Data-driven Cyber Security to Counterfeit Malicious Attacks

Yang Xiang

Swinburne University of Technology, Australia

yxiang@swin.edu.au

Using digital technology to enable businesses and industries beyond traditional boundaries

Social Innovation

- Environment
- Healthcare
- Education
- Urban Development
- Food

Technology Innovation

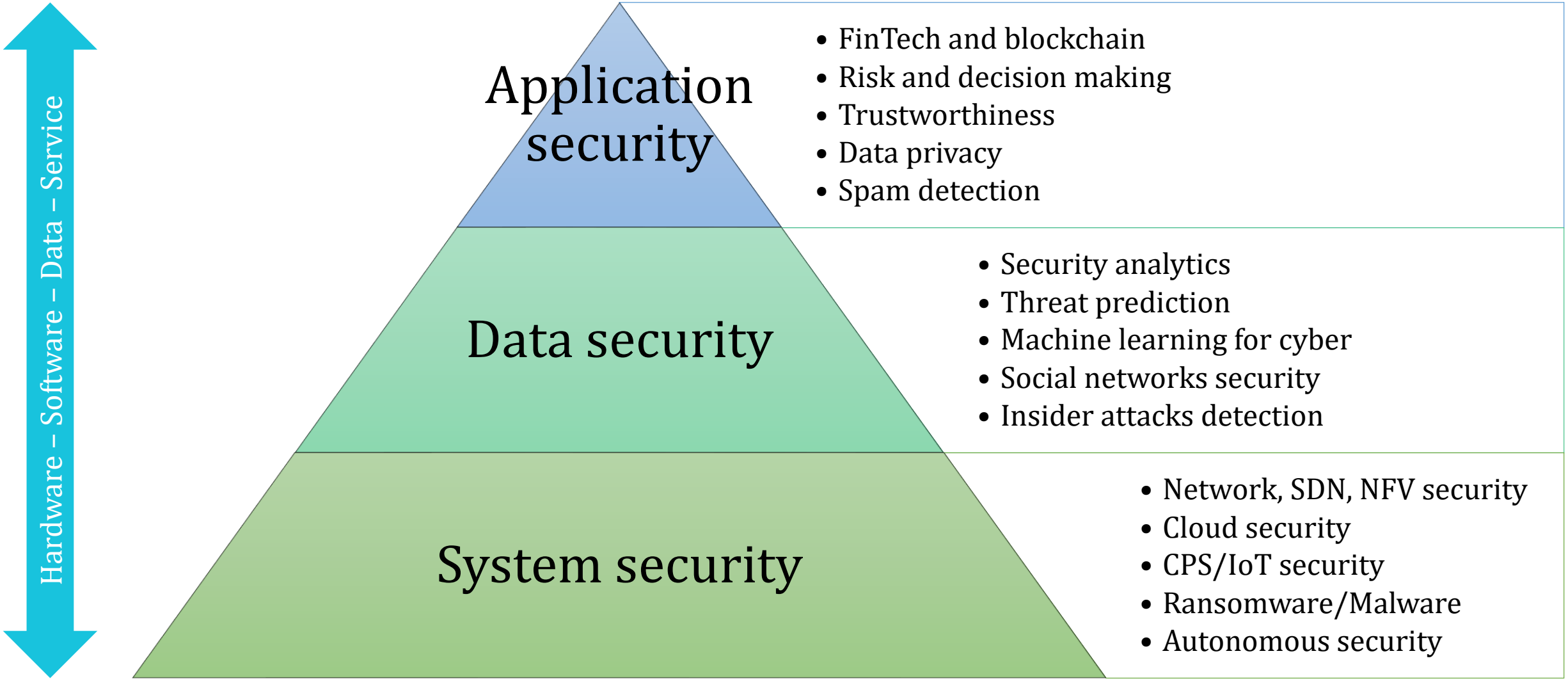
- Finance
- Manufacturing
- Marketing
- Retail
- Transport

Technology Drive

- Data Analytics
- Cybersecurity
- Visualization
- IoT
- Blockchain
- AI

Digital Research Innovation Capability Platform

Cybersecurity Lab Core Capabilities

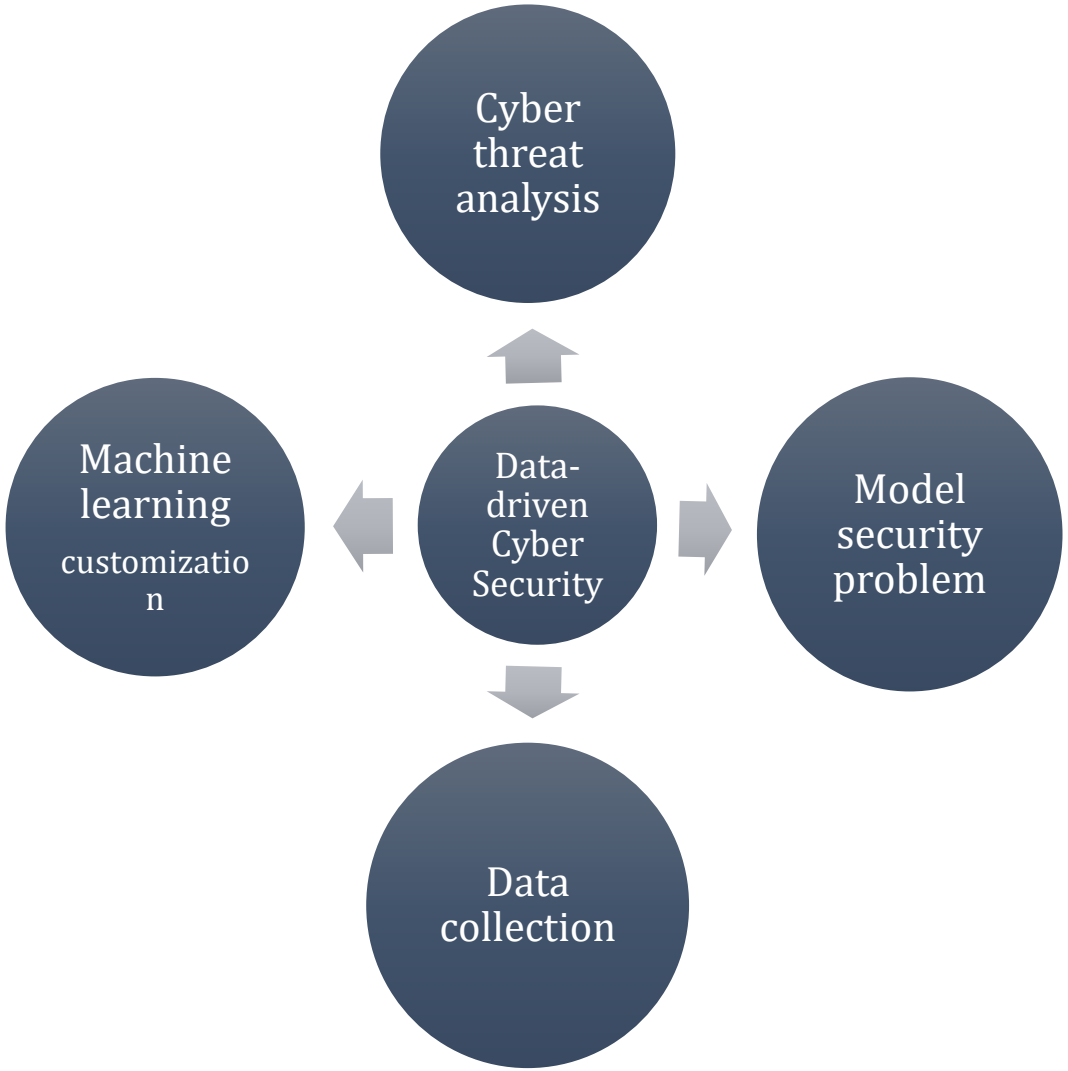


Real-world Data

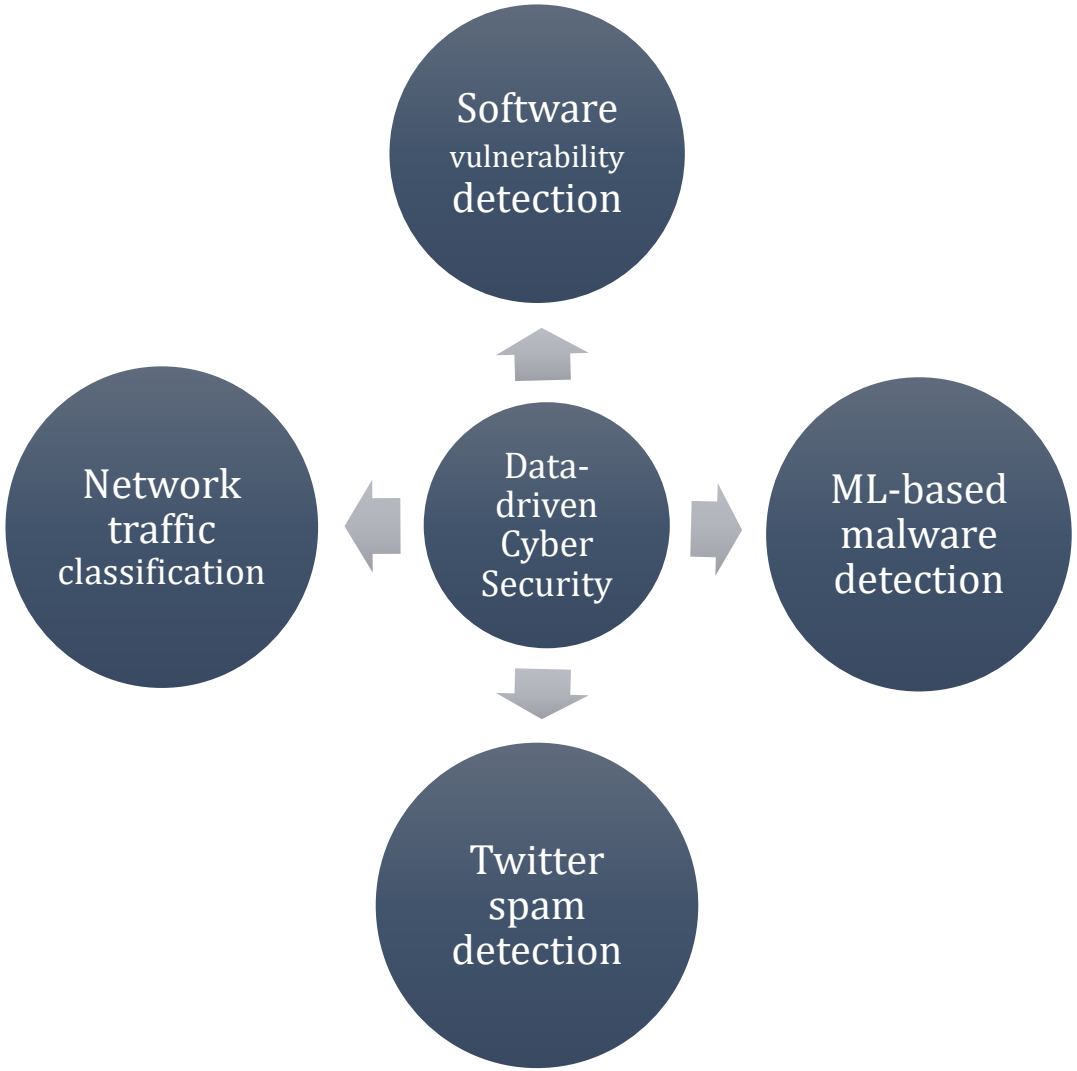
Modelling

+ Reasoning

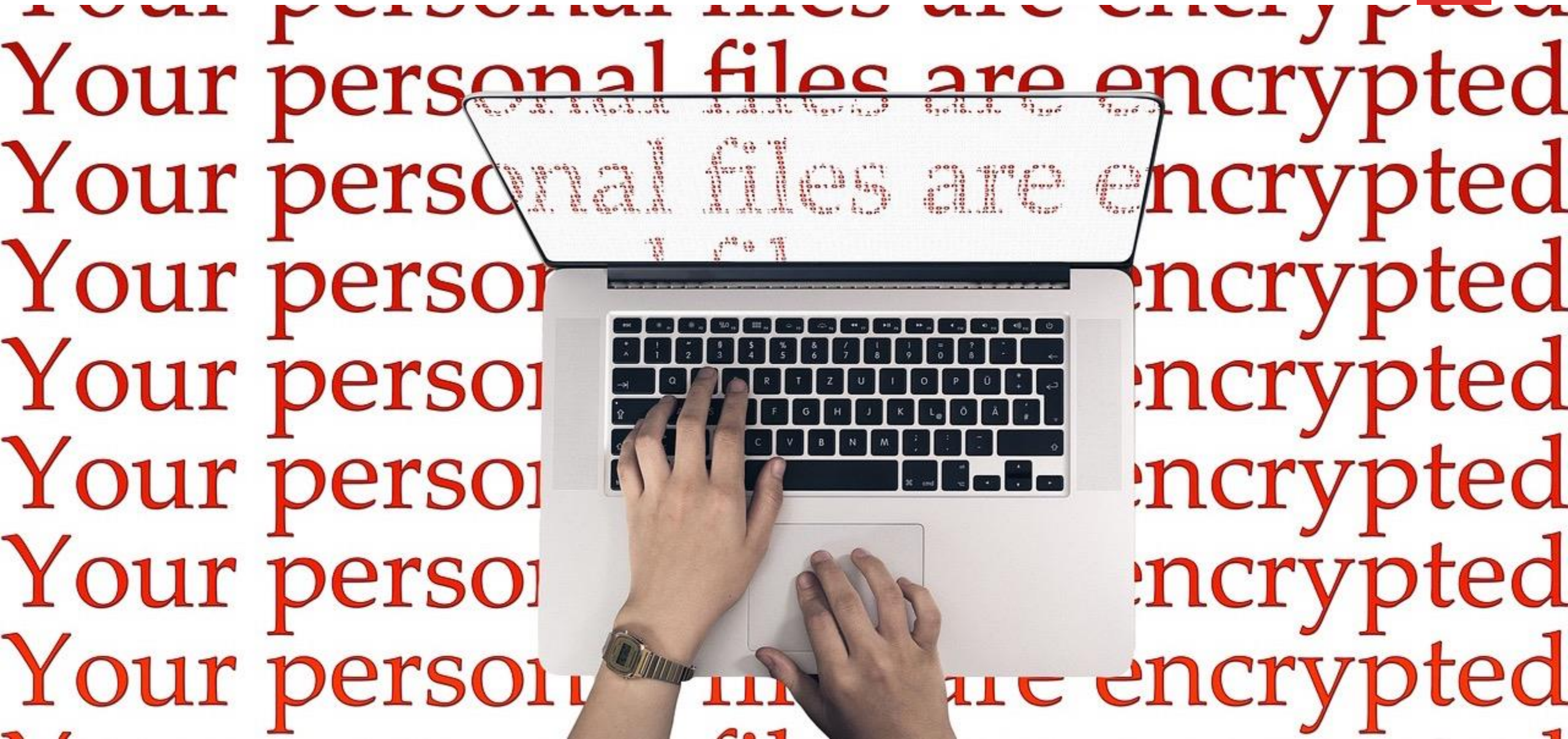
Research Methodology



Examples


Software vulnerability detection
Network traffic classification
Data-driven Cyber Security
ML-based malware detection
Twitter spam detection

Software vulnerability detection



2014

500,000
Servers
Affected

Millions
Servers
Attacked

2017

150
Countries
Affected

$4
Billion
Loss

2017

7~8%
CPU Loss

2018

Intel
SGX

$XXX Loss

Challenge 1

Software Complexity

45 million lines

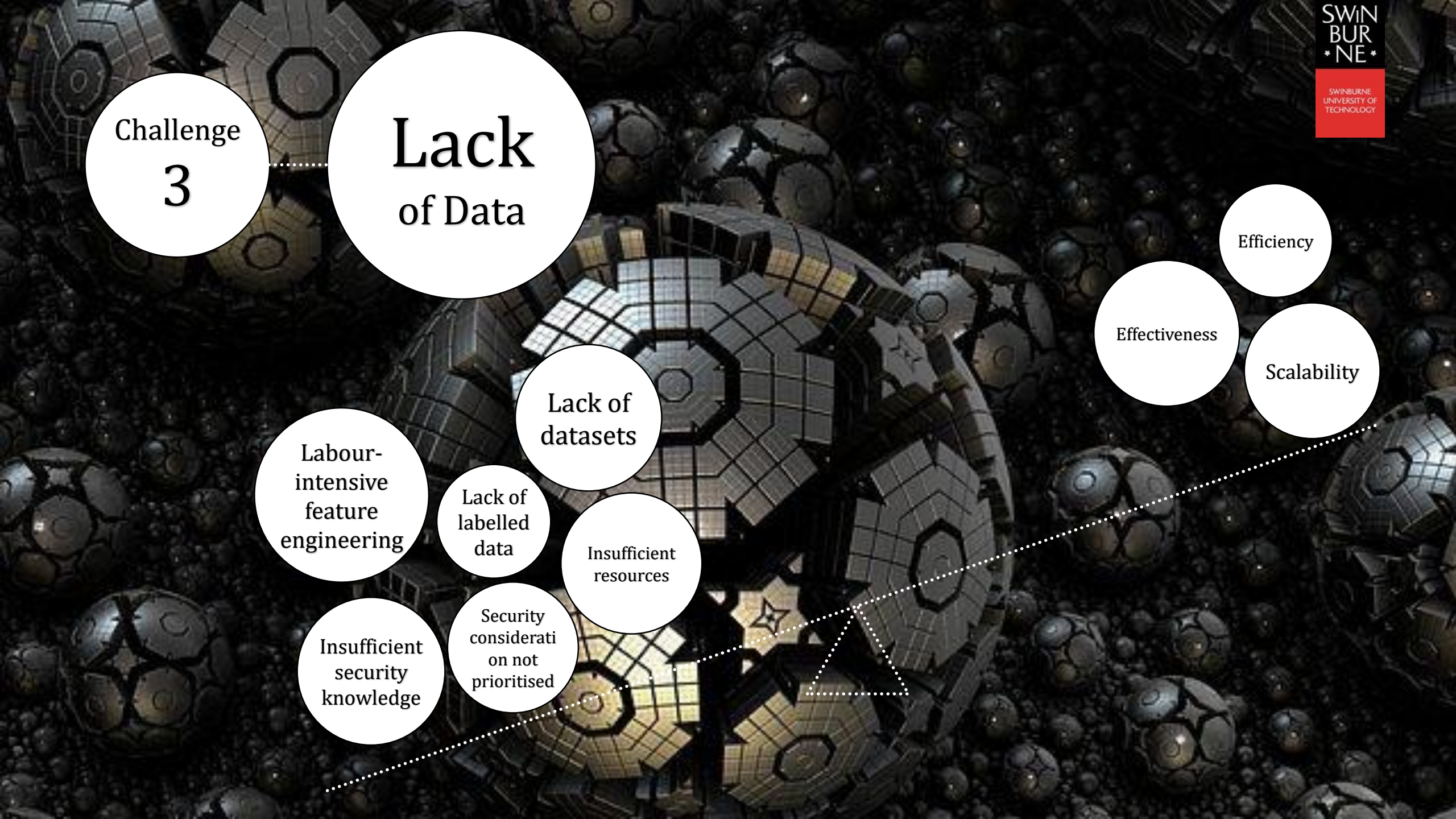
61 million lines

70 million lines

100+ million lines

SWINBURNE UNIVERSITY OF TECHNOLOGY


Challenge
2
Vulnerability
Numbers
2015
6,480
2016
6,447
2017
14,714
2018
20,000+
54+/day
SWINBURNE
UNIVERSITY OF
TECHNOLOGY


Challenge 3
Lack of Data
Lack of datasets
Labour-intensive feature engineering
Lack of labelled data
Insufficient resources
Insufficient security knowledge
Security considerati on not prioritised
Efficiency
Effectiveness
Scalability
SWINBURNE UNIVERSITY OF TECHNOLOGY

Observations

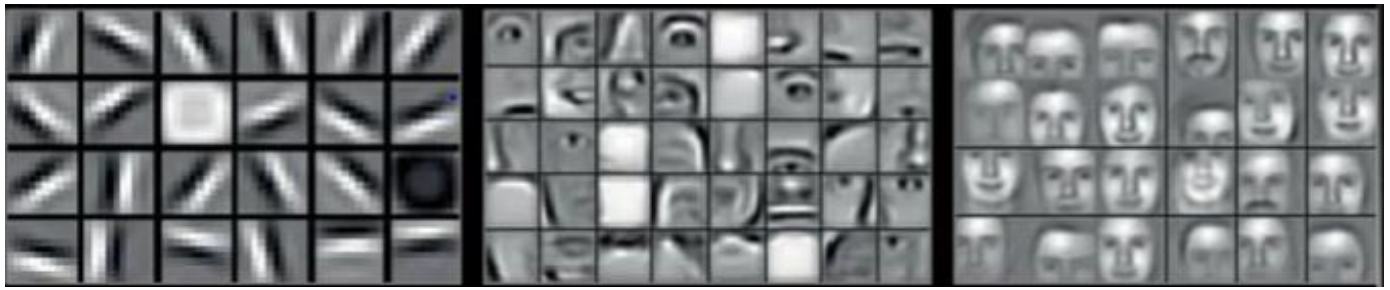
- **Abstract Syntax Trees (ASTs)**: an effective code representations.
- Software source code shares similar statistical properties to **natural language**.
- Vulnerabilities from different projects **share common knowledge,** which is discoverable by deep learning algorisms.

Representations learning

The input

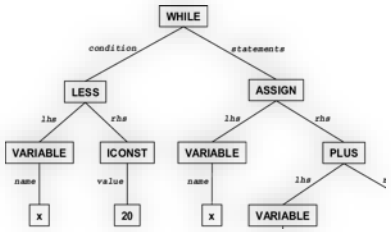
CNN

Low-level features Mid-level features High-level features



```
1  int foo (int a)
2  {
3     int c = 0;
4     /* Invoke function bar() */
5     int b = bar (a);
6     if (a == 2)
7     {
8        c = a + b;
9        return c;
10    }
11    return b;
12 }
```

AST



RNN

2.59E-01	1.37E+00	6.90E-01	-4.14E-01	1.41E+00	-1.53E+00	-1.42E+00	-2.28E-01
1.75E-03	1.22E+00	2.61E-01	-2.50E-01	1.10E+00	-1.30E+00	-1.33E+00	-1.32E-01
1.96E-01	4.06E-01	-2.32E-01	-7.27E-02	4.20E-01	-8.16E-01	-9.44E-01	-2.55E-01
1.13E-01	3.21E-01	-1.87E-01	-1.20E-01	5.27E-01	-6.96E-01	-8.28E-01	-3.29E-01
2.35E-01	8.47E-01	2.08E-01	-2.08E-01	1.09E+00	-1.20E+00	-1.01E+00	-1.77E-01
3.80E-01	7.35E-01	-6.99E-02	-1.61E-01	9.72E-01	-1.01E+00	-8.37E-01	-1.12E-02
2.66E-01	1.15E+00	6.23E-01	-3.41E-01	1.17E+00	-1.46E+00	-1.13E+00	-2.23E-01
4.03E-01	8.44E-01	1.07E-01	-2.43E-01	1.13E+00	-1.01E+00	-9.20E-01	4.43E-02
2.59E-01	1.46E+00	6.61E-01	-4.71E-01	1.69E+00	-1.66E+00	-1.43E+00	-1.87E-01
2.13E-01	1.11E+00	5.76E-01	-3.41E-01	1.35E+00	-1.42E+00	-1.26E+00	-1.19E-01
3.08E-01	1.50E+00	7.92E-01	-5.18E-01	1.64E+00	-1.51E+00	-1.46E+00	-2.12E-01

Latent, abstract features describing programming patterns/characteristics

Methodology

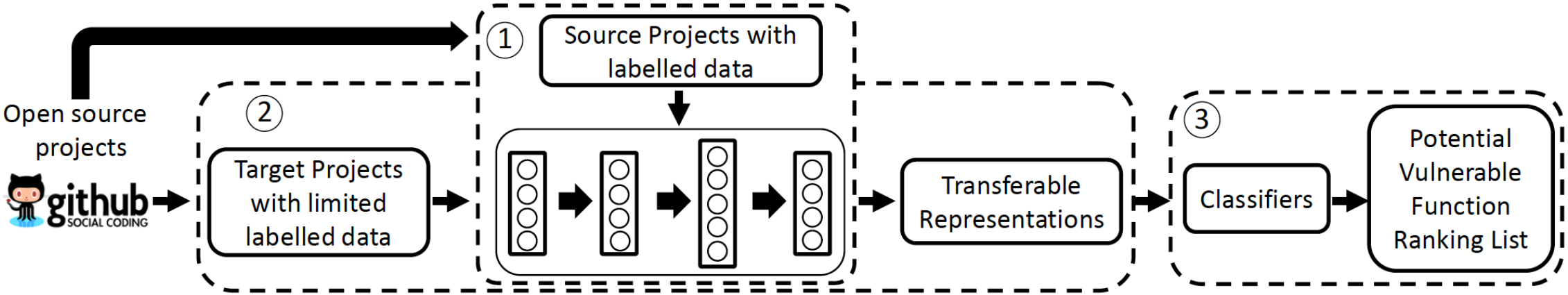


Fig. 1: The proposed framework for vulnerability discovery. It contains 3 stages: the first stage is to pre-train a Bi-LSTM network using source code projects; the second stage is to feed the trained network with the target project to obtain representations as features; the last stage is to train a ML classifier with the learned features.

Network Architecture

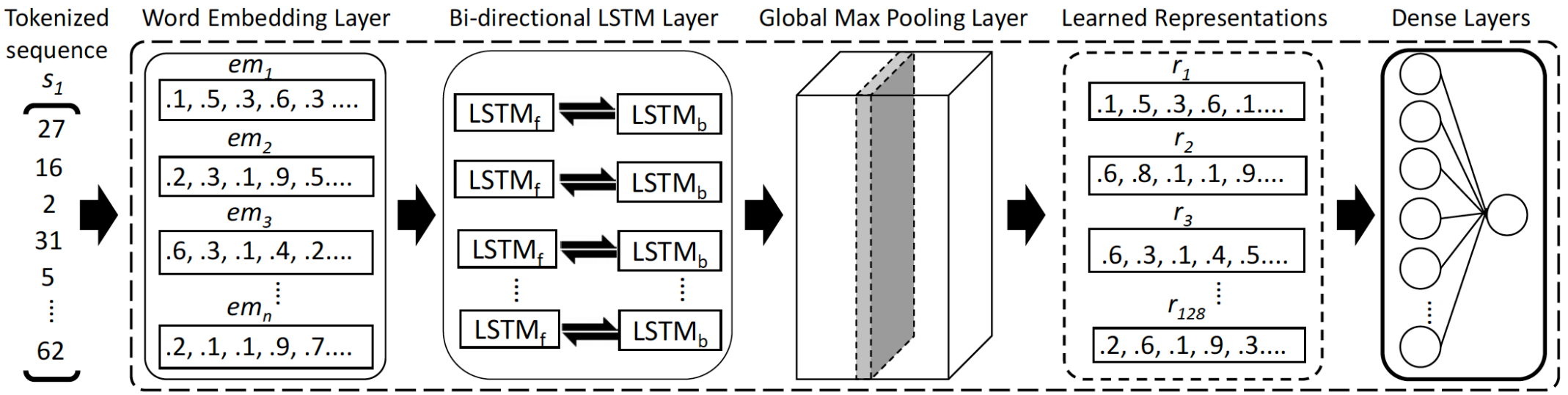
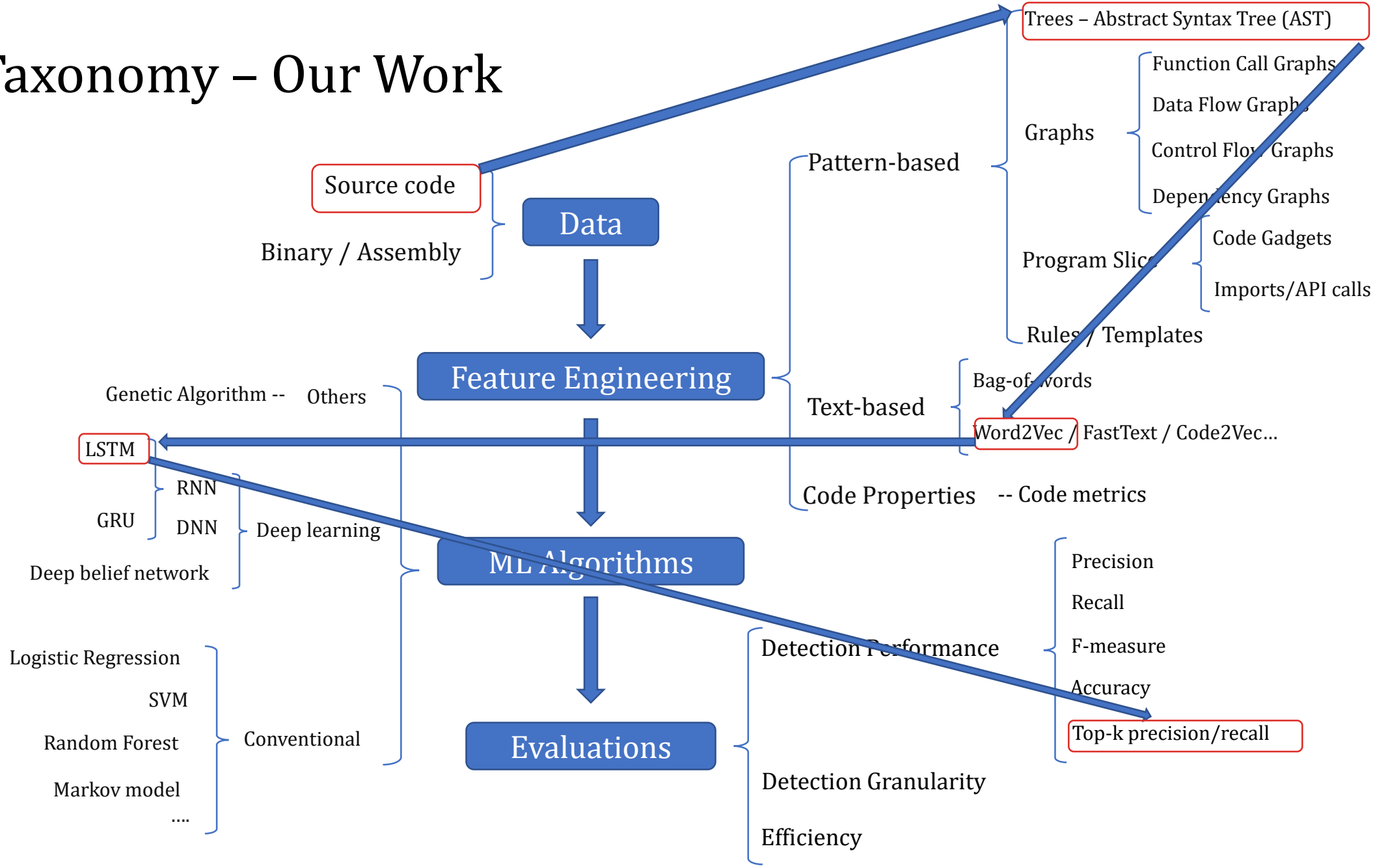
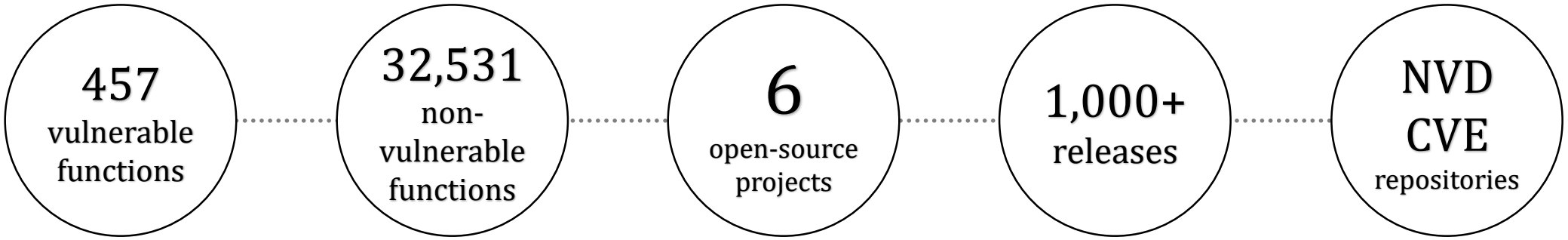


Fig. 4: The 5-layer architecture of the proposed LSTM network for learning deep AST representations. During the pre-training phase, the network takes a tokenized sequence converted from an AST as an input. In the representation learning phase, the last two dense layers are removed and the output of global max pooling layers are used as the learned deep AST representations as features for subsequent processing.

Taxonomy – Our Work



The Datasets



Project	# Vulnerable Functions Labeled	# Non-vulnerable Functions Used
LibTIFF	96	777
LibPNG	43	499
FFmpeg	191	4921
Pidgin	29	8050
VLC Media Player	42	3636
Asterisk	56	14648

Results

NUM	func_id	Type	Probabilities of Being Vulnerable	Label
1	_cve-2012-0854.txt.txt	Buffer Errors	0.966666667	1
2	_cve-2013-2277.txt.txt	Other	0.9	1
3	_cve-2016-6671.txt.txt	Buffer Errors	0.866666667	1
4	_cve-2012-2777.txt.txt	Insufficient Information	0.85	1
5	_cve-2013-7016.txt.txt	Buffer Errors	0.85	1
6	_cve-2012-2779.txt.txt	Insufficient Information	0.85	1
7	_cve-2012-2775.txt.txt	Insufficient Information	0.833333333	1
8	_cve-2012-2790.txt.txt	Insufficient Information	0.833333333	1
9	_cve-2012-2796.txt.txt	Insufficient Information	0.8	1
10	_vf_pullup.c_decide_frame_length.c.txt		0.733333333	0
11	_cve-2010-4704.txt.txt	Input Validation	0.716666667	1
12	_cve-2016-2329-2.txt.txt	Buffer Errors	0.7	1
13	_ffmpeg_dxva2.c_dxva2_create_decoder.c.txt		0.666666667	0
14	_cve-2014-9603.txt.txt	Input Validation	0.666666667	1
15	_cve-2015-8365.txt.txt	Buffer Errors	0.65	1
16	_cve-2016-2330-1.txt.txt	Buffer Errors	0.65	1
17	_cve-2012-2794.txt.txt	Insufficient Information	0.616666667	1
18	_cve-2011-3940.txt.txt	Buffer Errors	0.6	1
19	_8bps.c_decode_frame.c.txt		0.6	0
20	_ffplay.c_decoder_decode_frame.c.txt		0.6	0
21	_audio_mix.c_ff_audio_mix_set_matrix.c.txt		0.6	0
22	_vf_sab.c_open_filter_param.c.txt		0.583333333	0
23	_ffmpeg.c_init_output_stream.c.txt		0.566666667	0
24	_mpeg12dec.c_decode_chunks.c.txt		0.566666667	0
25	_opencl.c_init_opencl_env.c.txt		0.566666667	0

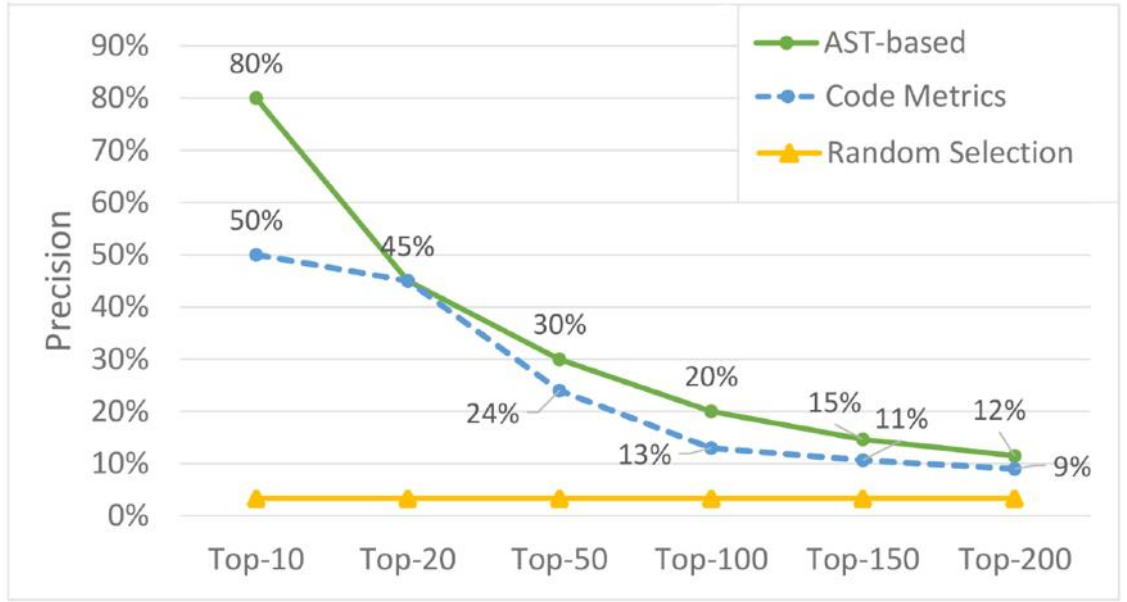
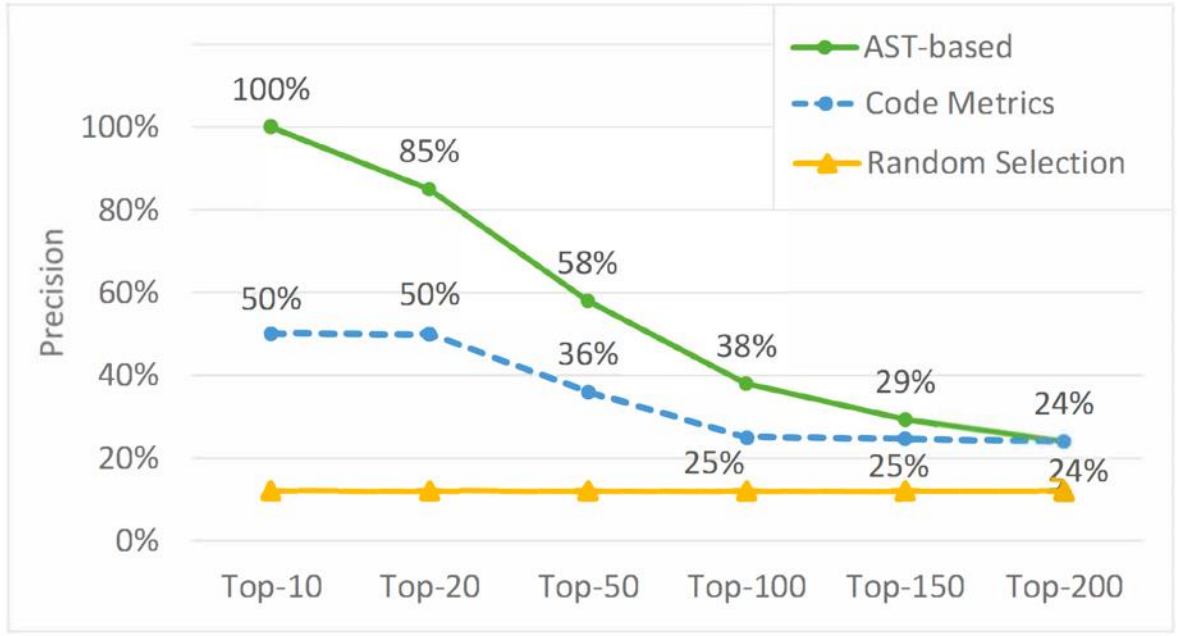
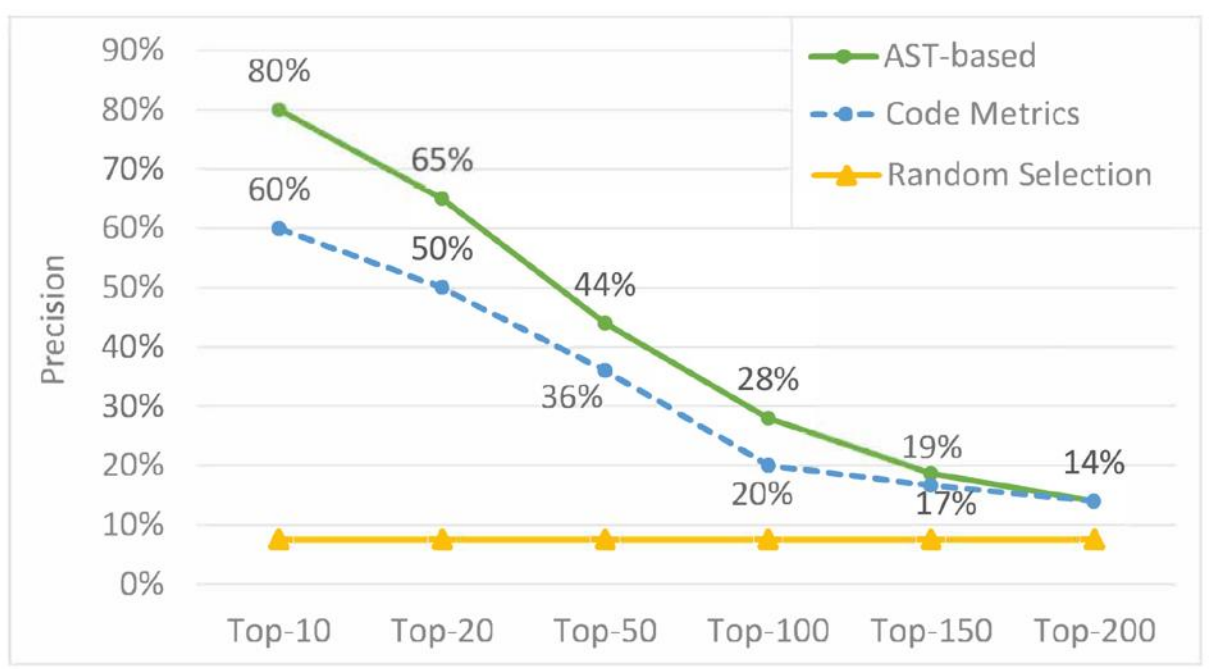


Fig. 5: Precision comparison between deep AST representations (AST-based), CMs and random selection on FFmpeg.

Results



(a) Testing on LibTIFF



(b) Testing on LibPNG

Results

(c) Testing on FFmpeg

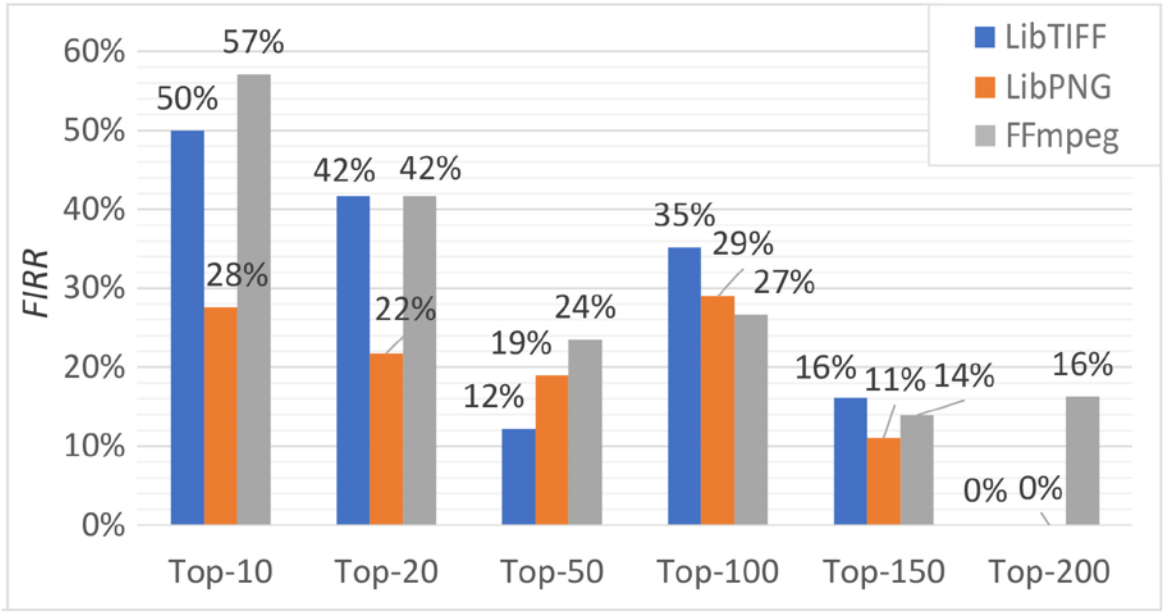
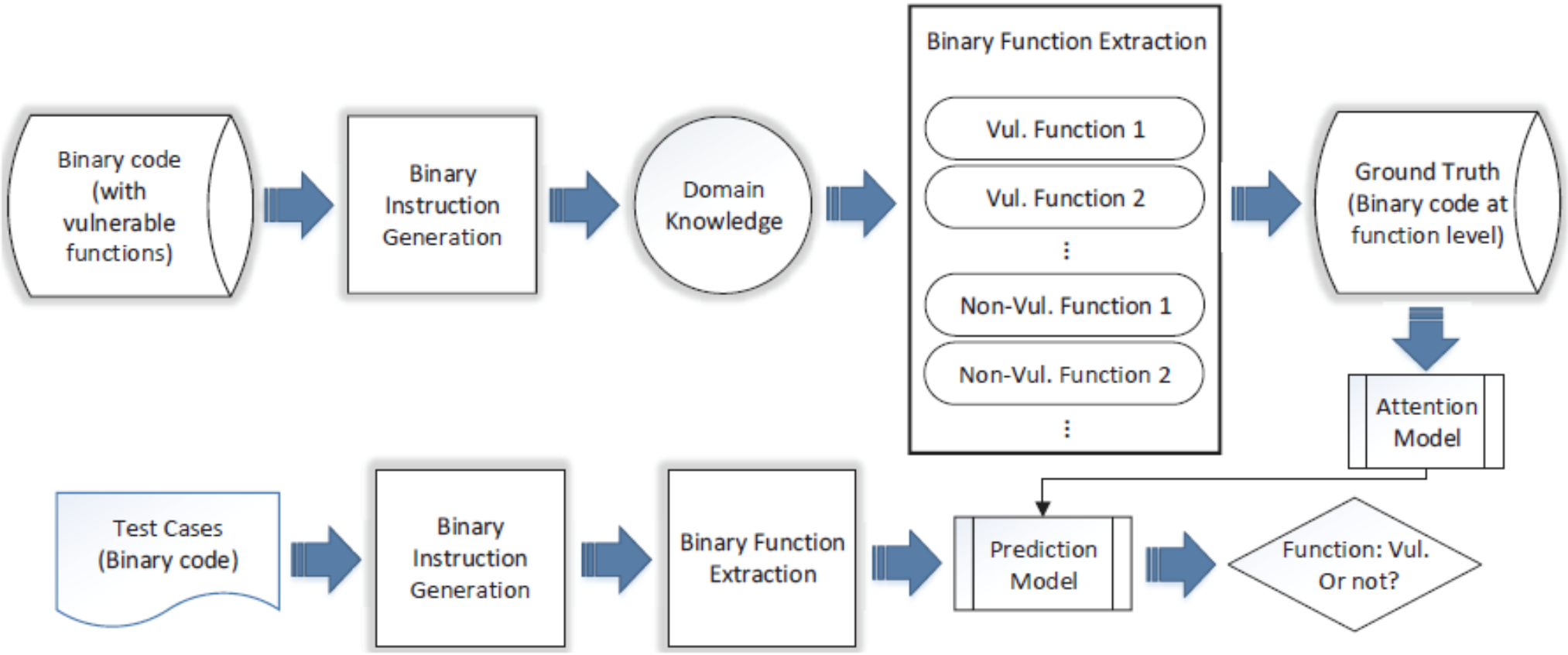
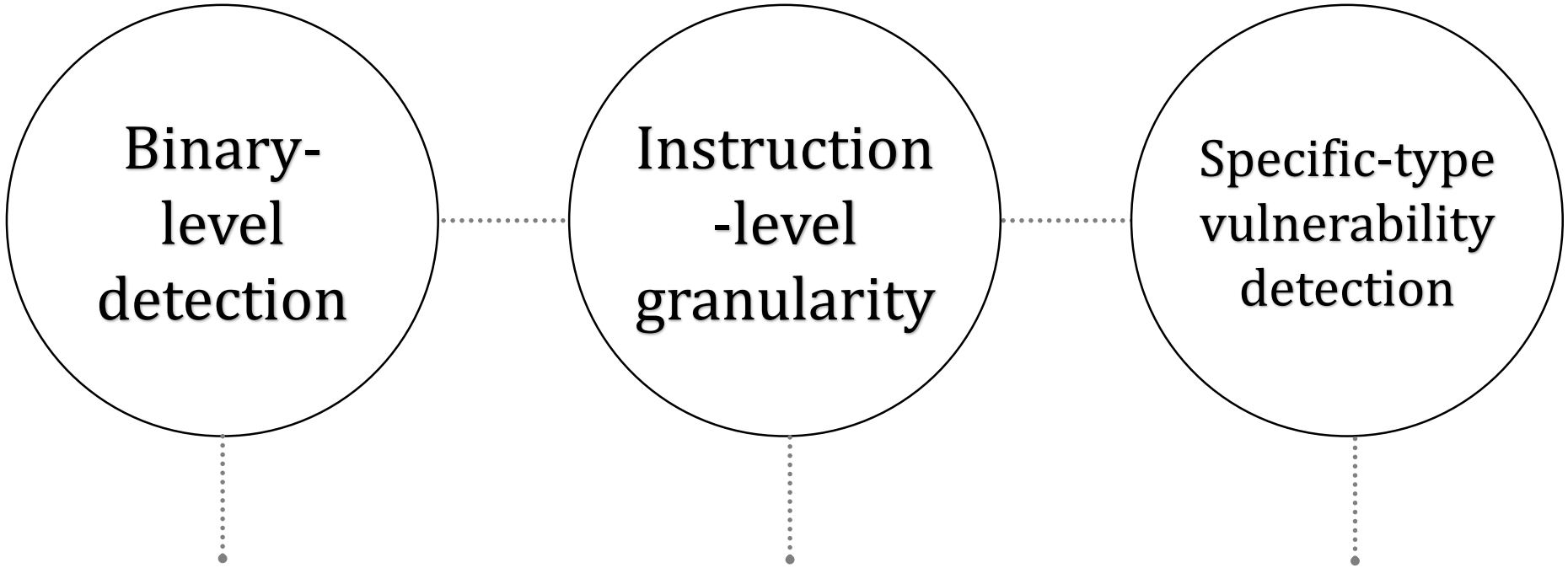


Fig. 7: FIRR of transfer-learned AST representations to CMs as features.

Binary Vulnerability Detection



Future Work


Binary-level detection
Instruction-level granularity
Specific-type vulnerability detection
Focusing on scenarios where the source code is unavailable
Identifying multiple instructions (reverse-engineering) that are potentially vulnerable
Focusing on vulnerabilities caused by missing checks (e.g. numeric errors).

Example 2 - ML-based malware detection

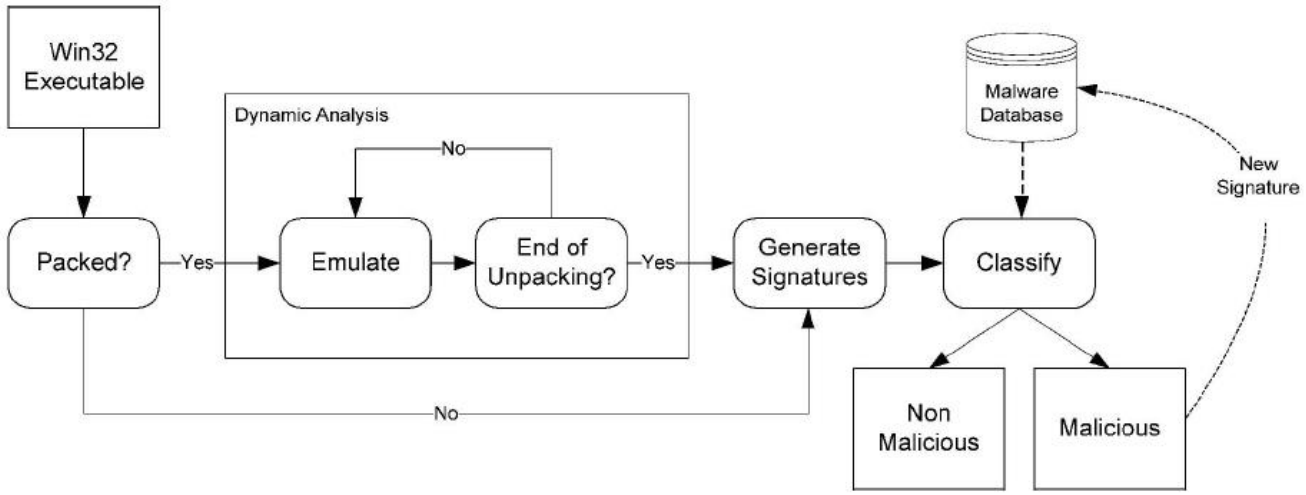


Fig. 1. Block diagram of the malware classification system.

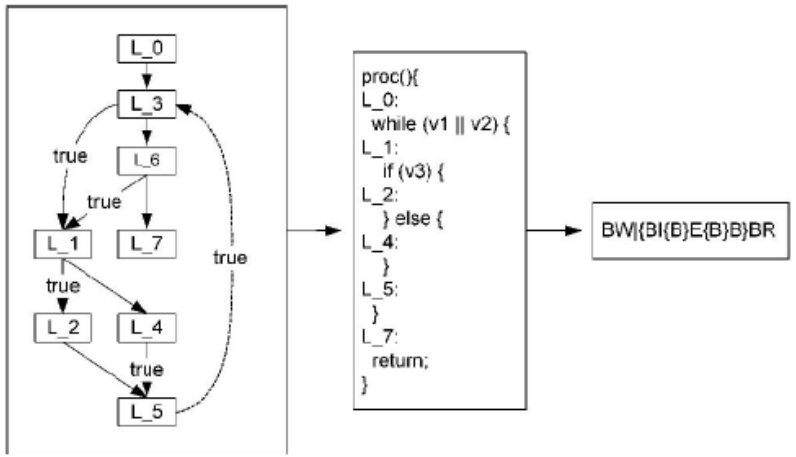


Fig. 4. The relationship between a control flow graph, a high-level structured graph, and a signature.

Example 3 – Twitter spam detection

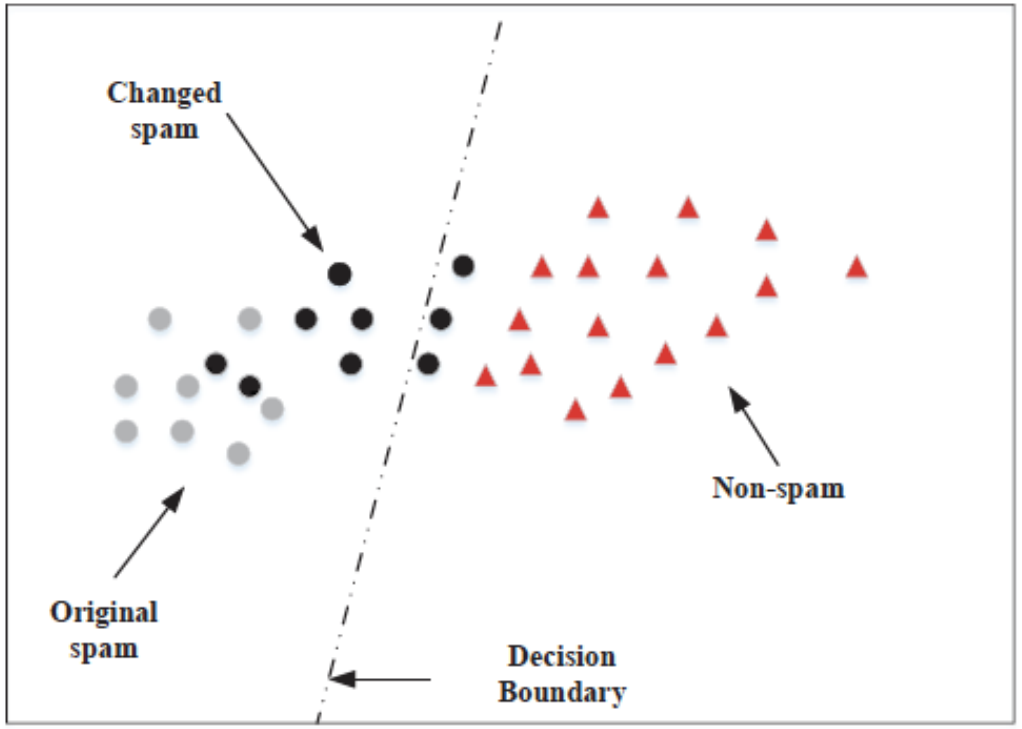


Fig. 2: Illustration of "Spam Drift"

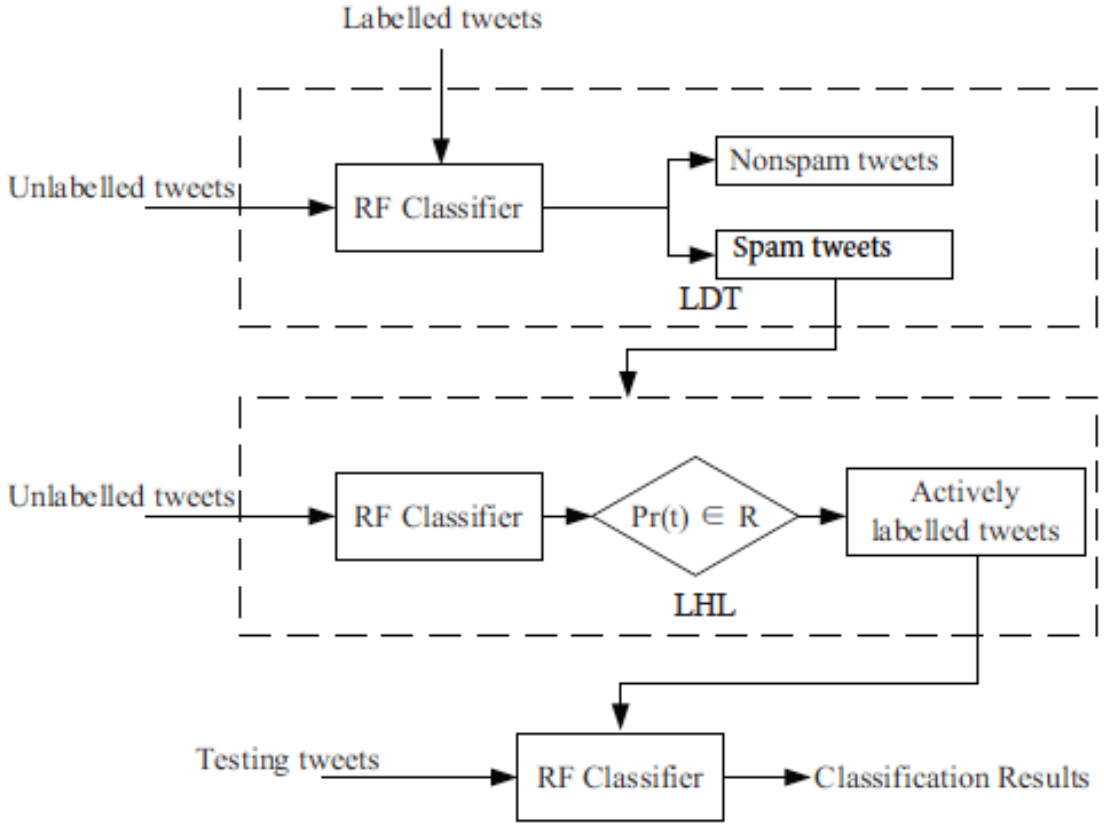


Fig. 3: Lfun Framework

Example 4 - Network traffic classification

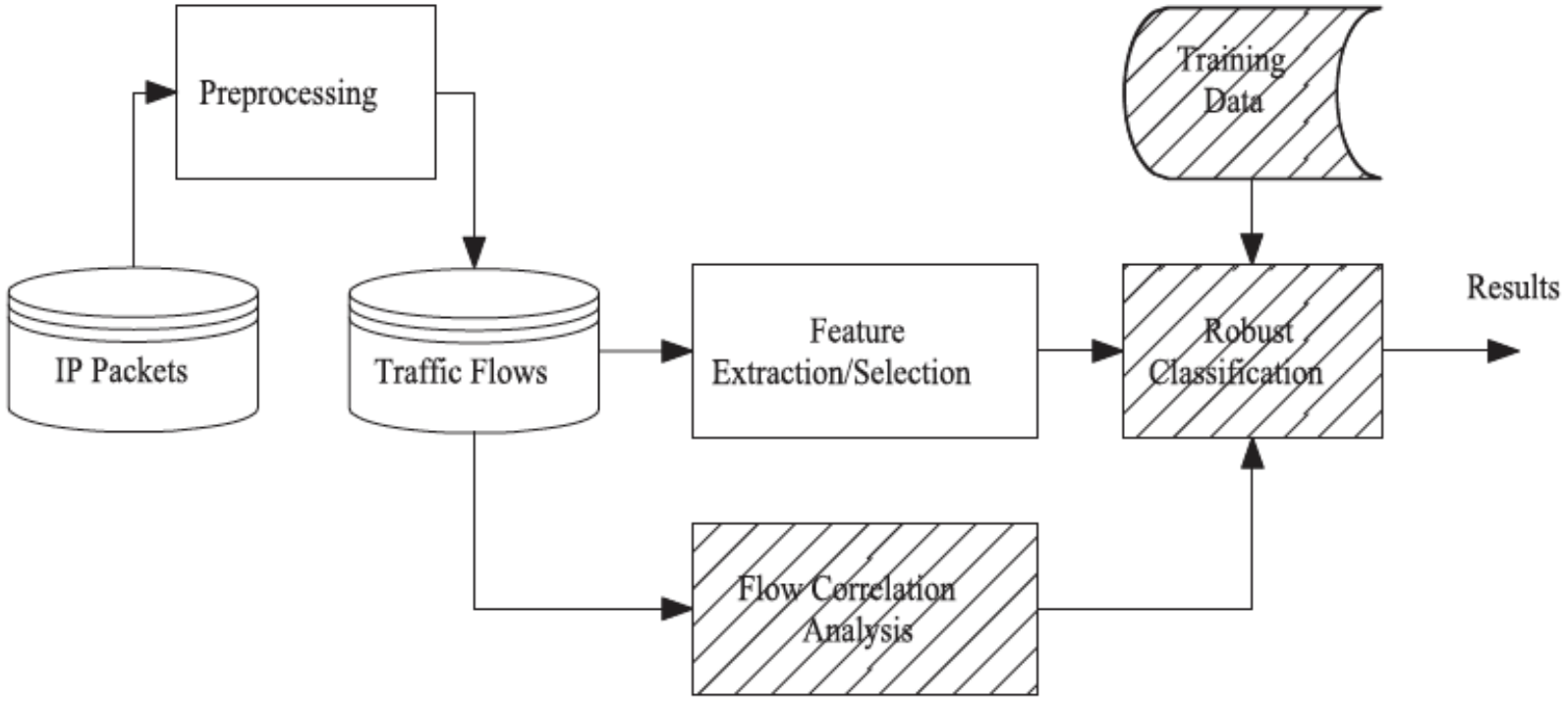


Fig. 1. A new traffic classification system model.

Data-driven Cyber Security

Security professionals

Domain knowledge

Model analytics

Resources

- G. Lin, J. Zhang, W. Luo, L. Pan, Y. Xiang, O. D. Vel, and P. Montague, “Cross-Project Transfer Representation Learning for Vulnerable Function Discovery,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3289-3297, 2018.
- C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min, “Statistical Features Based Real-time Detection of Drifted Twitter Spam,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 914-925, 2017.
- J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, “Robust Network Traffic Classification,” *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257-1270, 2015.
- S. Cesare, Y. Xiang, and W. Zhou, “Control Flow-based Malware Variant Detection,” *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 4, pp. 307-317, 2014.
- S. Cesare, Y. Xiang, and W. Zhou, “Malwise - An Effective and Efficient Classification System for Packed and Polymorphic Malware,” *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1193-1206, 2013.
- J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, “Network Traffic Classification Using Correlation Information,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 104-117, 2013.

Sponsors & Collaborators

Australian Government
Australian Research Council

Australian Government
Department of Defence
Defence Science and Technology Group

DATA 61

CSIRO

THE UNIVERSITY OF MELBOURNE

MONASH University