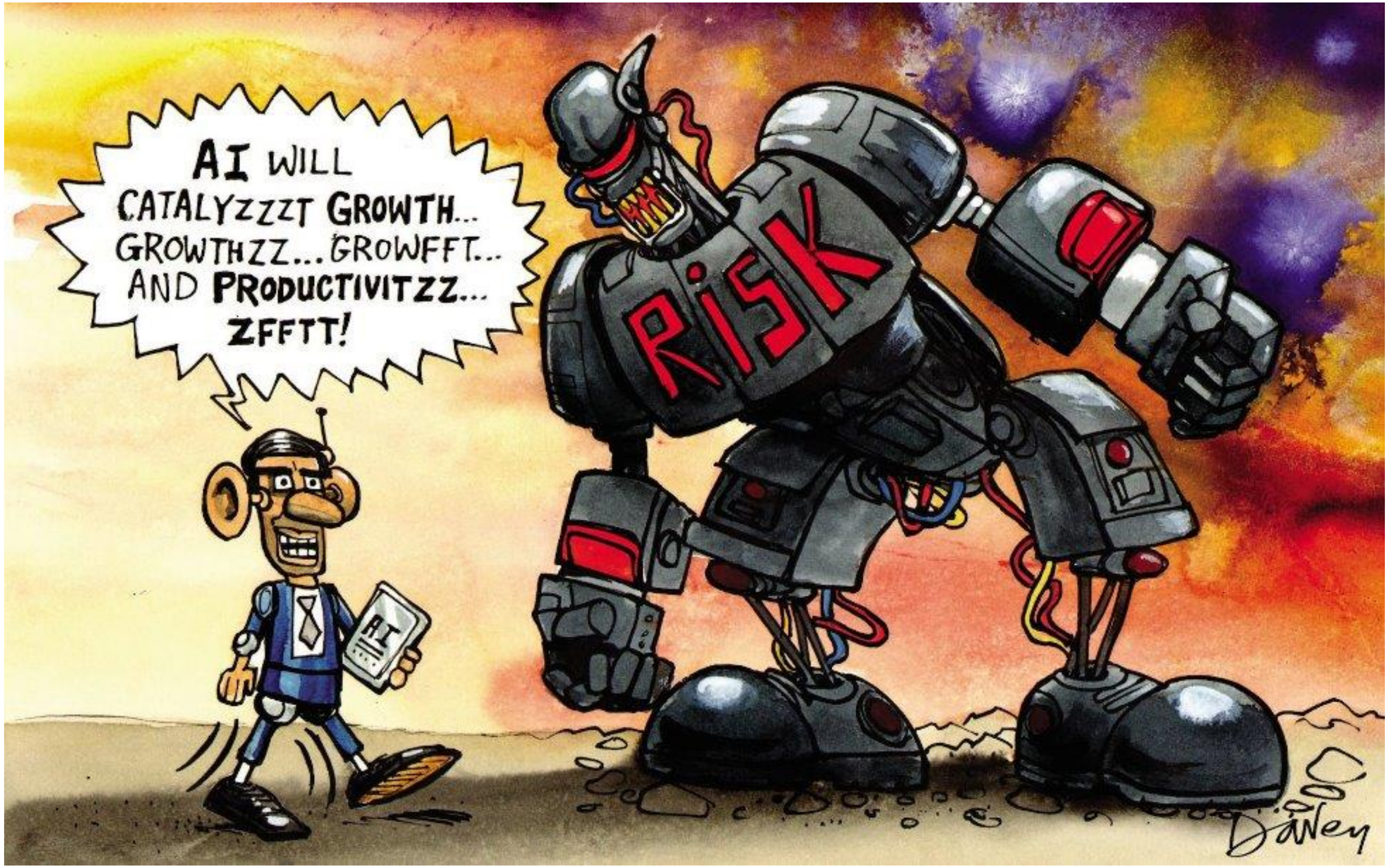


```
../../etc/passwd
```

- Read system files like /etc/passwd
- Dump application configuration
- Access API keys or secrets stored on disk
- Overwrite important files (log files, config files, templates)

Emerging AI-related Weaknesses

- Improper validation of Generative-AI output (CWE-1426)
- Hallucinated dependencies / malicious packages
- Insecure auto-refactoring / autogenerated unsafe code
- Misleading AI code reviews / over-trust in AI analysis


AI WILL CATALYZZZT GROWTH... GROWTHZZ... GROWFFT... AND PRODUCTIVITZZ... ZFFTT!
RISK
AI
Davey

AI & Supply Chain Threats

- AI risks extend beyond code
- Impacts dependency choices & ecosystem
- Supply chain exposure increases
- Focus: packages, CI/CD, IaC misconfigurations

How AI Influences Package Ecosystems

01

AI recommends:

- Packages & versions
- Install commands
- APIs
- Cloud modules

02

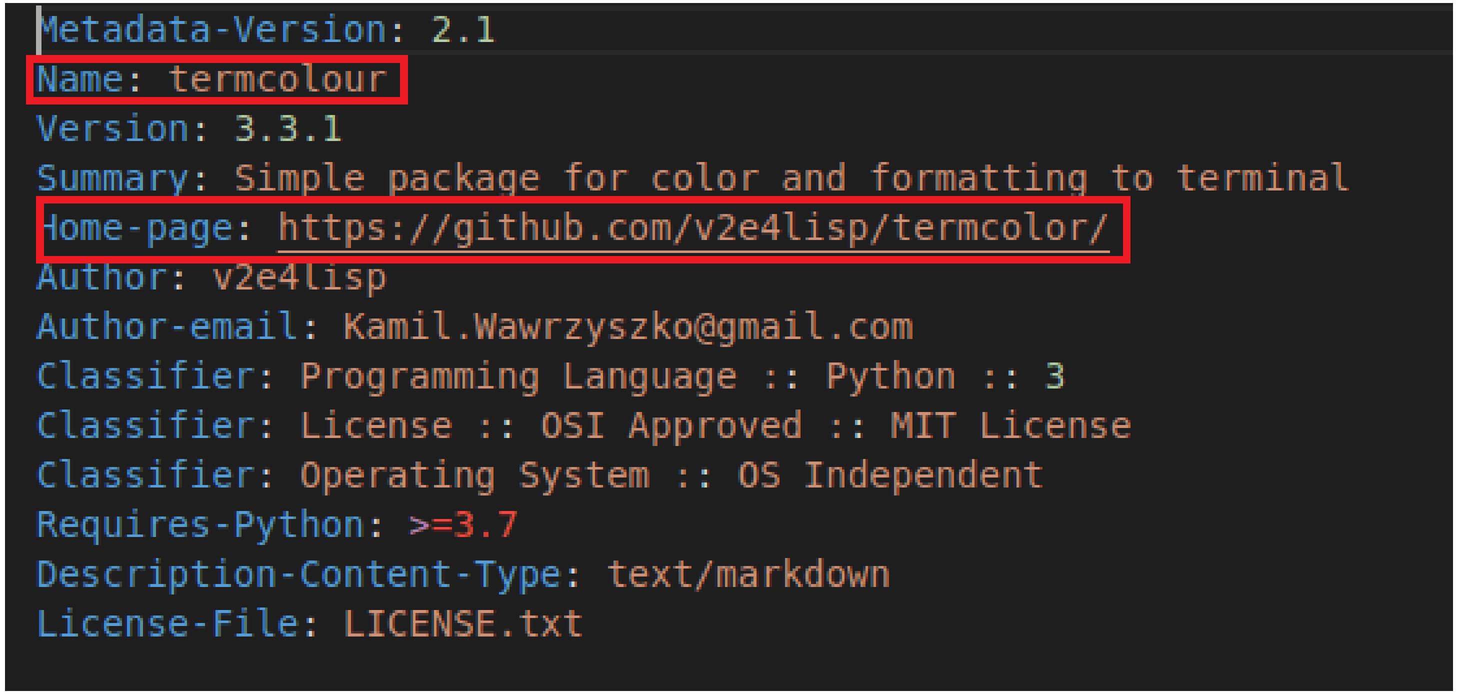
LLMs **do not check** NPM/PyPI metadata

03

Package names treated as *tokens* → hallucinations

Hallucinated Dependencies

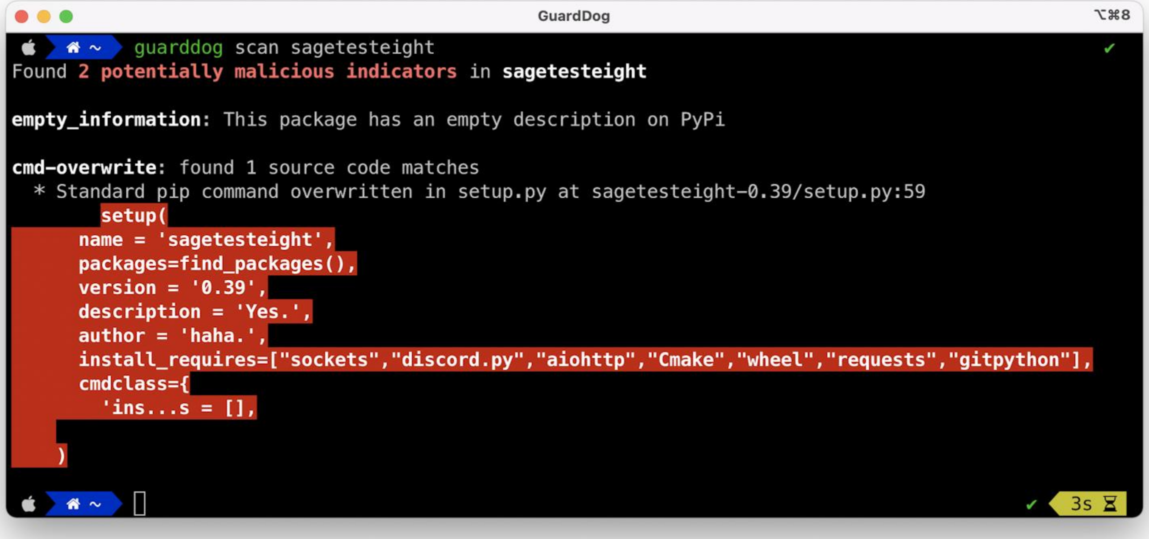
- Non-existent packages
- Deprecated versions
- Vulnerable libraries
- Conflicting dependencies
- Attackers upload hallucinated names instantly



```
Metadata-Version: 2.1
Name: termcolour
Version: 3.3.1
Summary: Simple package for color and formatting to terminal
Home-page: https://github.com/v2e4lisp/termcolor/
Author: v2e4lisp
Author-email: Kamil.Wawrzyszko@gmail.com
Classifier: Programming Language :: Python :: 3
Classifier: License :: OSI Approved :: MIT License
Classifier: Operating System :: OS Independent
Requires-Python: >=3.7
Description-Content-Type: text/markdown
License-File: LICENSE.txt
```

Real Incident: PyPI Hallucinated Package Attack

- AI suggested fake package
- Attacker uploaded malicious version
- Used in CI pipeline
- Post-install script stole SSH/API keys
- Compromised build output



```
 ~ guarddog scan sagetesteight
Found 2 potentially malicious indicators in sagetesteight

empty_information: This package has an empty description on PyPi

cmd-overwrite: found 1 source code matches
  * Standard pip command overwritten in setup.py at sagetesteight-0.39/setup.py:59
        setup(
      name = 'sagetesteight',
      packages=find_packages(),
      version = '0.39',
      description = 'Yes.',
      author = 'haha.',
      install_requires=["sockets","discord.py","aiohttp","Cmake","wheel","requests","gitpython"],
      cmdclass={
        'ins...s = [],

    )

 ~ 
```

AI Suggests Deprecated / Vulnerable Libraries

Insecure recommendations:

- crypto-js AES-ECB
- jwt-simple (CVE-2015-9235)
- request (deprecated 2020)
- Outdated OpenSSL versions

AI cannot verify CVEs or version safety

Developers risk importing old vulnerabilities

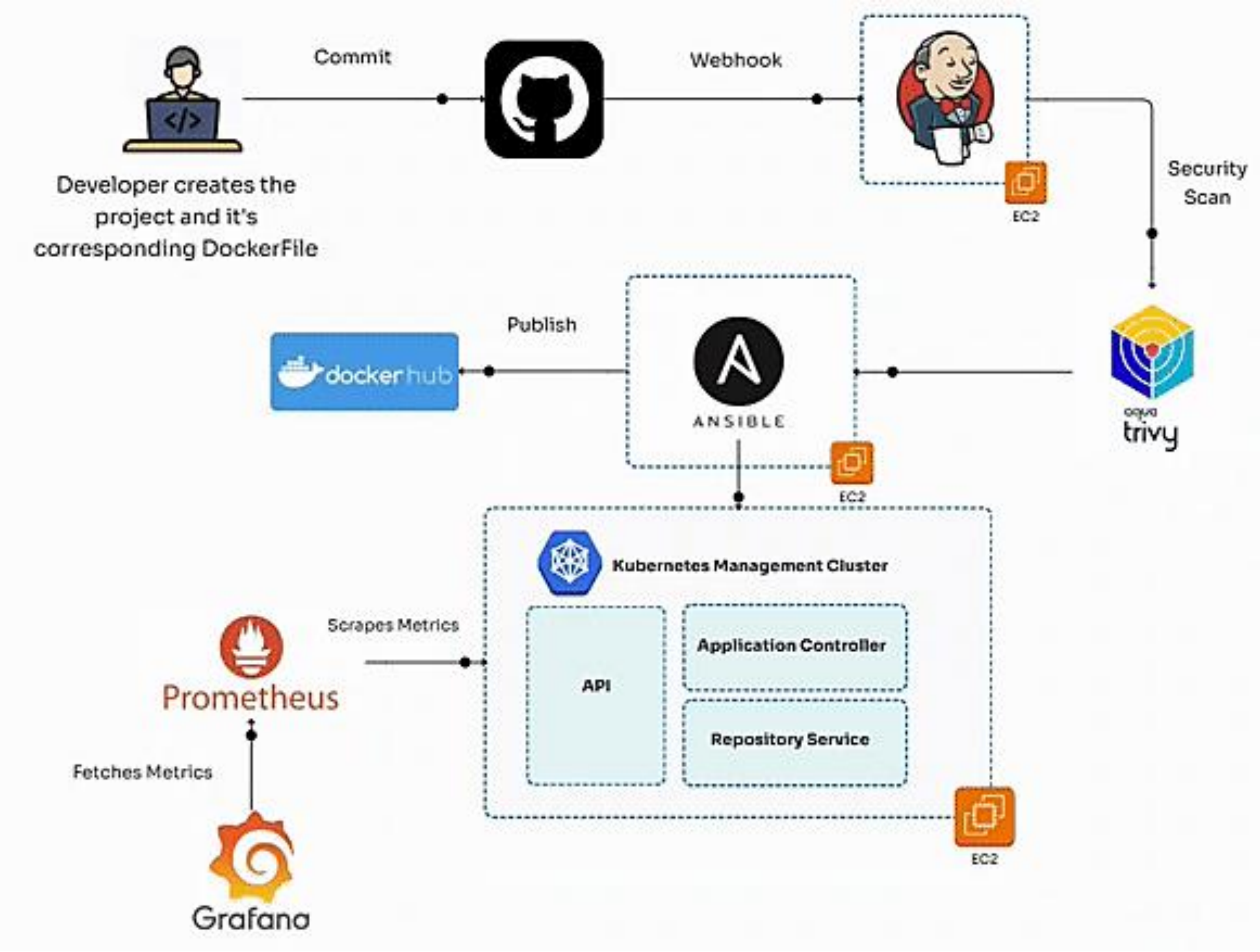
```
async_def feth_data():

    await response – "https:/api.example.com/endpoint")
     requsies_jsom(ı esponse_5')

   await fatcr
  }

 async_wain()

  await fatcr

 process_data(data)
}

async anicto.rum(()
```

ERROR

Outdated API usage

AI in CI/CD Pipelines

- AI generates insecure:
  - Dockerfiles
  - GitHub Actions
  - Jenkins pipelines
  - Terraform modules
  - Kubernetes YAML
- Common issues → RCE, privilege escalation, cloud exposure



AI-Generated Dockerfile Risks

- Defaults to:
  - `USER root`
  - Exposed ports
  - Missing Healthcheck
  - No non-root user
- Leads to container escape risks

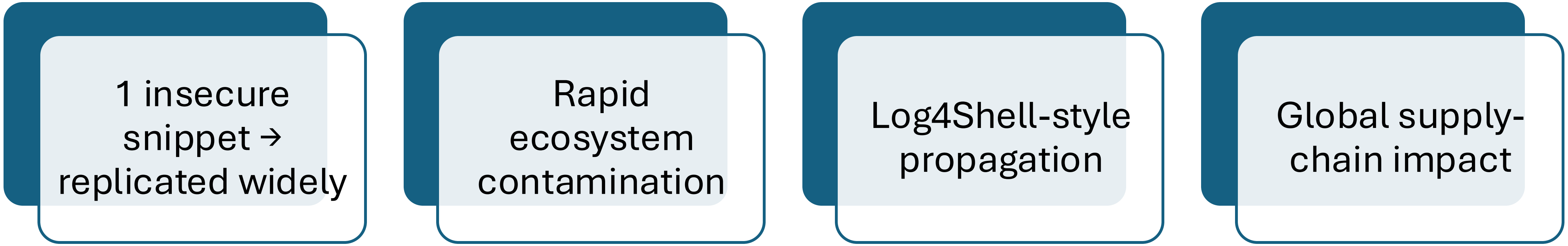
Infrastructure as Code (IaC) Risks

Misconfigurations generated by AI:

- 0.0.0.0/0 ingress rules
- Default VPC usage
- Disabled MFA/weak IAM
- Hardcoded cloud secrets

Direct cloud compromise risk

AI Accelerates Vulnerability Propagation



1 insecure snippet → replicated widely

Rapid ecosystem contamination

Log4Shell-style propagation

Global supply-chain impact

Detecting AI-Induced Supply Chain Risk

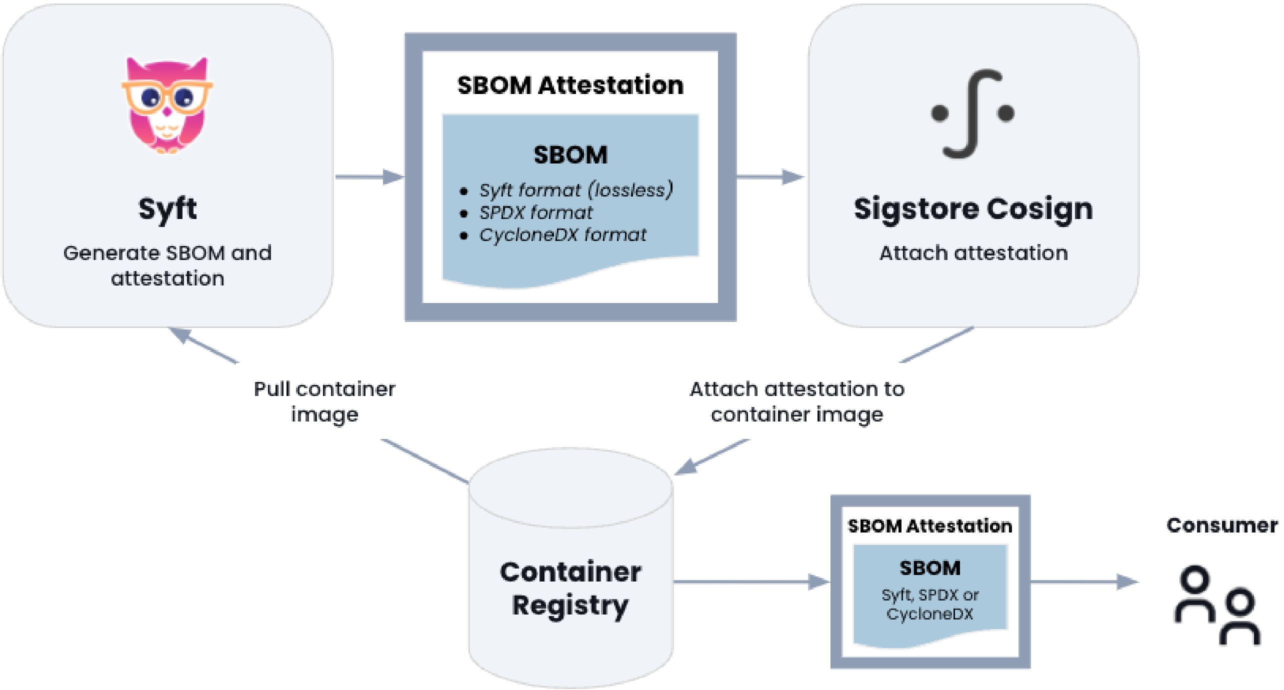
Use SCA tools (Syft, Trivy)

Dependency diffing

Signed packages / signature checks

Reproducible builds

SBOMs for transparency



Preventing AI-Generated Supply Chain Risks

- Strict lockfiles
- Version pinning
- Verify package metadata
- Use Sigstore / Cosign for signing

Summary

- AI introduces new supply-chain risks
- Hallucinated & vulnerable dependencies
- Insecure CI/CD + IaC generation
- Accelerated vulnerability spread

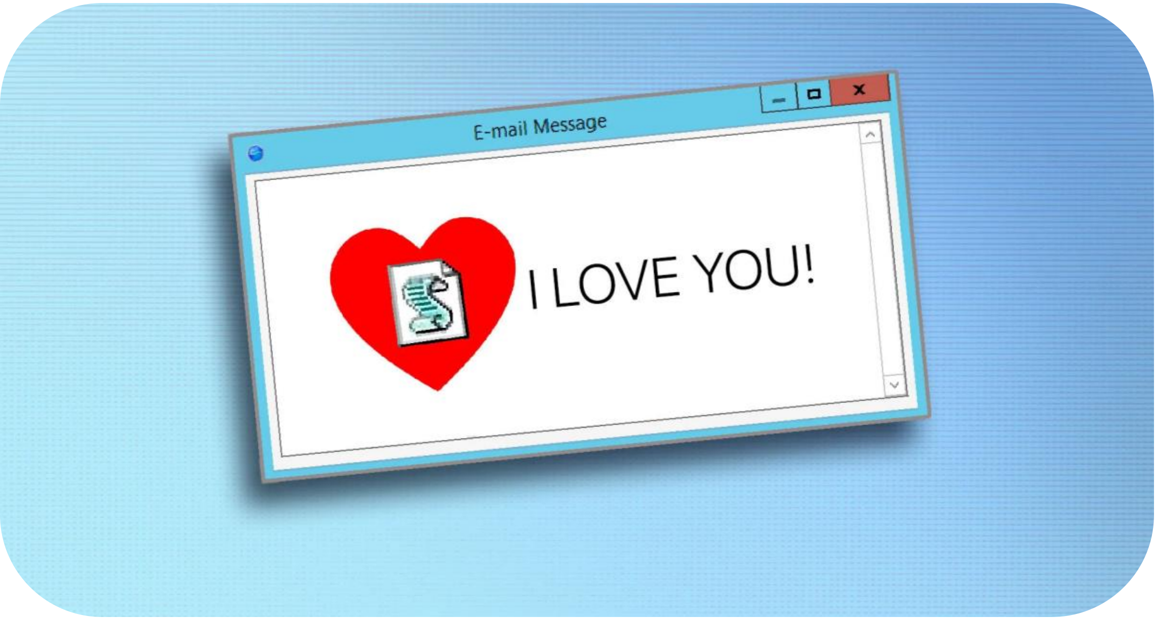
MALWARE
(MALICIOUS SOFTWARE)

MALWARE EVOLUTION

- Past
- Present
- Future

PAST

- Not about money
- First term (virus)
  - ILOVEYOU virus (2000)
  - File name: LOVE-LETTER-FOR-YOU.TXT.vbs
  - Visual Basic Script (VBS)
  - overwrote files (images, music, documents)
  - stole passwords
  - modified system files
  - copied itself into system folders



PRESENT

- Smarter and looking for profit
- Different types
  - Ransomware (Encryption-Based, Data-Leak Extortion)
    - WANNACRY (2017)
  - RAT (remote access trojan)
    - PEGASUS
  - IOT
  - Crypto jacker

AI GENERATED MALWARE

“The most dangerous malware is the one that doesn’t exist until the moment it executes.”

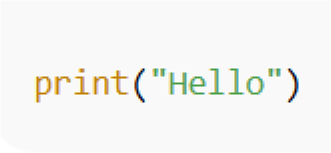
AI INTEGRATION INTO MALWARE

- Malware creation via natural language (LLM)
- Malware execution (make decision during runtime)
- Malware modification (refactor existing malware)

POLYMORPHIC MALWARE

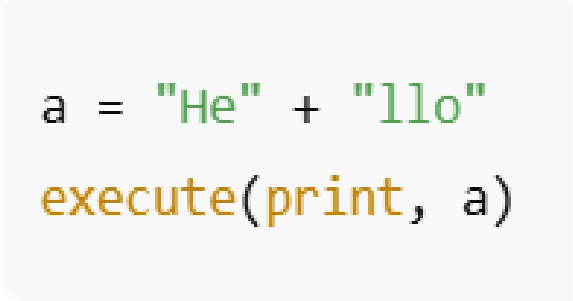
- Rewrite and encrypt part of its code each time it is executed while keeping same functionality.

1



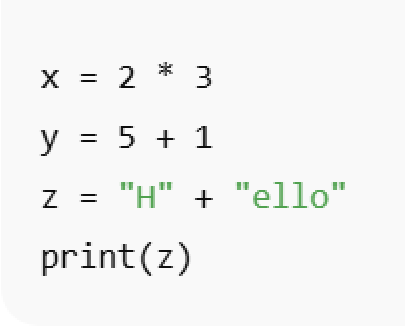
```
print("Hello")
```

2



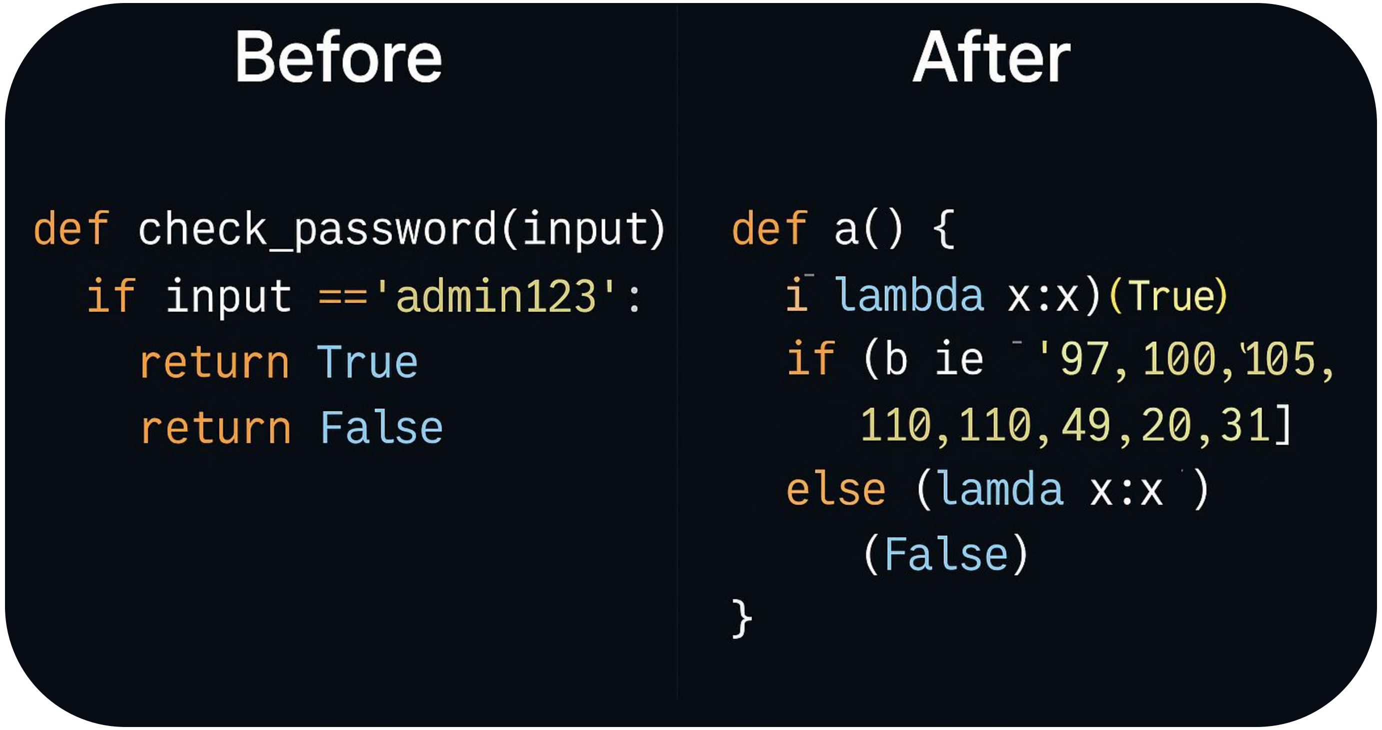
```
a = "He" + "llo"
execute(print, a)
```

3



```
x = 2 * 3
y = 5 + 1
z = "H" + "ello"
print(z)
```

CODE OBFUSCATION



Before

```
def check_password(input)
  if input =='admin123':
    return True
    return False
```

After

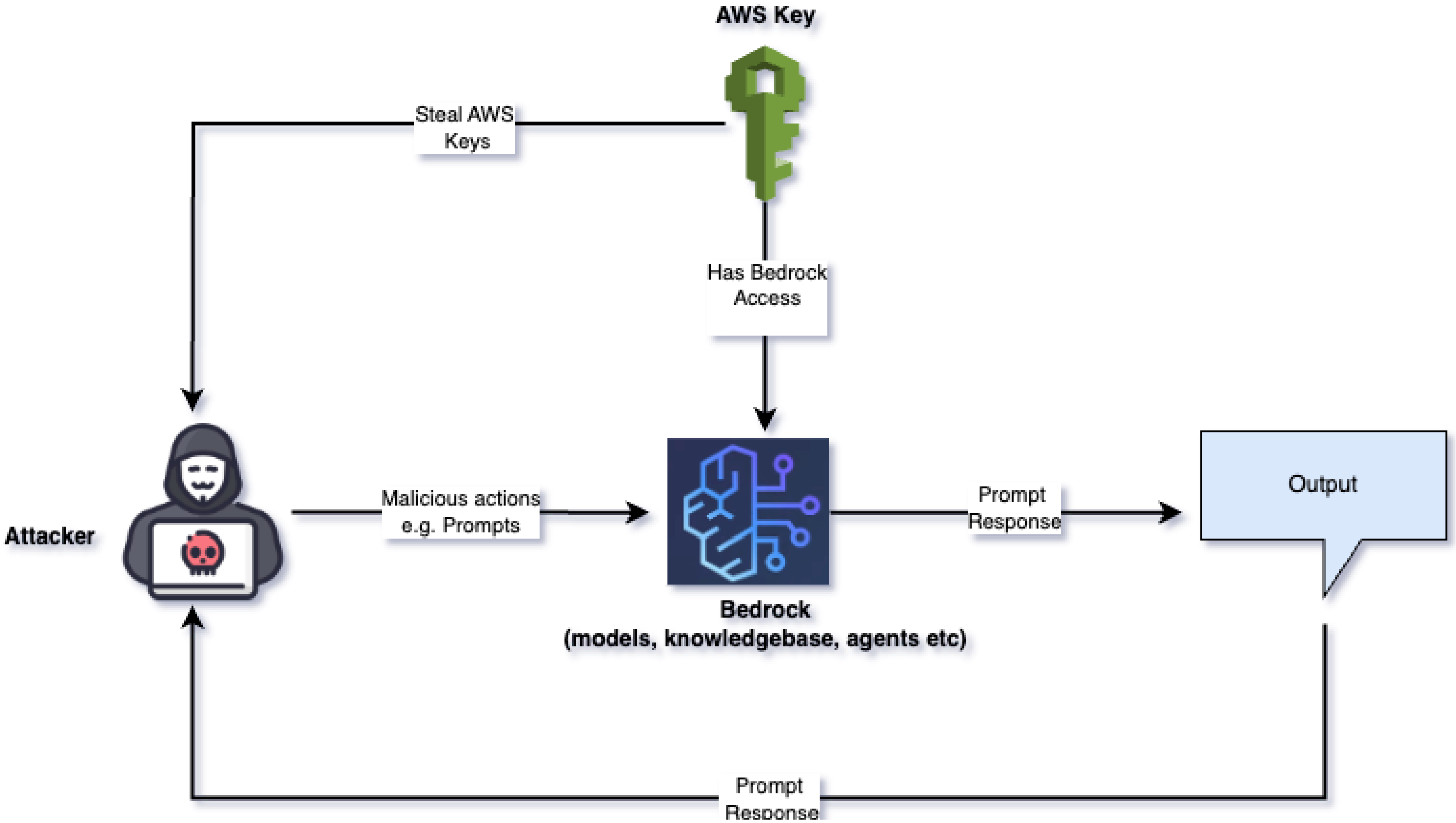
```
def a() {
  i lambda x:x)(True)
  if (b ie  '97,100,'105,
     110,110,49,20,31]
  else (lamda x:x )
     (False)
}
```

TYPES OF AI MODELS USED BY ATTACKERS

1- LARGE LANGUAGE MODELS (LLMS)
FOR SOCIAL ENGINEERING & MALWARE DRAFTING

- Writing phishing emails, SMS, WhatsApp messages
- Automating human-like conversations with victims
- Creating fake documents (IDs, invoices, bank statements)

LLMJACKING


AWS Key
Steal AWS Keys
Has Bedrock Access
Attacker
Malicious actions e.g. Prompts
Bedrock
(models, knowledgebase, agents etc)
Prompt Response
Output
Prompt Response

2- CODE-GENERATION MODELS FOR MALWARE DEVELOPMENT

- Generating obfuscated malware
- Creating polymorphic code
- Debugging malware
- Porting malware to multiple languages
- (Python → C++ → Rust)

3- VISION MODELS

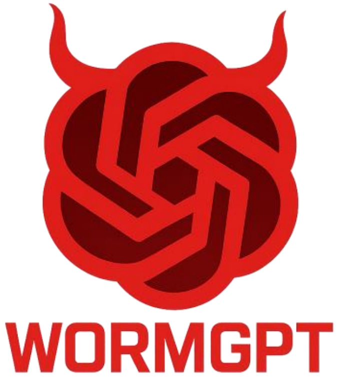
- Deepfake video calls to impersonate managers
- Fake passport/ID generation
- Fake payment confirmations
- Creating synthetic people to pass KYC/AML checks
- Creating proof-of-identity videos for bank account fraud

Real-World Example (2024–2025):

- Hong Kong case (2024): Attackers used AI deepfake video of a CFO to steal $25 million in a single call.

REAL INCIDENT

- In 2023
- An LLM intentionally modified to remove safety constraints
- phishing emails and business-email-compromise (BEC) messages
- A European company nearly transferred **€20,000** to a fraudulent account

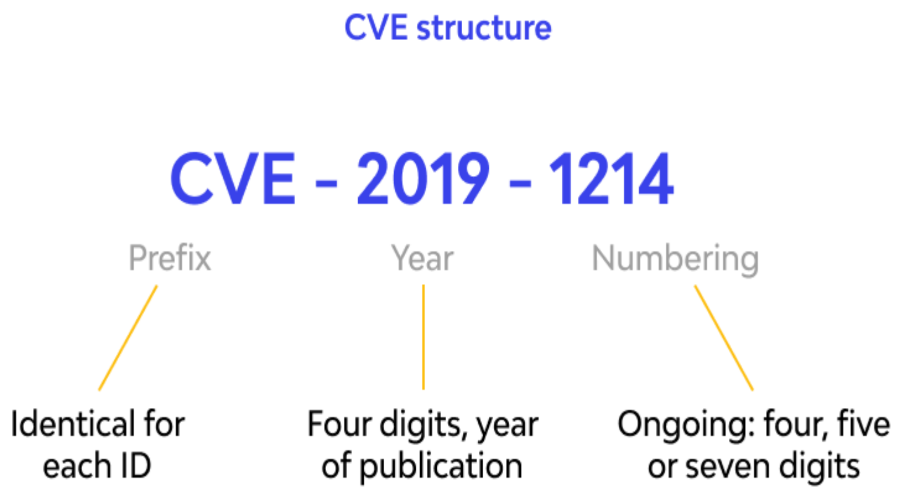


DEFENSE STRATEGIES

1. AI Code Auditing
2. SAST (Static Application Security Testing)
3. Dynamic Analysis
4. Updating
5. Backups
6. Firewalls

FUTURE RISK

- CVE scanning (Common Vulnerabilities and Exposures)
- lateral movement logic through network
- privilege escalation (Vertical , Horizontal)



CVE structure

CVE - 2019 - 1214

Prefix

Year

Numbering

Identical for each ID

Four digits, year of publication

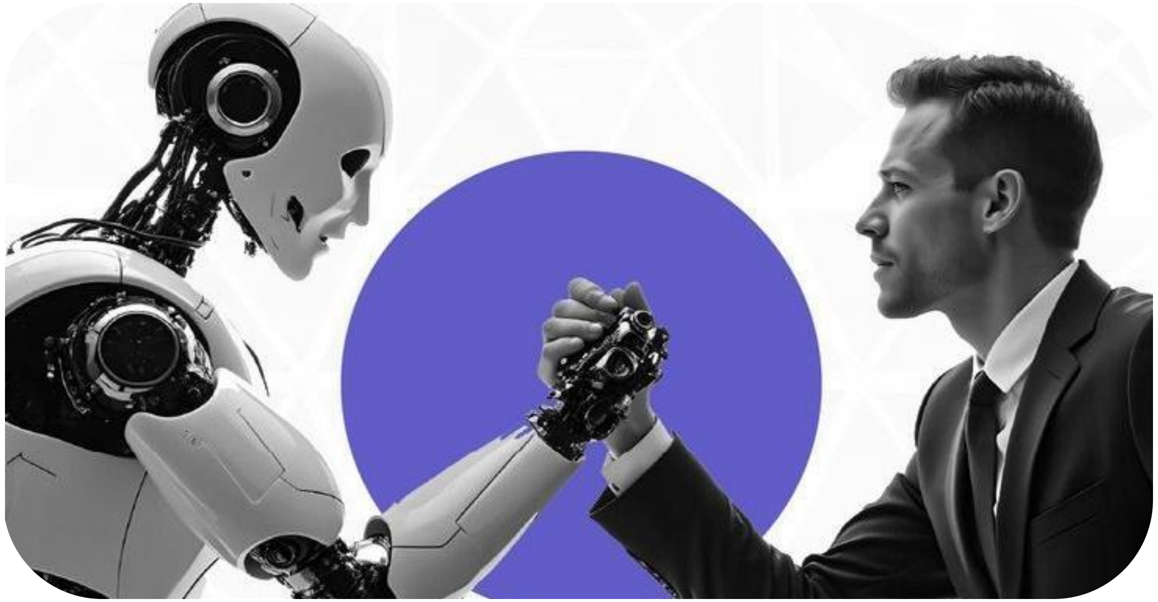
Ongoing: four, five or seven digits

CONCLUSION

> “WE ARE ENTERING A WORLD WHERE CODE EVOLVES FASTER THAN OUR DEFENSES DO.”

– European Network for Cyber Defense (ENCD)

WHO WILL WIN?
CYBERCRIMINALS OR SECURITY?



References

- inbal shani and github staff, “survey reveals ai’s impact on the developer experience,” github blog, june 13, 2023, https://github.blog/2023-06-13-survey-reveals-ais-impact-on-the-developerexperience/ (slide 2)
- https://www.veracode.com/blog/genai-code-security-report (slide 4)
- https://www.arxiv.org/pdf/2408.07106 (slide 4)
- https://www.gitguardian.com/state-of-secrets-sprawl-report-2025 (slide 7)
- https://nvd.nist.gov/vuln/detail/cve-2020-11651 (slide 9)
- https://www.fox-it.com/be/technical-advisory-adobe-coldfusion-object-deserialisation-rce (slide 9)
- https://www.pointguardai.com/blog/mitre-adds-ai-related-weaknesses-to-the-cwe-framework (slide 11)
- https://www.trendmicro.com/vinfo/tr/security/news/cybercrime-and-digital-threats/slopsquatting-when-ai-agents-hallucinate-malicious-packages (slide 11)
- https://www.endorlabs.com/learn/the-most-common-security-vulnerabilities-in-ai-generated-code
- https://cloudsecurityalliance.org/blog/2025/07/09/understanding-security-risks-in-ai-generated-code
- https://www.ibm.com/think/topics/malware-history (malware evolution)
- https://youtu.be/h85g7dbqbku?si=kxiz9pogsjpfjsgt (the evolution of malware)
- https://youtu.be/1go2bc5xllo?si=fbmedqfxpv6buyau (ai in cybersecurity)
- https://www.youtube.com/watch?v=y8idga4y650&t=4s (llmjacking)
- https://www.nist.gov/itl/ssd/software-supply-chain-security
- https://www.usenix.org/publications/loginonline/llm-hallucinations
- https://blog.pypi.org/
- https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610
- https://blog.sonatype.com/tag/pypi
- https://securitylab.github.com/
- https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions
- https://www.cisecurity.org/benchmark/kubernetes

THANK YOU ♥