Smarter Contracts:

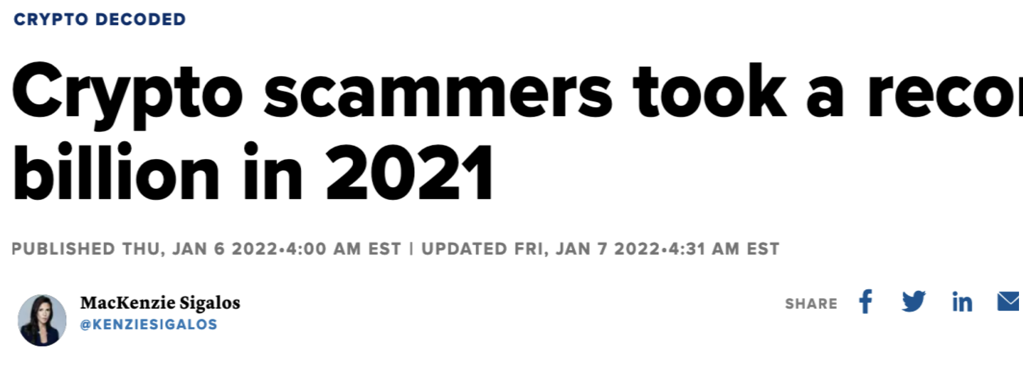
Detecting Vulnerabilities in Smart Contracts with Deep Transfer Learning

Christoph Sendner[1], Huili Chen[2], Hossein Fereidooni[3], Lukas Petzi[1], Jan König[1], Jasper Stang[1], Alexandra Dmitrienko[1], Ahmad-Reza Sadeghi[3], Farinaz Koushanfar[2]

[1]University of Wuerzburg, [2]UC San-Diego, [3]TU Darmstadt

NDSS SYMPOSIUM 30TH ANNIVERSARY

Security Problems of Smart Contracts

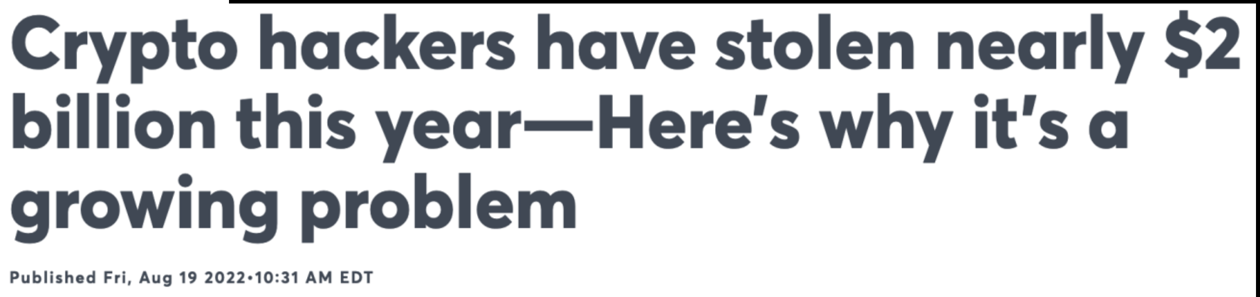


CRYPTO DECODED

Crypto scammers took a record $14 billion in 2021

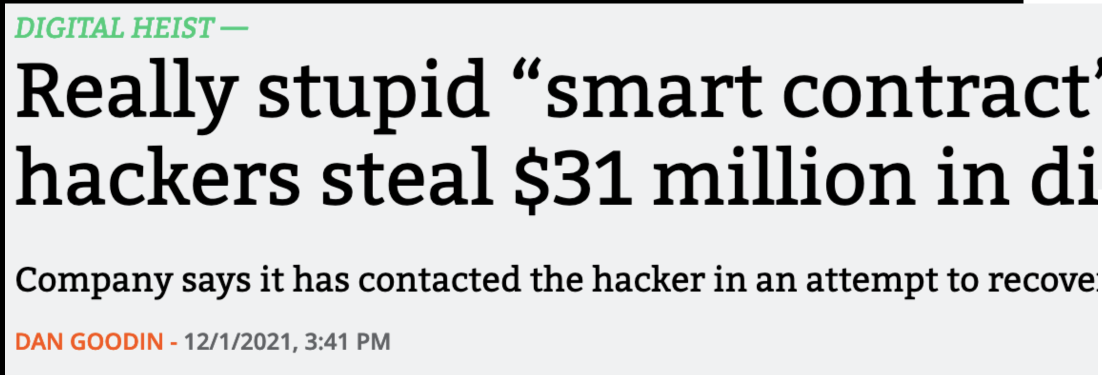
PUBLISHED THU, JAN 6 2022•4:00 AM EST | UPDATED FRI, JAN 7 2022•4:31 AM EST

MacKenzie Sigalos
@KENZIESIGALOS



Crypto hackers have stolen nearly $2 billion this year—Here's why it's a growing problem

Published Fri, Aug 19 2022•10:31 AM EDT

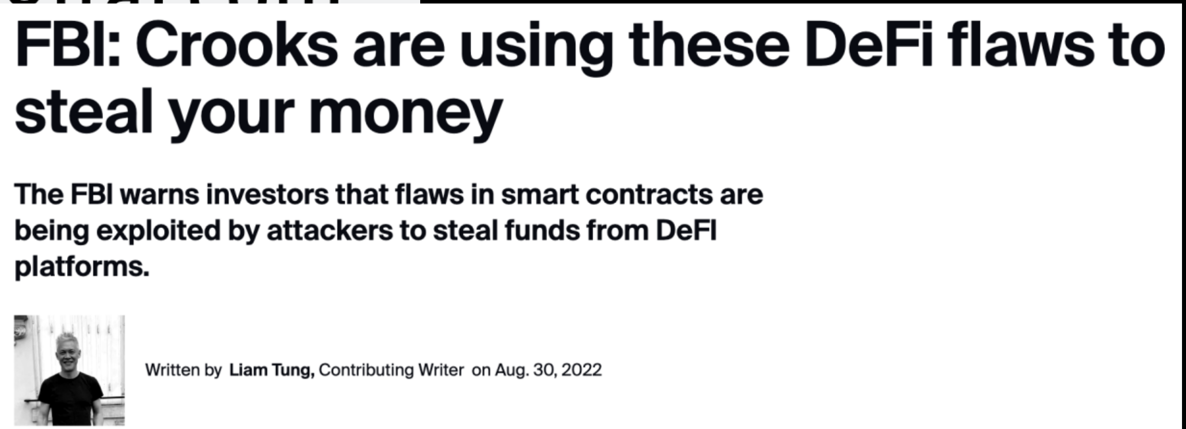


DIGITAL HEIST —

Really stupid "smart contract" bug let hackers steal $31 million in digital coin

Company says it has contacted the hacker in an attempt to recove

DAN GOODIN - 12/1/2021, 3:41 PM



FBI: Crooks are using these DeFi flaws to steal your money

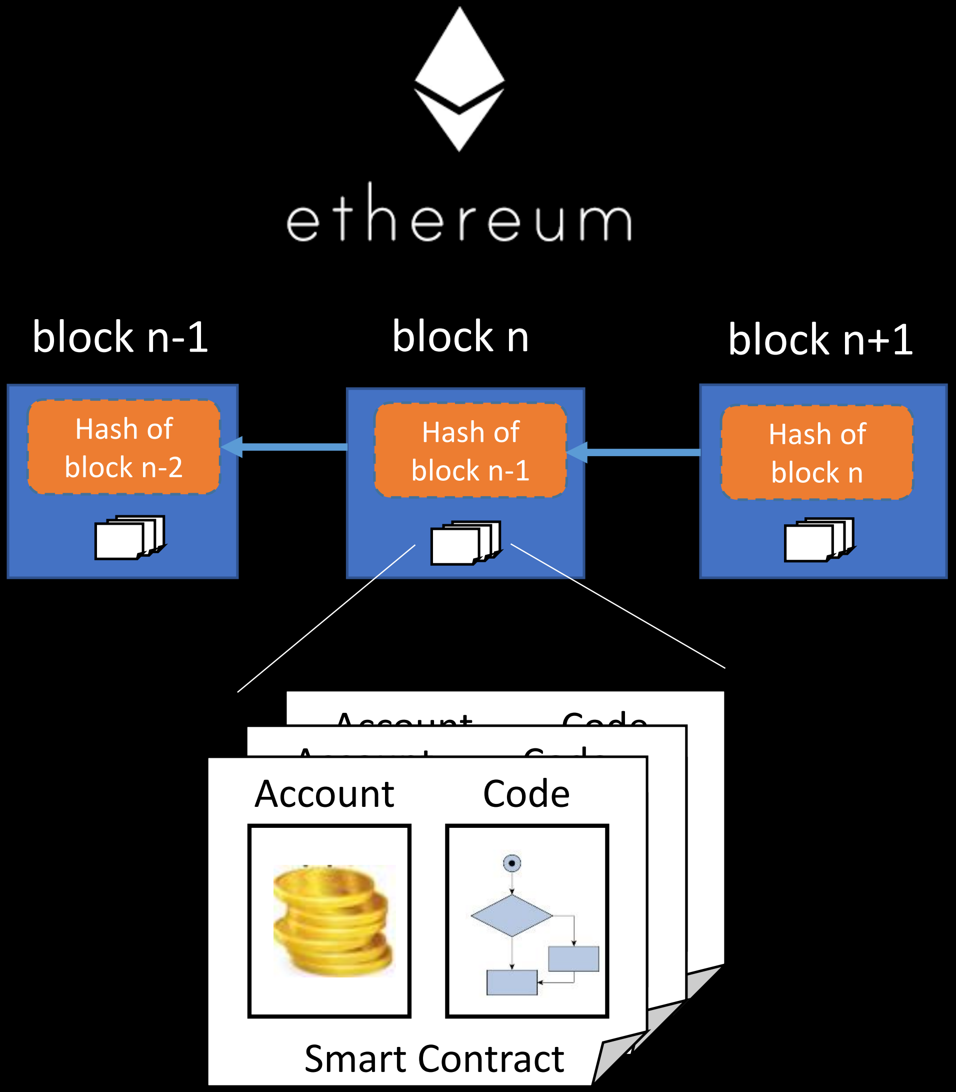
The FBI warns investors that flaws in smart contracts are being exploited by attackers to steal funds from DeFI platforms.

Written by Liam Tung, Contributing Writer on Aug. 30, 2022

Smart Contracts Basics

- Smart Contracts
  - Software programs hosted by blockchains
  - Manage financial assets
  - Automatically manage their own accounts
  - In charge of significant financial assets
  - Public entities

- Our focus is on Ethereum



Vulnerabilities (selected)

Software

Accessible Selfdestruct

- Attackers may destroy contracts by calling self-destruct function

Arbitrary Jump

- An attacker may jump to an arbitrary function

Assert Violation

- Attackers abuse constant error state of a smart contract

Integer Under/Overflow

- Integers flip when incremented over the feasible range

Runtime

Call-stack Depth

Unchecked Calls

Reentrancy

- Attackers abuse external calls

Greedy Contracts

- Contracts can receive, but not send, cryptocurrency

Blockchain

Money Concurrency

- Transaction reordering may lead to race conditions

DoS with Failed Call

- Unchecked Calls in a loop may lead to DoS

Time Dependency

- Dependence on block values may lead to unpredictable state

Weak Randomness

- Generation of randomness in smart contracts is difficult

Many Vulnerabilities

Difficult Patching

Extensive Security Testing

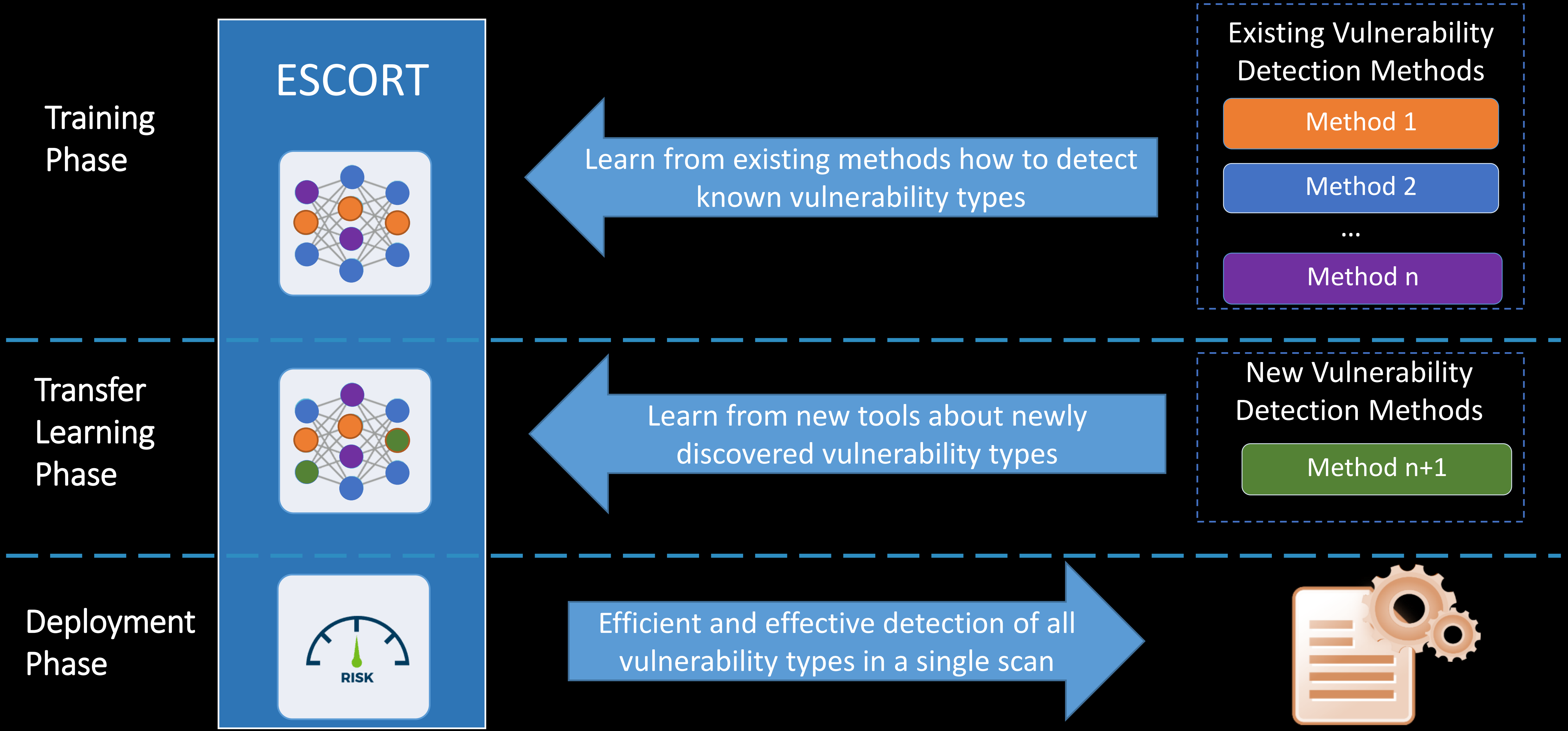
More info available at Smart Contract Weakness Classification (SWC) Registry: https://swcregistry.io/

Security Testing of Smart Contracts

Can we combine all tools into one?

- 👎 Limited in scope
- 👎 Limited detection capability
- 👎 r efficiency
- 👎 Access to source code
- 👎 Interoperability issues

Idea: One ML-based Tool that Learns from Many



Training Phase

ESCORT

Learn from existing methods how to detect known vulnerability types

Existing Vulnerability Detection Methods

Method 1

Method 2

…

Method n

Transfer Learning Phase

Learn from new tools about newly discovered vulnerability types

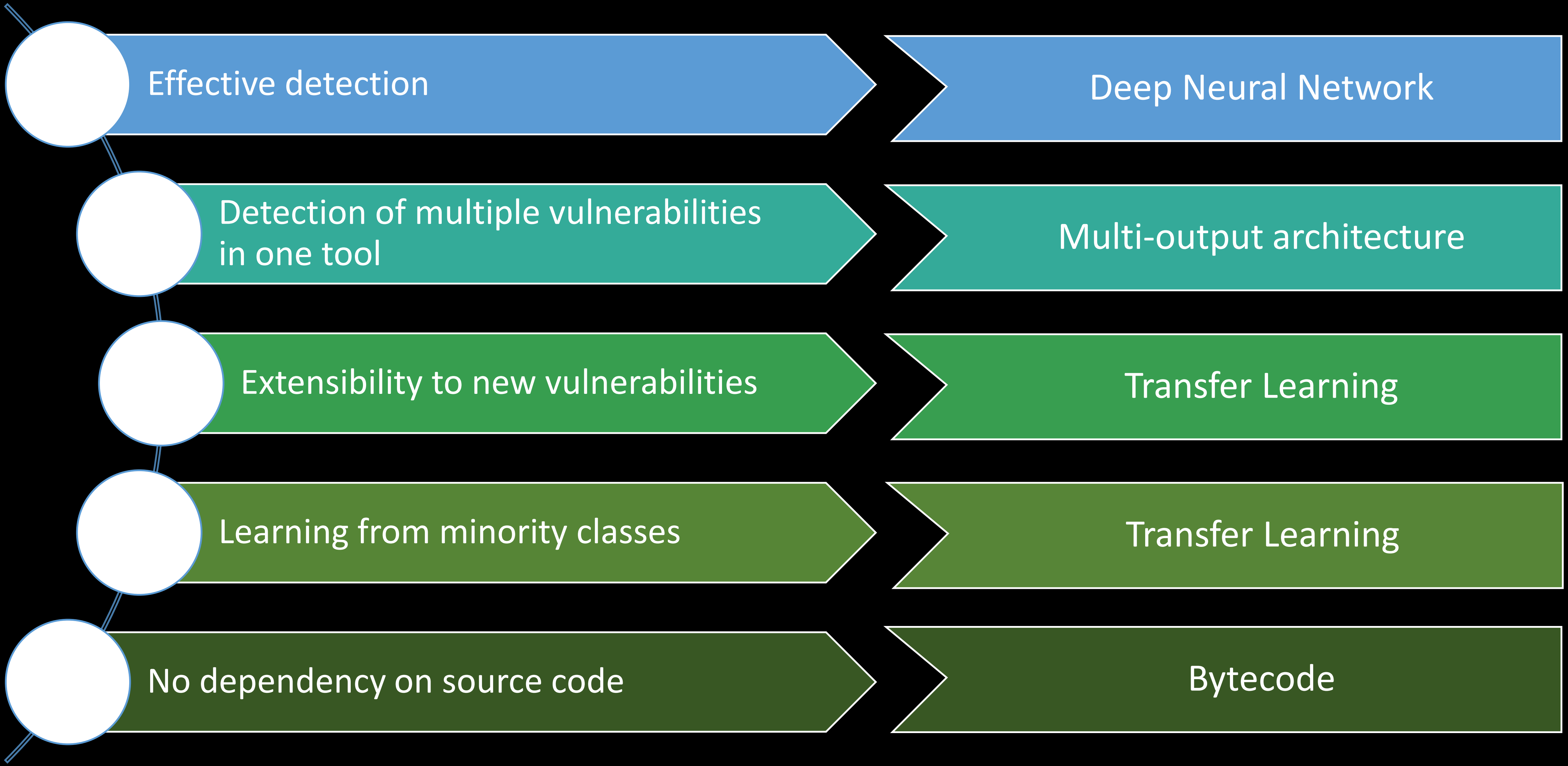
New Vulnerability Detection Methods

Method n+1

Deployment Phase

Efficient and effective detection of all vulnerability types in a single scan

Tackled Challenges



Effective detection

Deep Neural Network

Detection of multiple vulnerabilities in one tool

Multi-output architecture

Extensibility to new vulnerabilities

Transfer Learning

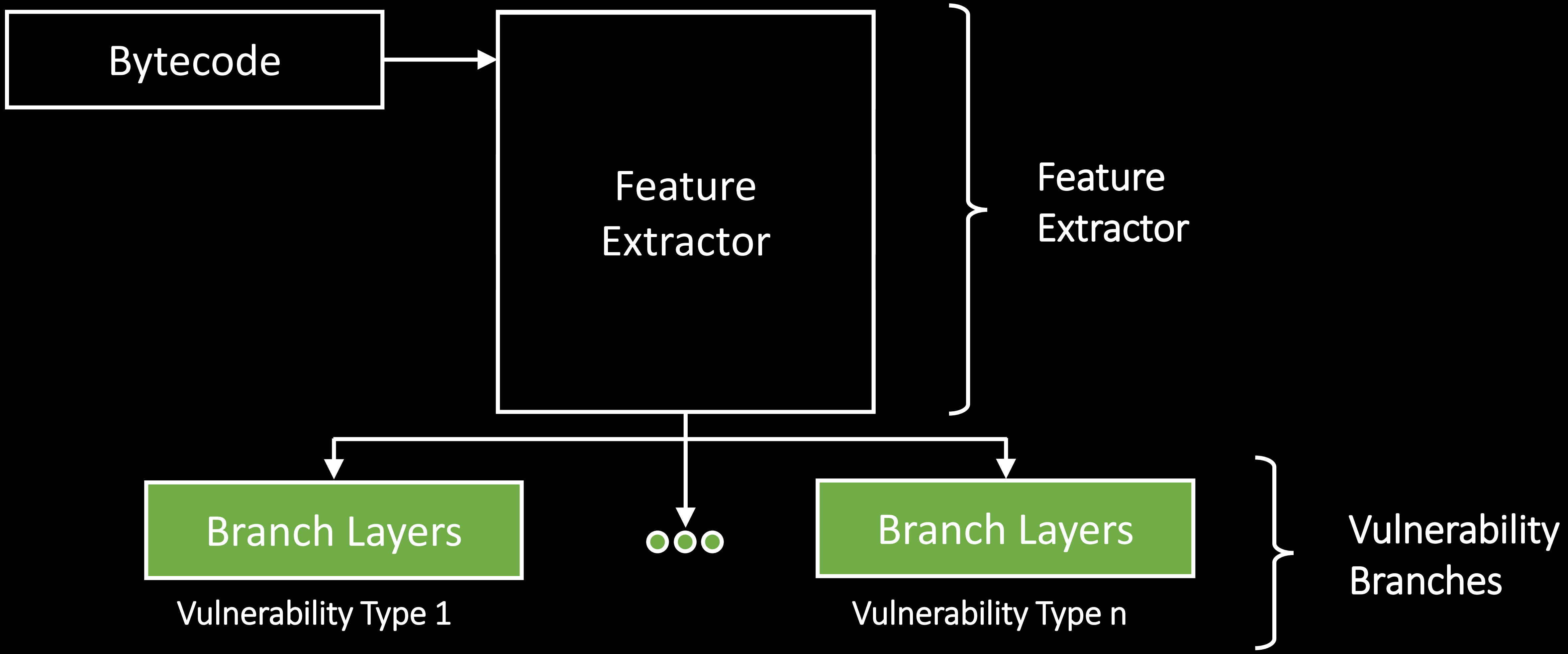
Learning from minority classes

Transfer Learning

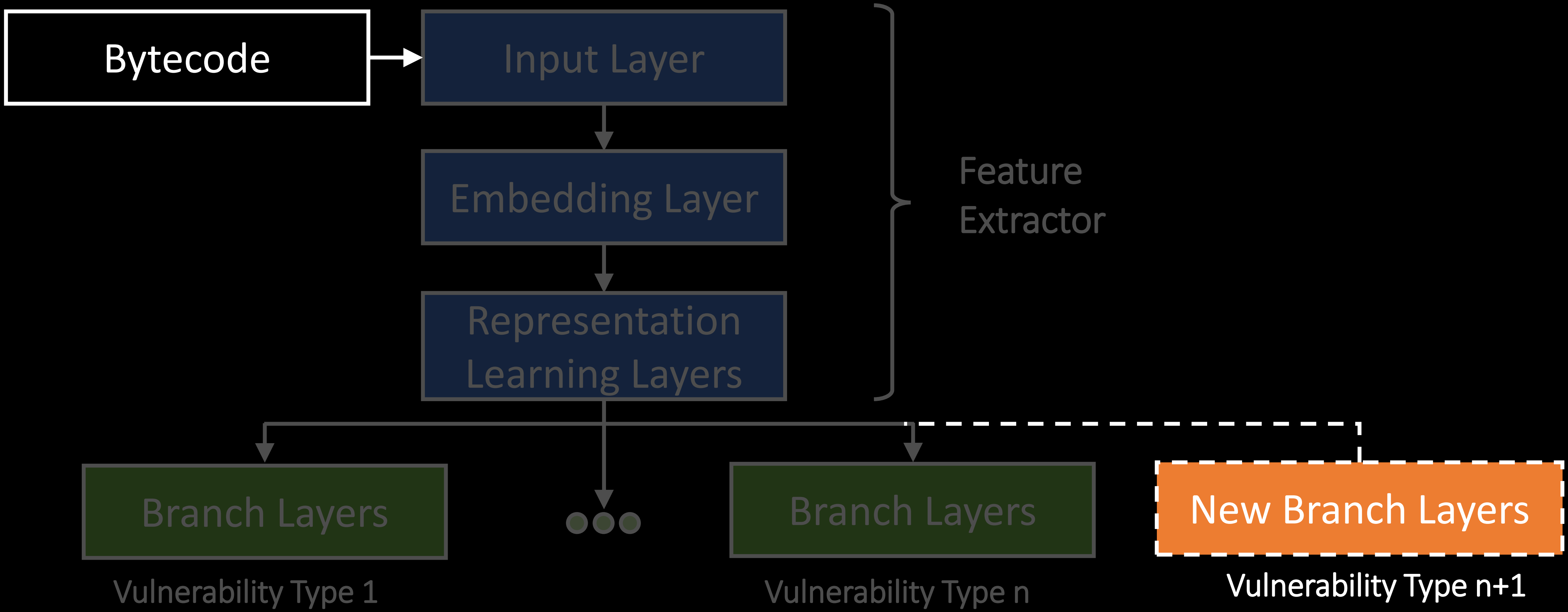
No dependency on source code

Bytecode

Approach: Multi-output Architecture

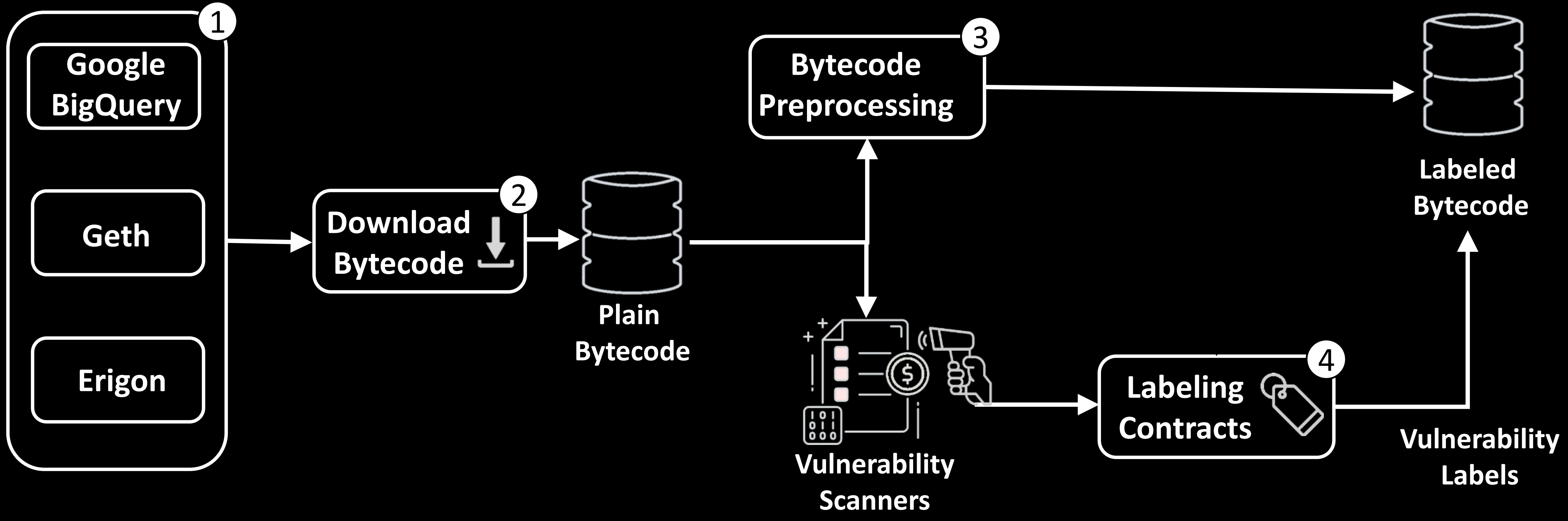


Approach: Transfer Learning



Dataset and Data Labeling

- ~3.6 million Smart Contracts
- 4 vulnerability scanning tools

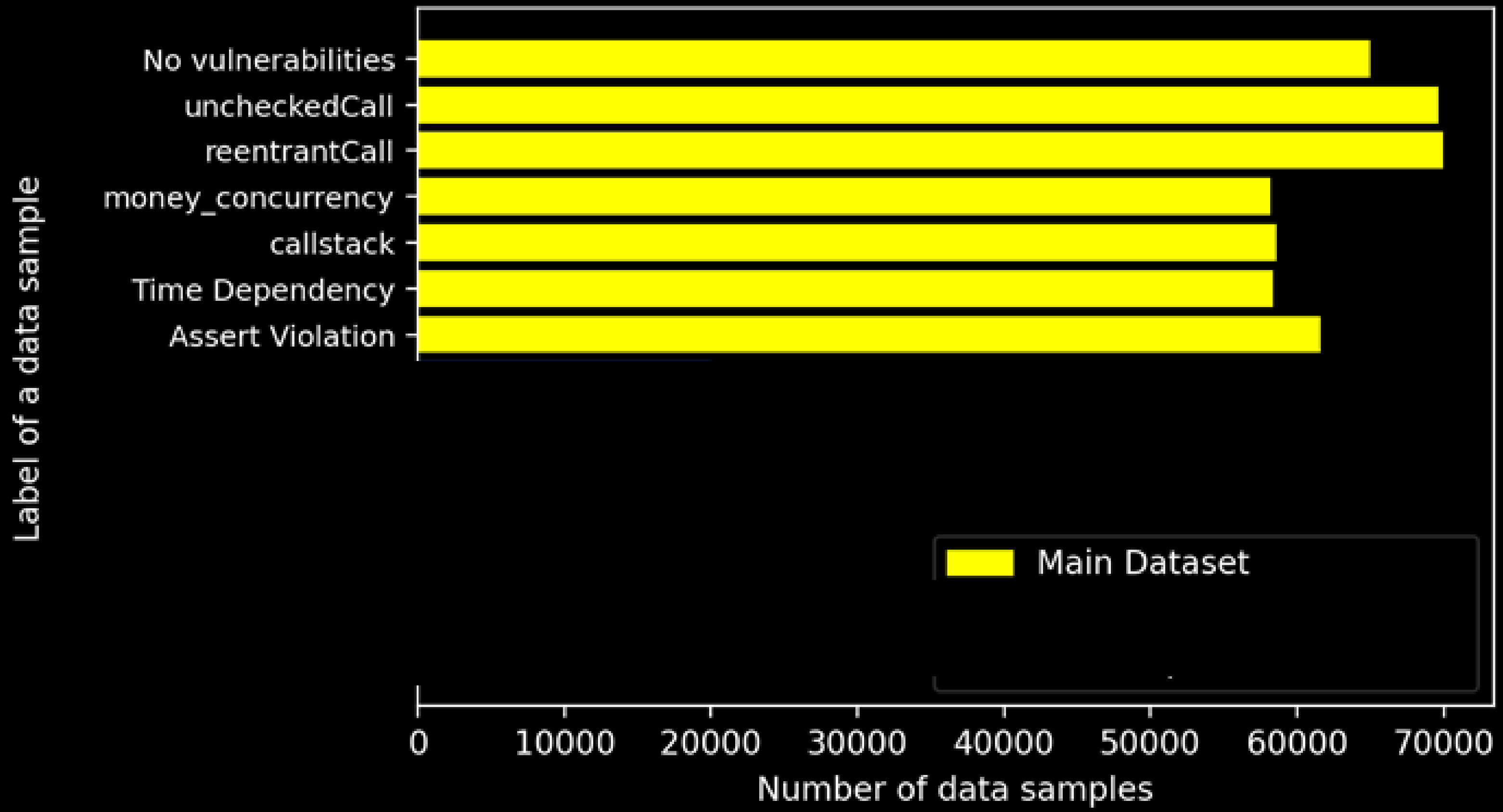


Ethereum and Testnets

Mythril, Oyente, Vandal, Maian

Our Datasets

- 279.726 instances after cleaning up and deduplicating ~3.6 million smart contracts
- **Main Dataset** is used in initial training (ca. 60.000 samples per vulnerability)
- **Extension Dataset** is utilized for Transfer Learning (ca. 20.000 samples per vulnerability)
- **Underrepresented Dataset** is used for Transfer Learning to show applicability for minority classes
- Labeling done using 3 vulnerability scanning tools: Mythril (T1), Oyente (T2), Vandal (T3)



80% training set
10% validation set
10% test set

Evaluation of Model and Transfer Learning

- We can detect all 11 vulnerabilities using single scan
- Efficient inference: scanning the smart contract in less then 0.2 sec (with GPU)

Ground Truth Analysis

- Studied thousands of security audits
- 373 available, compilable, and relevant samples

Conclusion

- We presented DNN-based vulnerability detection approach for smart contracts
- ESCORT is the first framework extendable to new vulnerability types
- It has good effectiveness across different vulnerability classes
- It operates directly on bytecode, yet independent from decompilers
- It has superior performance during inference time
- Future work
  - Investigating the effectiveness of transfer learning with less training data
  - Localization of vulnerabilities in bytecode