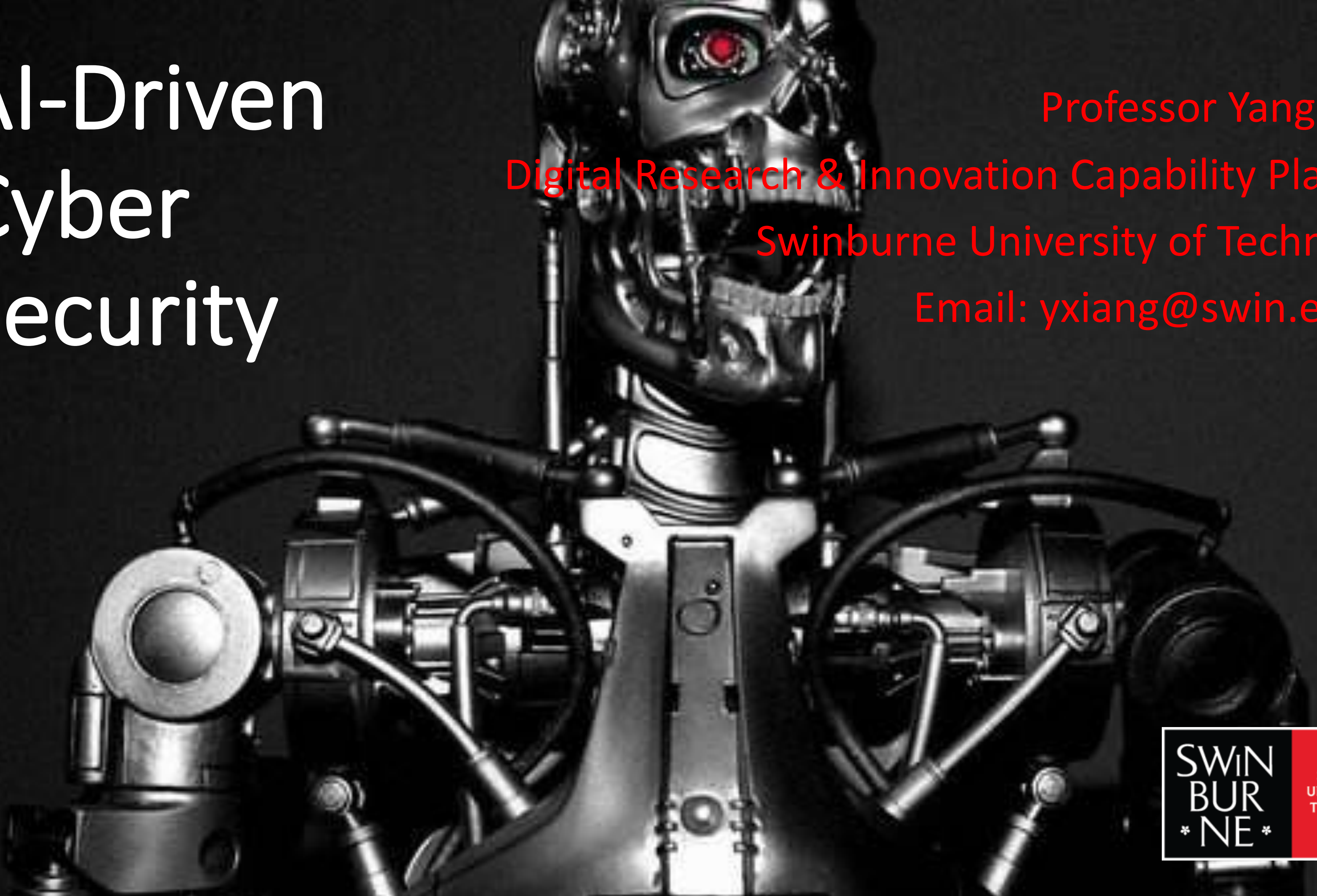
AI-Driven Cyber Security



Professor Yang Xiang
Digital Research & Innovation Capability Platform
Swinburne University of Technology
Email: yxiang@swin.edu.au

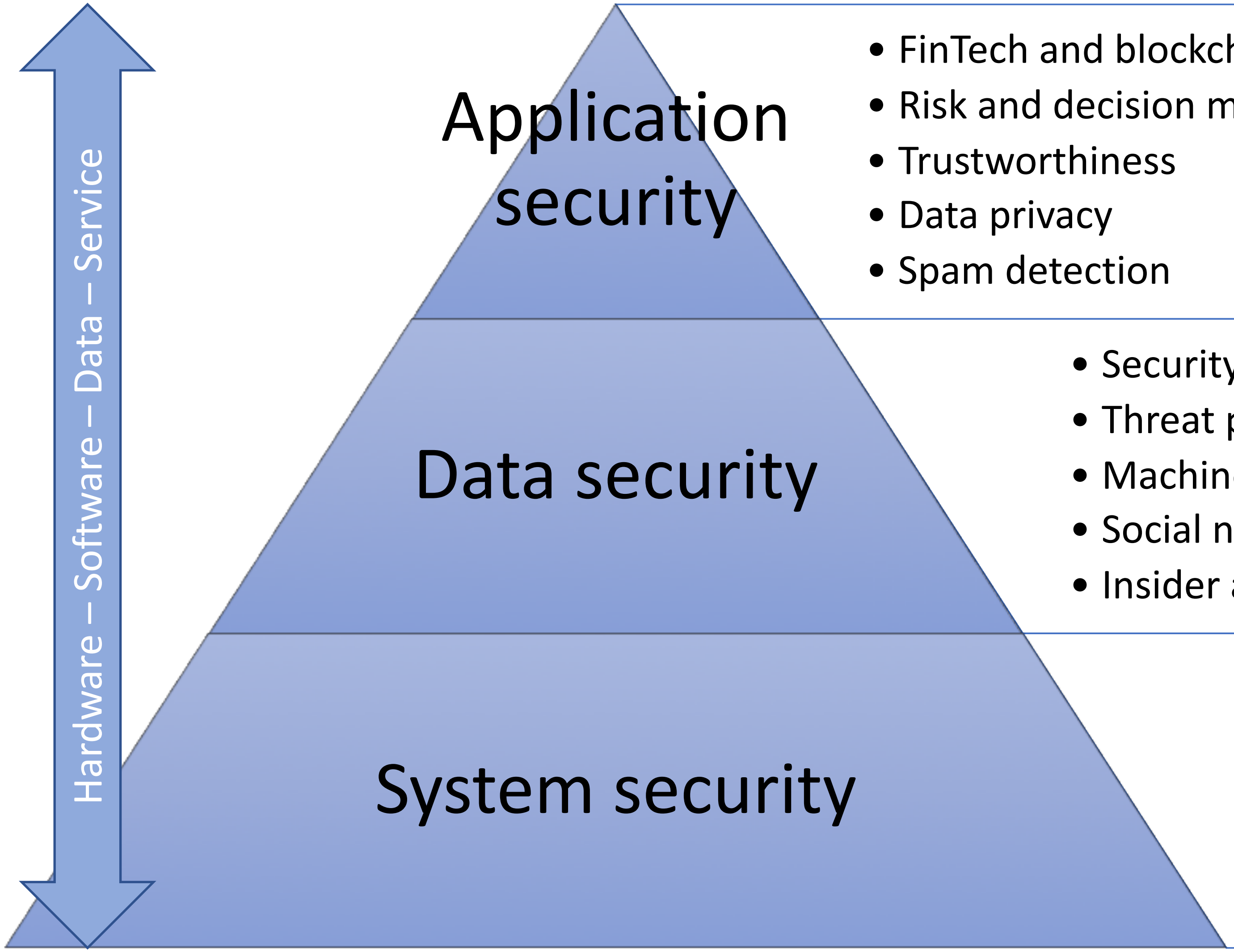
SWINBURNE UNIVERSITY OF TECHNOLOGY

Research on Cyber Security @ Swinburne

- A team of world leading capabilities in cyber security
- We develop innovative technologies for securing cyberspace
- We work with industry and government to provide protection from major cyber security threats

SWINBURNE UNIVERSITY OF TECHNOLOGY

Core Capabilities @ Swinburne



- FinTech and blockchain
- Risk and decision making
- Trustworthiness
- Data privacy
- Spam detection

- Security analytics
- Threat prediction
- Machine learning for cyber
- Social networks security
- Insider attacks detection

- Network, SDN, NFV security
- Cloud security
- CPS/IoT security
- Ransomware/Malware
- Autonomous security

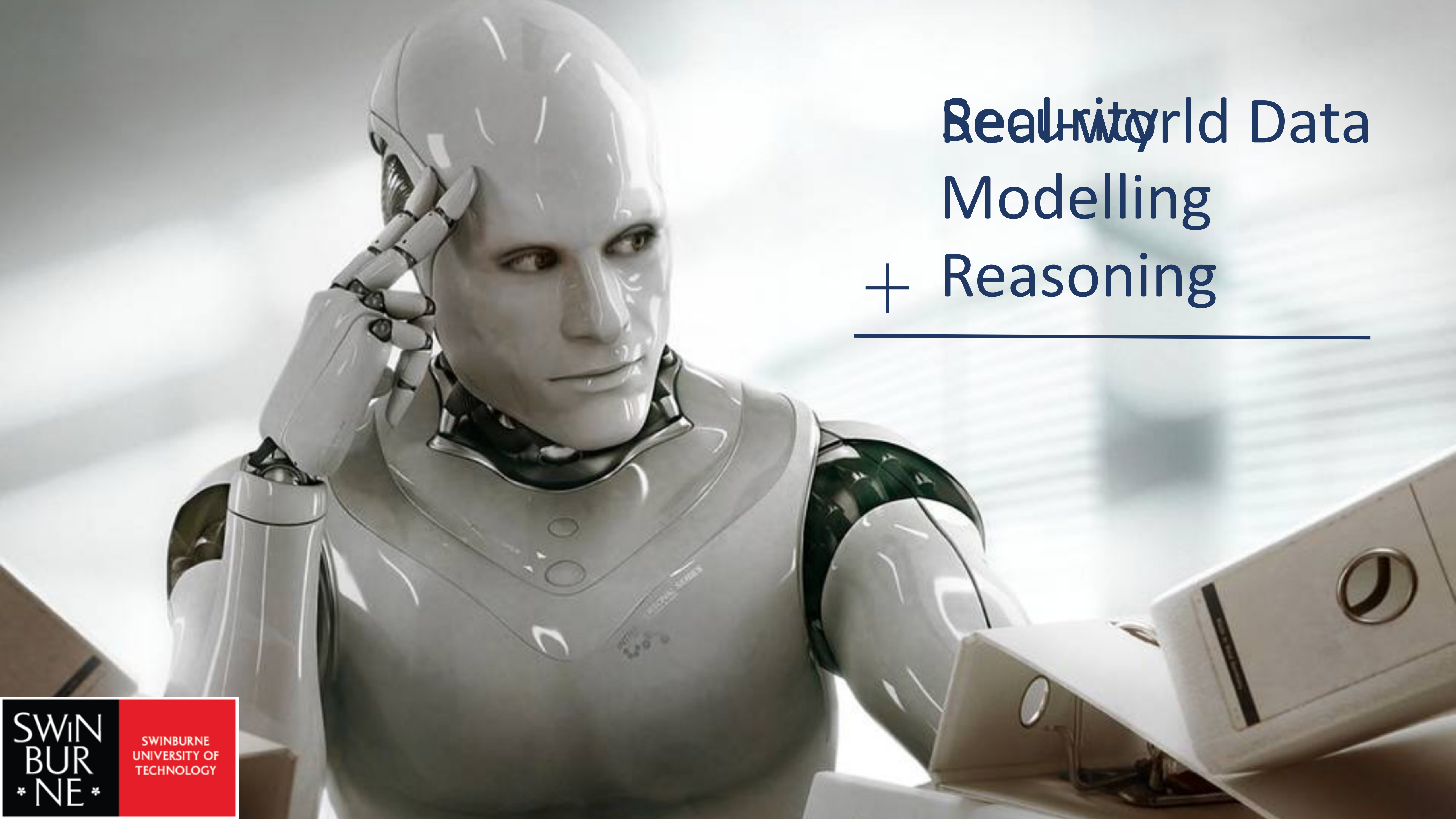
Cutting Edge Facilities

- Dedicated Equipment

- Huge Amount of Data

How AI models learn and understand what is normal and what is abnormal?

SWINBURNE UNIVERSITY OF TECHNOLOGY

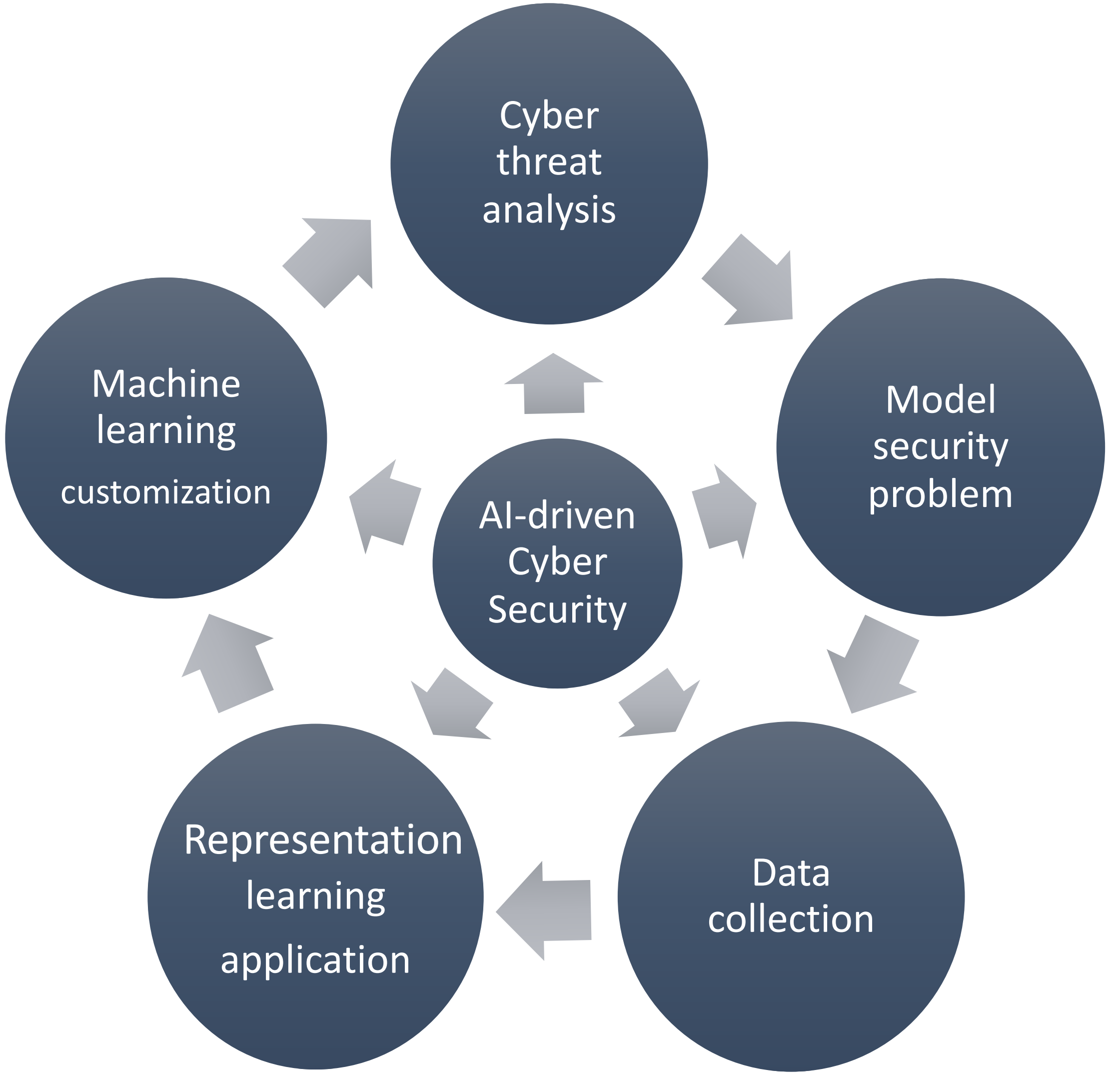


Security

Real-world Data Modelling
+ Reasoning

SWINBURNE UNIVERSITY OF TECHNOLOGY

Research Methodology



SWINBURNE UNIVERSITY OF TECHNOLOGY

Use Cases

AI-Driven Cyber Security

Deep vulnerable code analysis	ML based malware detection	Insider attack prediction	Adversarial machine learning	Security incident prediction	Intelligent network monitoring	…

SWIN BUR NE
SWINBURNE UNIVERSITY OF TECHNOLOGY

Project – Ransomware Detection

- A ransomware attack can encrypt critical data, causing huge damages
- Our research
  - Ransomware life cycle
  - Software similarity and classification
  - Propagation modelling
  - Intelligent detection
  - Global sensors


ABC NEWS

Just In | Australia | World | Business | Sport | Science | Arts | Analysis | Fact Che

Print | Email | Facebook | Twitter | More

Cyber expert warns against supporting criminal syndicates amid global hacking

By Katri Uibu, wires

Updated 13 May 2017, 4:14pm

Companies affected by global ransomware attacks should not pay the ransom so as not to feed into the growing business of organised cyber crime, a security expert warns.

Attackers have used encryption algo files, which owners cannot access u a ransom.

Over 57,000 infections in 99 countrie detected, with Russia, Ukraine and T top targets, security software maker

The attacks have led to hospitals an

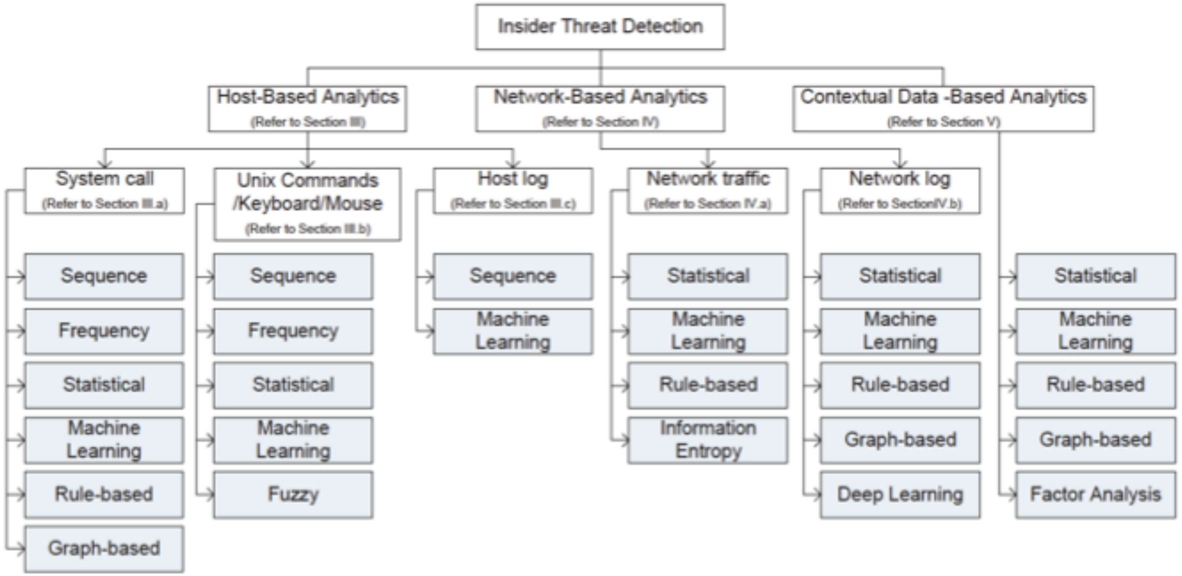
Traffic cameras in Victoria infected by WannaCry ransomware

State government says 55 cameras were affected after a contractor introduced the virus to the system by mistake

Speed cameras affected by the problem will be fixed in the 'next couple of days'. Photograph: Alan Porritt/AAP

Project – Insider Attack Detection

- Insider attacks are highlighted as "the most damaging risk"
- We design a novel fine-grained anomaly behaviour identification system to predict cyber insider attacks
- It can analyse big behaviour data, making real-time decision, and learning varying behaviour features,
- It can provide maximised protection to large-scale private networks

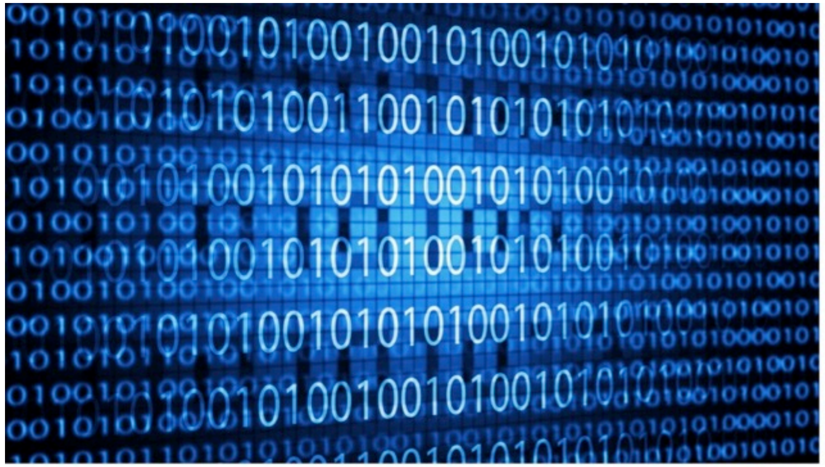


Australian Government
Department of Defence
Defence Science and Technology Group

SWINBURNE UNIVERSITY OF TECHNOLOGY

Project – Deep Learning for Cyber

- Deep Learning techniques to discover vulnerabilities in source and/or binary code bases
  - Deep source analysis: New approaches, classification with representation learning and deep learning with multiple sources for code analysis
  - Deep binary analysis: Innovatively convert binary code to different data presentation such as image, and employ deep neural networks to assist binary analysis



Australian Government
Department of Defence
Defence Science and Technology Group

DATA 61

CSIRO

SWINBURNE UNIVERSITY OF TECHNOLOGY

THE UNIVERSITY OF MELBOURNE

DEAKIN UNIVERSITY AUSTRALIA

Project – Adversarial Machine Learning for Cyber

- Advance the unexplored territory of adversarial machine learning defences, with focus on network security
    - Randomised projection reducing the curse of dimensionality that benefits attackers
    - Game theoretic formulations that seek limit points of cat-and-mouse attack and defence

Australian Government
Department of Defence
Defence Science and Technology Group

DATA 61

CSIRO

SWINBURNE
SWINBURNE UNIVERSITY OF TECHNOLOGY

THE UNIVERSITY OF MELBOURNE

DEAKIN

Project – Classifying Internet Traffic for Security Applications

- Develop a set of novel techniques for Internet traffic classification, which is important to defend against the serious cyber-attacks and effectively minimise the damage
    - Real-time
    - Scalable
    - Robust
    - Private

Australian Government
Australian Research Council

SWINBURNE UNIVERSITY OF TECHNOLOGY

How AI that uses machine learning and other technologies can differentiate benign or harmful binary or source codes?

Detecting Software Vulnerabilities

SWINBURNE UNIVERSITY OF TECHNOLOGY

Significance

Software vulnerabilities can critically:

- undermine the security of computer systems
- endanger the IT infrastructure of organizations

Intel Chip Vulnerabilities

High-impact Vulnerabilities

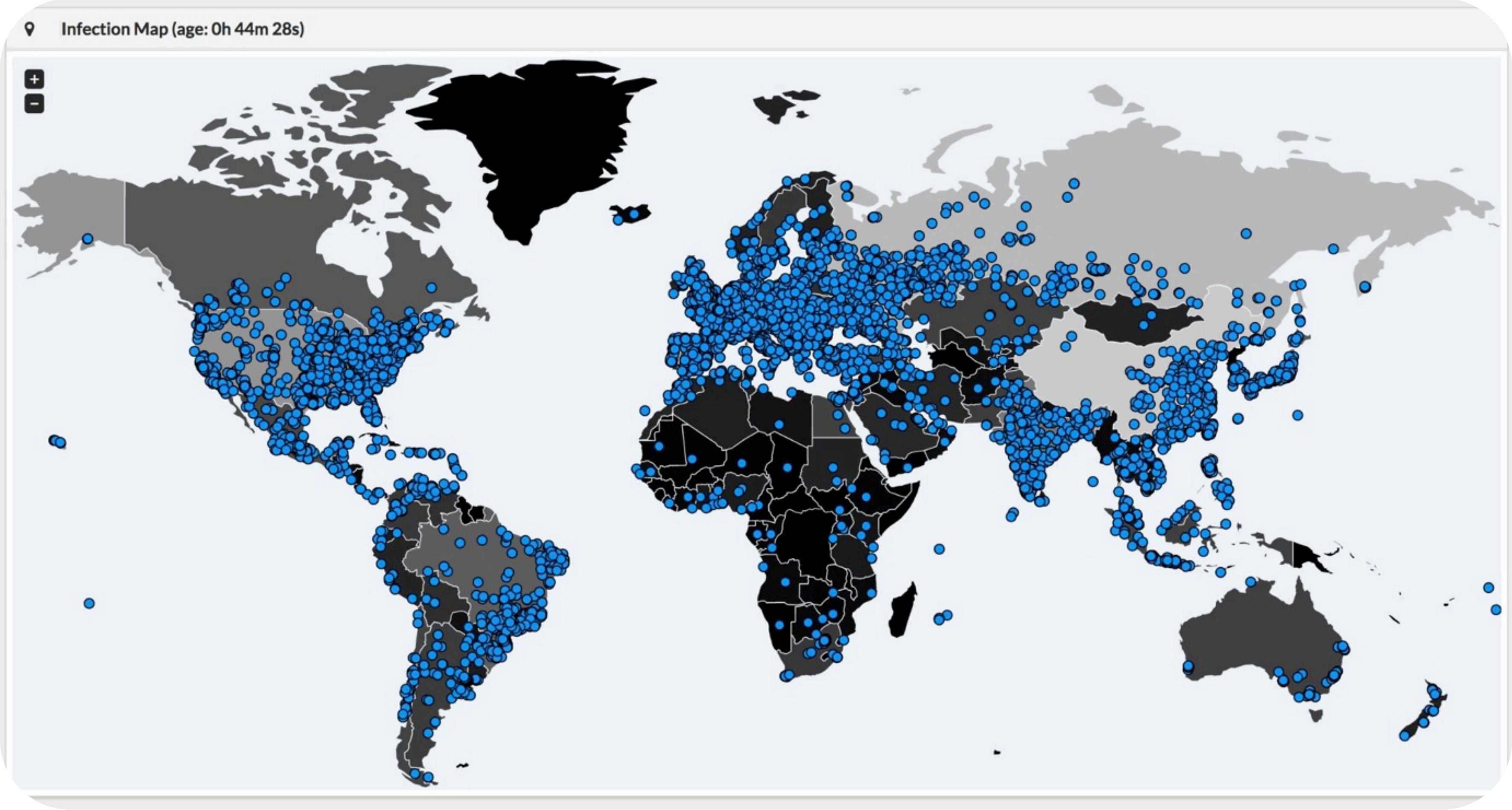
“Heartbleed” in OpenSSL library

High-impact Vulnerabilities

“Heartbleed” in OpenSSL library

Accounted for almost ***66%*** of active websites on the Internet.

Spreading of WannaCry



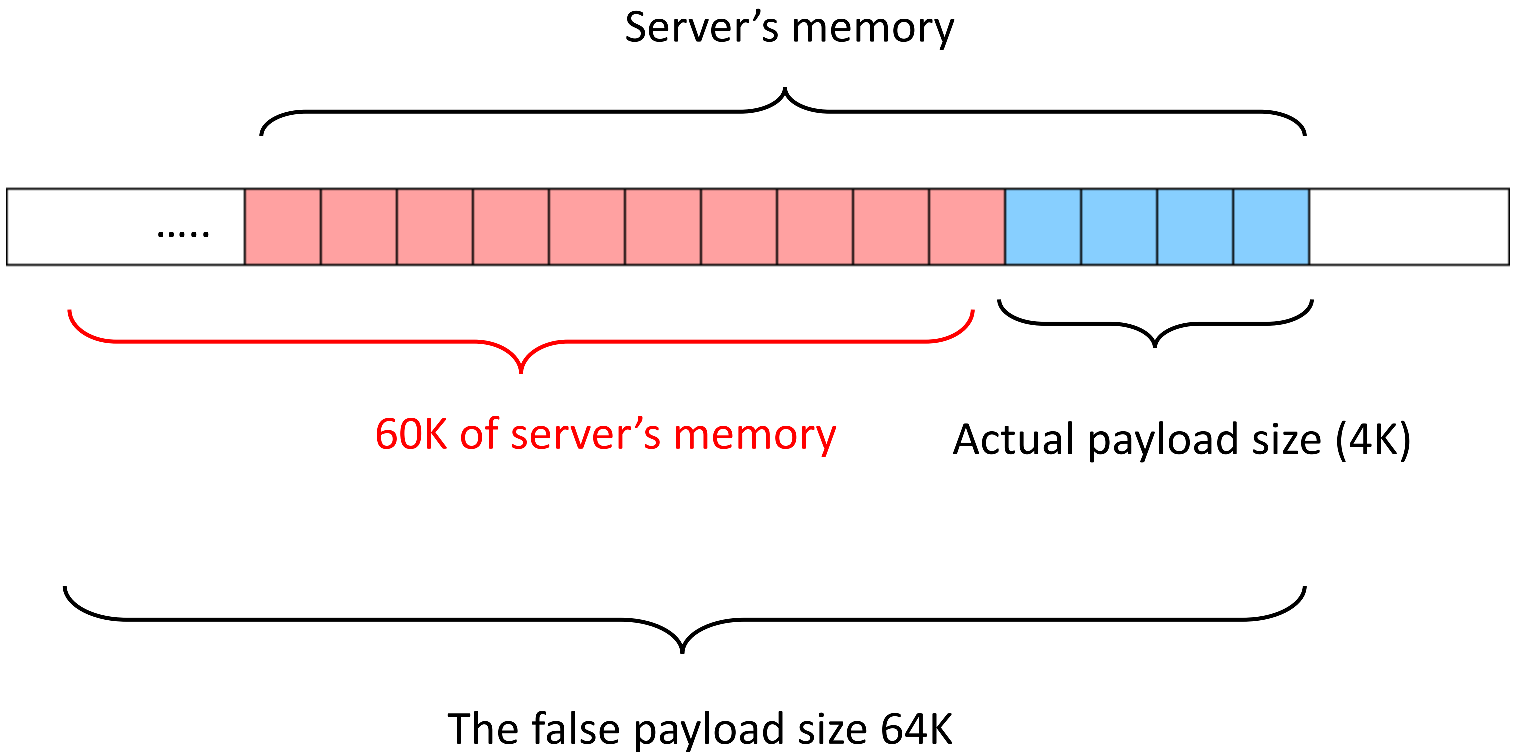
SWINBURNE UNIVERSITY OF TECHNOLOGY

A Realistic Example – Heartbleed Vulnerability

```c
/* ssl/d1_both.c */
// [...]
int dtls1_process_heartbeat(SSL *s)
{
   unsigned char *p =
    &s->s3->rrec.data[0], *pl;
   unsigned short hbtype;
   unsigned int payload;
   unsigned int padding = 16;
   hbtype = *p++;
   n2s(p, payload);
   pl = p;
   // [...]
   if (hbtype == TLS1_HB_REQUEST){
      unsigned char *buffer, *bp;
      int r;
      // [...]
      buffer = OPENSSL_malloc(1 + 2 +
    payload + padding);
      bp = buffer;
      *bp++ = TLS1_HB_RESPONSE;
      s2n(payload, bp);
      memcpy(bp, pl, payload);
      bp += payload;
      RAND_pseudo_bytes(bp, padding);
      r = dtls1_write_bytes(s,
    TLS1_RT_HEARTBEAT,buffer, 3 + payload
    + padding);
      // [...]
   }
   // [...]
   return 0;
}
```

The "Heartbleed" vulnerability in OpenSSL

There is no validation of the size of the variable **payload**



A Realistic Example – Heartbleed Vulnerability

```c
/* ssl/d1_both.c */
// [...]
int dtls1_process_heartbeat(SSL *s)
{
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */
  /* Read type and payload length first */
  hbtype = *p++;
  n2s(p, payload);
  if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0; /* silently discard per RFC 6520 sec.4*/
  pl = p;
  // [...]
  if (hbtype == TLS1_HB_REQUEST){
   unsigned char *buffer, *bp;
   int r;
   // [...]
   buffer = OPENSSL_malloc(1 + 2 + payload + padding);
   bp = buffer;
   /* Enter response type, length and copy payload */
   *bp++ = TLS1_HB_RESPONSE;
   s2n(payload, bp);
   memcpy(bp, pl, payload);
   bp += payload;
   /* Random padding */
   RAND_pseudo_bytes(bp, padding);
   r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT,buffer,
                         3 + payload + padding);
   // [...]
   if (r < 0) return r;
  }
  // [...]
  return 0;
}
```

Fig. 1: The "Heartbleed" vulnerability in OpenSSL.

The integer ***payload*** is defined by the macro ***n2s*** that reads a sixteen bit integer from a network stream.

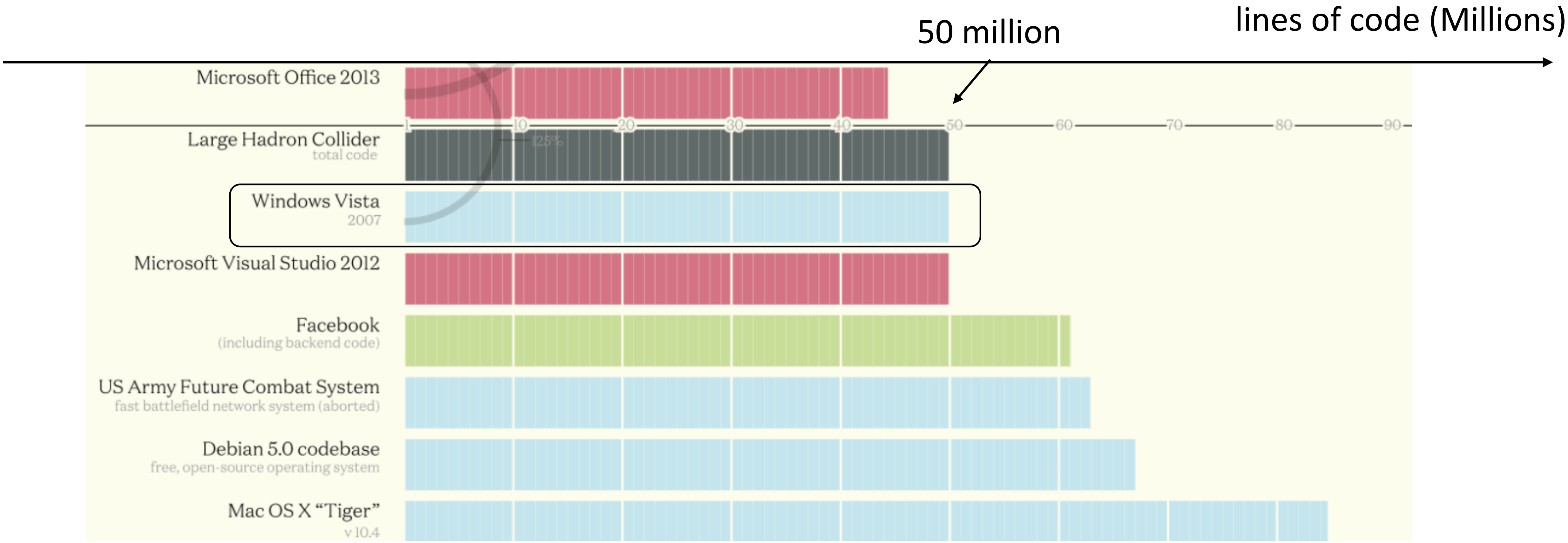
Hence, there is a need to inspect:

1) How information propagates from one statement to another

2) How the information flow is controlled by conditions

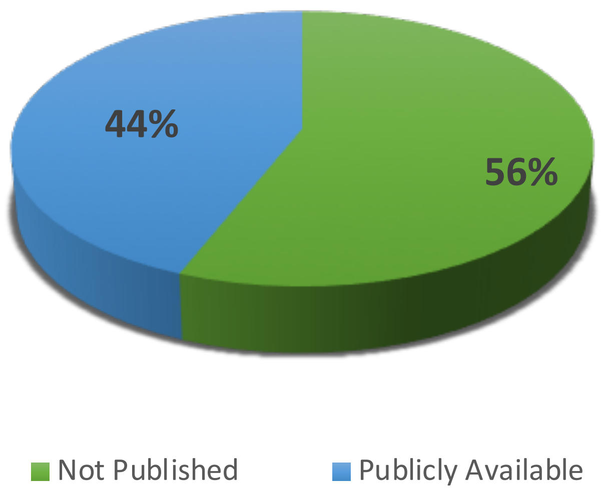
Challenges

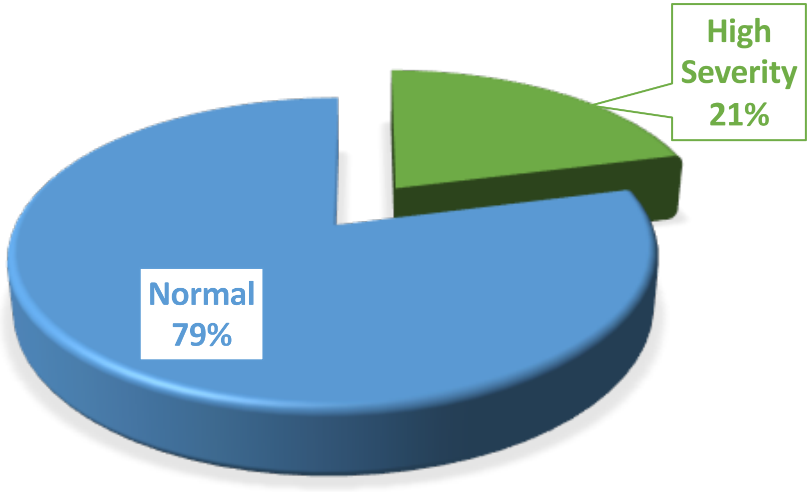
1. The growing complexity of software



Challenges

2. Vulnerabilities are plenty
   - Over 15,000 vulnerabilities reported in 2016 (over 280/week)



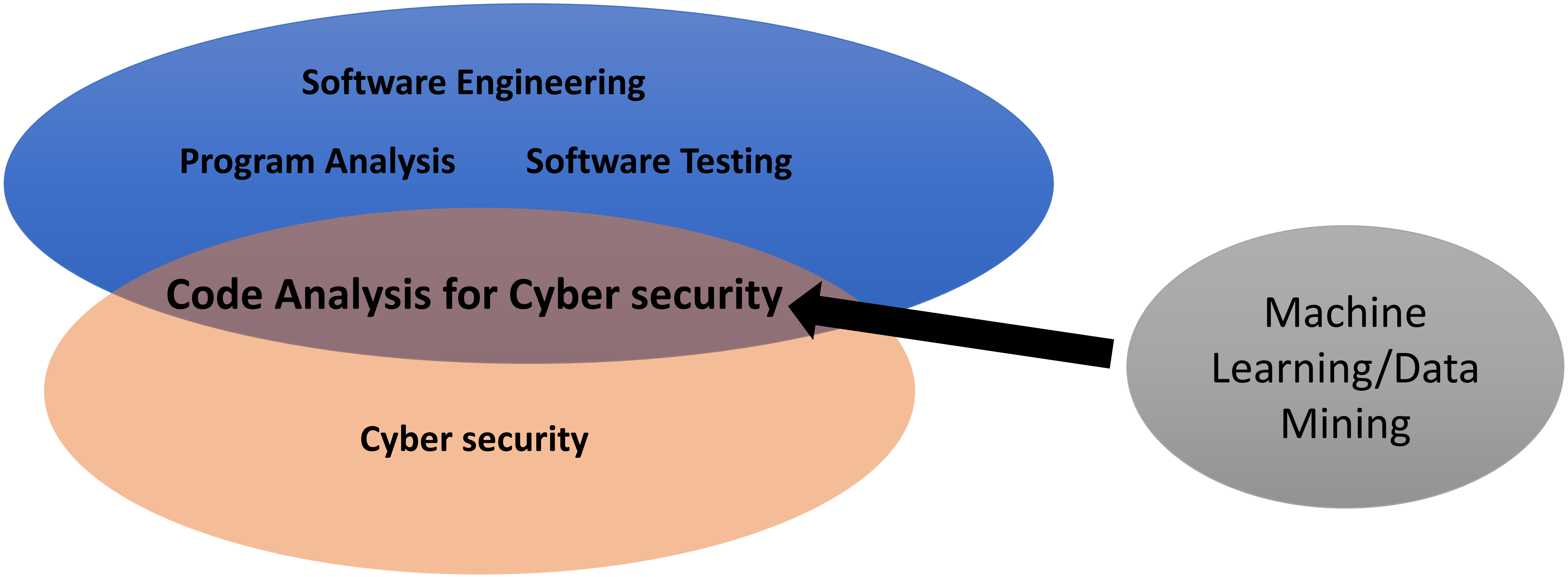


SWINBURNE UNIVERSITY OF TECHNOLOGY

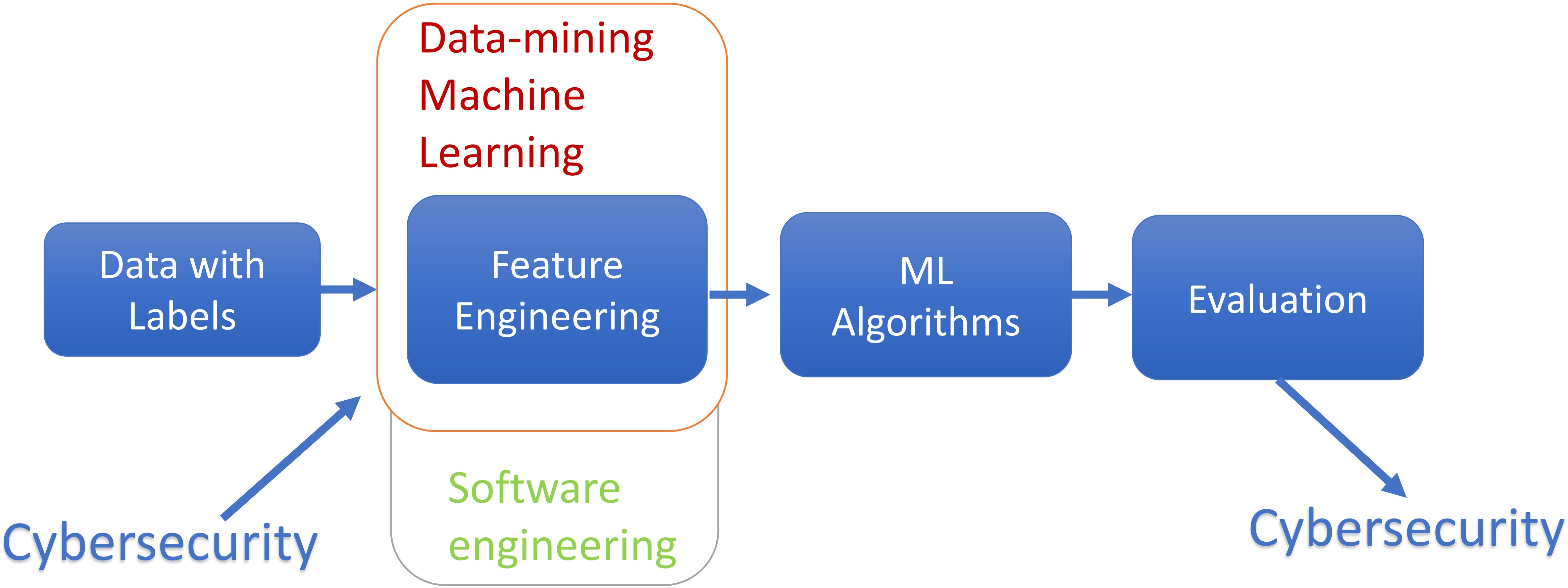
Challenges

3. Vulnerabilities are difficult to detect
    - Tedious and time-consuming
    - Sufficient understandings of projects and knowledge of security required
    - Security considerations are not prioritized and well-recognized

Scope


Software Engineering
Program Analysis
Software Testing
Code Analysis for Cyber security
Cyber security
Machine Learning/Data Mining

Machine Learning's Perspective



Feature Engineering

How do we convert **vulnerable code** to **effective features**?

Source code

```
/* ssl/d1_both.c */
// [...]
int dtls1_process_heartbeat(SSL *s)
{
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */
  /* Read type and payload length first */
  hbtype = *p++;
  n2s(p, payload);
  if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0; /* silently discard per RFC 6520 sec.4*/
  pl = p;
  // [...]
  if (hbtype == TLS1_HB_REQUEST){
   unsigned char *buffer, *bp;
   int r;
   // [...]
   buffer = OPENSSL_malloc(1 + 2 + payload + padding);
   bp = buffer;
   /* Enter response type, length and copy payload */
   *bp++ = TLS1_HB_RESPONSE;
   s2n(payload, bp);
   memcpy(bp, pl, payload);
   bp += payload;
   /* Random padding */
   RAND_pseudo_bytes(bp, padding);
   r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT,buffer,
                         3 + payload + padding);
   // [...]
   if (r < 0) return r;
  }
  // [...]
  return 0;
}
```

Pre-process

Features that are:

- Representative
- Invariant
- Distinctive

Classifiers

Feature Engineering

How do we convert **vulnerable code** to **effective features**?

Binary

```
94 06 4A 4A 18 4F 9A D3 AF 03 67 69 91 3D 24 88  ..JJ.O....gi.=$.
93 9D 5B C6 DD F0 72 01 43 6F 5F DC 9F 17 17 A0  ..[...r.Co_.....
84 87 27 1C 43 45 1E 92 12 DE 0E 3E 69 6C 21 9B  ..'.CE.....>il!.
C6 3C F8 60 5A 38 A1 C8 20 9C EF DA B8 A9 95 F9  .<.`Z8.. .......
CA AC 57 1C 4C 27 7B 17 FF B7 C1 98 61 7D E0 E5  ..W.L'{.....a}..
12 0A DE 53 3D C7 6C DE 1B DF 8F 22 68 77 EC 5F  ...S=.l...."hw._
57 98 69 5A 70 0E B6 06 CD 32 20 93 82 32 25 D1  W.iZp....2 ..2%.
7D 58 94 B0 CD 6D 15 37 A2 B9 F5 85 98 D9 65 F7  }X...m.7......e.
59 14 B4 93 8C 41 AD 3E 59 D5 26 53 C0 C5 8D 72  Y....A.>Y.&S...r
08 5A B4 03 19 17 A0 70 BE BE 0C 87 87 55 7B 04  .Z.....p.....U{.
57 2F 74 7F B5 54 73 8C 9F F8 6E DE 0B 28 36 18  W/t..Ts...n..(6.
81 73 3B C9 FF D0 8D 7A FD E8 3D 5B 7C 03 96 EA  .s;....z..=[|...
3D 31 51 D9 FD 46 1C F0 9A 3F C8 3C 18 7E 07 02  =1Q..F...?.<.~..
29 23 97 EE F8 64 58 3E 80 EB 84 99 82 3E 92 F5  )#...dX>.....>..
94 EB F0 6B 0A C8 CF D2 71 A2 27 41 73 0C 71 74  ...k....q.'As.qt
39 7C 59 DB A8 28 1B 3F D6 21 10 6A 68 4C 2A 05  9|Y..(.?.!.jhL*.
```

→ Pre-process →

Features that are:

- Representative
- Invariant
- Distinctive

→ Classifiers

Feature Engineering

How do we convert **vulnerable code** to **effective features**?

Binary

```
94 06 4A 4A 18 4F 9A D3 AF 03 67 69 91 3D 24 88  ..JJ.O....gi.=$.
93 9D 5B C6 DD F0 72 01 43 6F 5F DC 9F 17 17 A0  ..[...r.Co_.....
84 87 27 1C 43 45 1E 92 12 DE 0E 3E 69 6C 21 9B  ..'.CE.....>il!.
C6 3C F8 60 5A 38 A1 C8 20 9C EF DA B8 A9 95 F9  .<.`Z8.. .......
CA AC 57 1C 4C 27 7B 17 FF B7 C1 98 61 7D E0 E5  ..W.L'{.....a}..
12 0A DE 53 3D C7 6C DE 1B DF 8F 22 68 77 EC 5F  ...S=.l...."hw._
57 98 69 5A 70 0E B6 06 CD 32 20 93 82 32 25 D1  W.iZp....2 ..2%.
7D 58 94 B0 CD 6D 15 37 A2 B9 F5 85 98 D9 65 F7  }X...m.7......e.
59 14 B4 93 8C 41 AD 3E 59 D5 26 53 C0 C5 8D 72  Y....A.>Y.&S...r
08 5A B4 03 19 17 A0 70 BE BE 0C 87 87 55 7B 04  .Z.....p.....U{.
57 2F 74 7F B5 54 73 8C 9F F8 6E DE 0B 28 36 18  W/t..Ts...n..(6.
81 73 3B C9 FF D0 8D 7A FD E8 3D 5B 7C 03 96 EA  .s;....z..=[|...
3D 31 51 D9 FD 46 1C F0 9A 3F C8 3C 18 7E 07 02  =1Q..F...?.<.~..
29 23 97 EE F8 64 58 3E 80 EB 84 99 82 3E 92 F5  )#...dX>......>..
94 EB F0 6B 0A C8 CF D2 71 A2 27 41 73 0C 71 74  ...k....q.'As.qt
39 7C 59 DB A8 28 1B 3F D6 21 10 6A 68 4C 2A 05  9|Y..(.?.!.jhL*.
```

```
C01E 8D F0    INHEX   BSR     INCH      GET A CHAR
C020 81 30            CMP A   #'0       ZERO
C022 2B 11            BMI     HEXERR    NOT HEX
C024 81 39            CMP A   #'9       NINE
C026 2F 0A            BLE     HEXRTS    GOOD HEX
C028 81 41            CMP A   #'A
C02A 2B 09            BMI     HEXERR    NOT HEX
```

Pre-process

Features that are:

- Representative
- Invariant
- Distinctive

Classifiers

Feature Engineering

How do we convert **vulnerable code** to **effective features**?

Source code

```
/* ssl/d1_both.c */
// [...]
int dtls1_process_heartbeat(SSL *s)
{
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */
  /* Read type and payload length first */
  hbtype = *p++;
```

Binary

```
94 06 4A 4A 18 4F 9A D3 AF 03 67 69 91 3D 24 88  ..JJ.O....gi.=$.
93 9D 5B C6 DD F0 72 01 43 6F 5F DC 9F 17 17 A0  ..[...r.Co_.....
84 87 27 1C 43 45 1E 92 12 DE 0E 3E 69 6C 21 9B  ..'.CE.....>il!.
C6 3C F8 60 5A 38 A1 C8 20 9C EF DA B8 A9 95 F9  .<.`Z8.. .......
CA AC 57 1C 4C 27 7B 17 FF B7 C1 98 61 7D E0 E5  ..W.L'{.....a}..
12 0A DE 53 3D C7 6C DE 1B DF 8F 22 68 77 EC 5F  ...S=.l...."hw._
57 98 69 5A 70 0E B6 06 CD 32 20 93 82 32 25 D1  W.iZp....2 ..2%.
7D 58 94 B0 CD 6D 15 37 A2 B9 F5 85 98 D9 65 F7  }X...m.7......e.
... 26 53 C0 C5 8D 72  Y....A.>Y.&S...r
... 87 87 55 7B 04  .Z.....p.....U{.
```

Pre-process

"Patterns"

"Text"

"Code Properties"

Features

Feature Engineering

How to convert **vulnerable code** to **effective features**?

"Patterns"

- Trees → Abstract Syntax Trees
- Graphs
  - Function Call Graphs
  - Data Flow Graphs
  - Control Flow Graphs
  - Dependency Graphs
  - ....
- ....

SWINBURNE UNIVERSITY OF TECHNOLOGY

Feature Engineering

How to convert **vulnerable code** to **effective features**?

Source code

```
/* ssl/d1_both.c */
// [...]
int dtls1_process_heartbeat(SSL *s)
{
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */
  /* Read type and payload length first */
  hbtype = *p++;
```

Binary

```
94 06 4A 4A 18 4F 9A D3 AF 03 67 69 91 3D 24 88  ..JJ.O....gi.=$.
93 9D 5B C6 DD F0 72 01 43 6F 5F DC 9F 17 17 A0  ..[...r.Co_.....
84 87 27 1C 43 45 1E 92 12 DE 0E 3E 69 6C 21 9B  ..'.CE.....>il!.
C6 3C F8 60 5A 38 A1 C8 20 9C EF DA B8 A9 95 F9  .<.`Z8.. .......
CA AC 57 1C 4C 27 7B 17 FF B7 C1 98 61 7D E0 E5  ..W.L'{.....a}..
12 0A DE 53 3D C7 6C DE 1B DF 8F 22 68 77 EC 5F  ...S=.l...."hw._
57 98 69 5A 70 0E B6 06 CD 32 20 93 82 32 25 D1  W.iZp....2 ..2%.
7D 58 94 B0 CD 6D 15 37 A2 B9 F5 85 98 D9 65 F7  }X...m.7......e.
                           D5 26 53 C0 C5 8D 72  Y....A.>Y.&S...r
                            E 0C 87 87 55 7B 04  .Z.....p.....U{.
```

Pattern recognition problems

Patterns

Features that are:

- Representative
- Invariance
- Distinctive

SWINBURNE UNIVERSITY OF TECHNOLOGY

Feature Engineering

How to convert **vulnerable code** to **effective features**?

Source code

```c
/* ssl/d1_both.c */
// [...]
int dtls1_process_heartbeat(SSL *s)
{
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */
  /* Read type and payload length first */
  hbtype = *p++;
```

Binary

```
94 06 4A 4A 18 4F 9A D3 AF 03 67 69 91 3D 24 88  ..JJ.O....gi.=$.
93 9D 5B C6 DD F0 72 01 43 6F 5F DC 9F 17 17 A0  ..[...r.Co_.....
84 87 27 1C 43 45 1E 92 12 DE 0E 3E 69 6C 21 9B  ..'.CE.....>il!.
C6 3C F8 60 5A 38 A1 C8 20 9C EF DA B8 A9 95 F9  .<.`Z8.. .......
CA AC 57 1C 4C 27 7B 17 FF B7 C1 98 61 7D E0 E5  ..W.L'{.....a}..
12 0A DE 53 3D C7 6C DE 1B DF 8F 22 68 77 EC 5F  ...S=.l...."hw._
57 98 69 5A 70 0E B6 06 CD 32 20 93 82 32 25 D1  W.iZp....2 ..2%.
7D 58 94 B0 CD 6D 15 37 A2 B9 F5 85 98 D9 65 F7  }X...m.7......e.
                           D5 26 53 C0 C5 8D 72  Y....A.>Y.&S...r
                            E 0C 87 87 55 7B 04  .Z.....p.....U{.
```

Natural Language Processing (NLP) Problem

Text

Features that are:

- Representative
- Invariance
- Distinctive

Feature Engineering

How to convert **vulnerable code** to **effective features?**

Source code

```
/* ssl/d1_both.c */
// [...]
int dtls1_process_heartbeat(SSL *s)
{
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */
  /* Read type and payload length first */
  hbtype = *p++;
  n2s(p, payload);
  if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0; /* silently discard per RFC 6520 sec.4*/
  pl = p;
  // [...]
  if (hbtype == TLS1_HB_REQUEST){
   unsigned char *buffer, *bp;
   int r;
   // [...]
   buffer = OPENSSL_malloc(1 + 2 + payload + padding);
   bp = buffer;
   /* Enter response type, length and copy payload */
   *bp++ = TLS1_HB_RESPONSE;
   s2n(payload, bp);
   memcpy(bp, pl, payload);
   bp += payload;
   /* Random padding */
   RAND_pseudo_bytes(bp, padding);
   r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT,buffer,
                         3 + payload + padding);
   // [...]
   if (r < 0) return r;
  }
  // [...]
  return 0;
}
```

Code Properties

Count Metrics	Blank Lines of Code
	Lines of Code
	Lines with Comments
	Statements
	Physical Lines of Code
	Declarative Lines of Code
	Executable Lines of Code
	Lines with Comments
	Inactive Lines
	Preprocessor Lines
	FanIn [11]
	FanOut [11]
	Path
Complexity Metrics	Cyclomatic Complexity (CC) [14]
	Modified CC [14]
	Strict CC [14]
	Essential Complexity [14]
	Knots [36]
	Nesting [10]

Feature Engineering

How to convert **vulnerable code** to **effective features**?

```
/* ssl/d1_both.c */
// [...]
int dtls1_process_heartbeat(SSL *s)
{
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */
  /* Read type and payload length first */
  hbtype = *p++;
  n2s(p, payload);
  if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0; /* silently discard per RFC 6520 sec.4*/
  pl = p;
  // [...]
  if (hbtype == TLS1_HB_REQUEST){
   unsigned char *buffer, *bp;
   int r;
   // [...]
   buffer = OPENSSL_malloc(1 + 2 + payload + padding);
   bp = buffer;
   /* Enter response type, length and copy payload */
   *bp++ = TLS1_HB_RESPONSE;
   s2n(payload, bp);
   memcpy(bp, pl, payload);
   bp += payload;
   /* Random padding */
   RAND_pseudo_bytes(bp, padding);
   r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT,buffer,
                      3 + payload + padding);
   // [...]
   if (r < 0) return r;
  }
  // [...]
  return 0;
```

Written by developers

Developer relationships

Developer & file relationships

Graphical models

SWINBURNE UNIVERSITY OF TECHNOLOGY

Machine Learning Algorithms

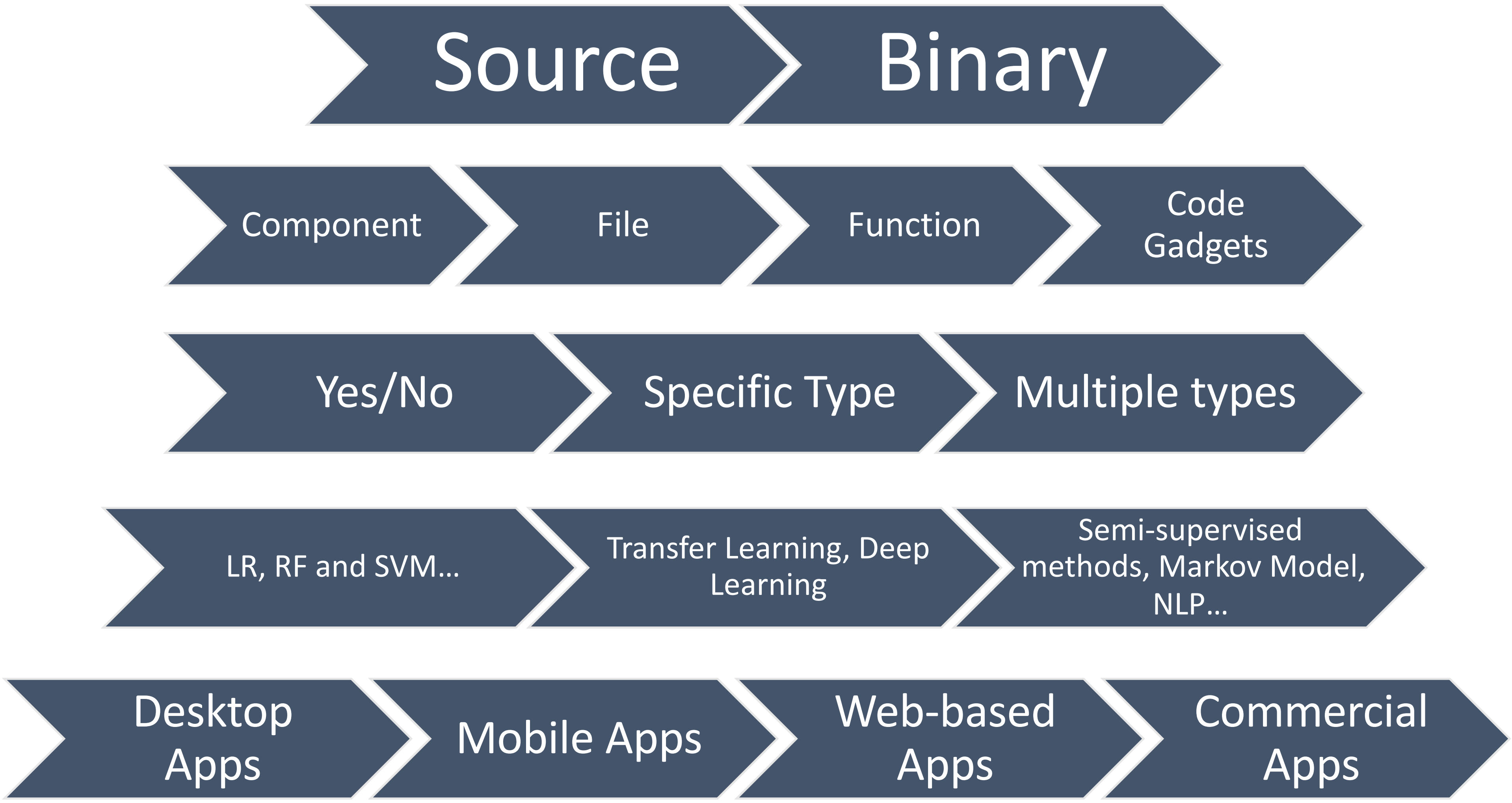
Data with Labels → Feature Engineering → ML Algorithms → Evaluation

Research Trends

- LR, RF and SVM…
- Transfer Learning, Deep Learning
- Semi-supervised methods, Markov Models, NLP…

SWINBURNE UNIVERSITY OF TECHNOLOGY

Research Trends

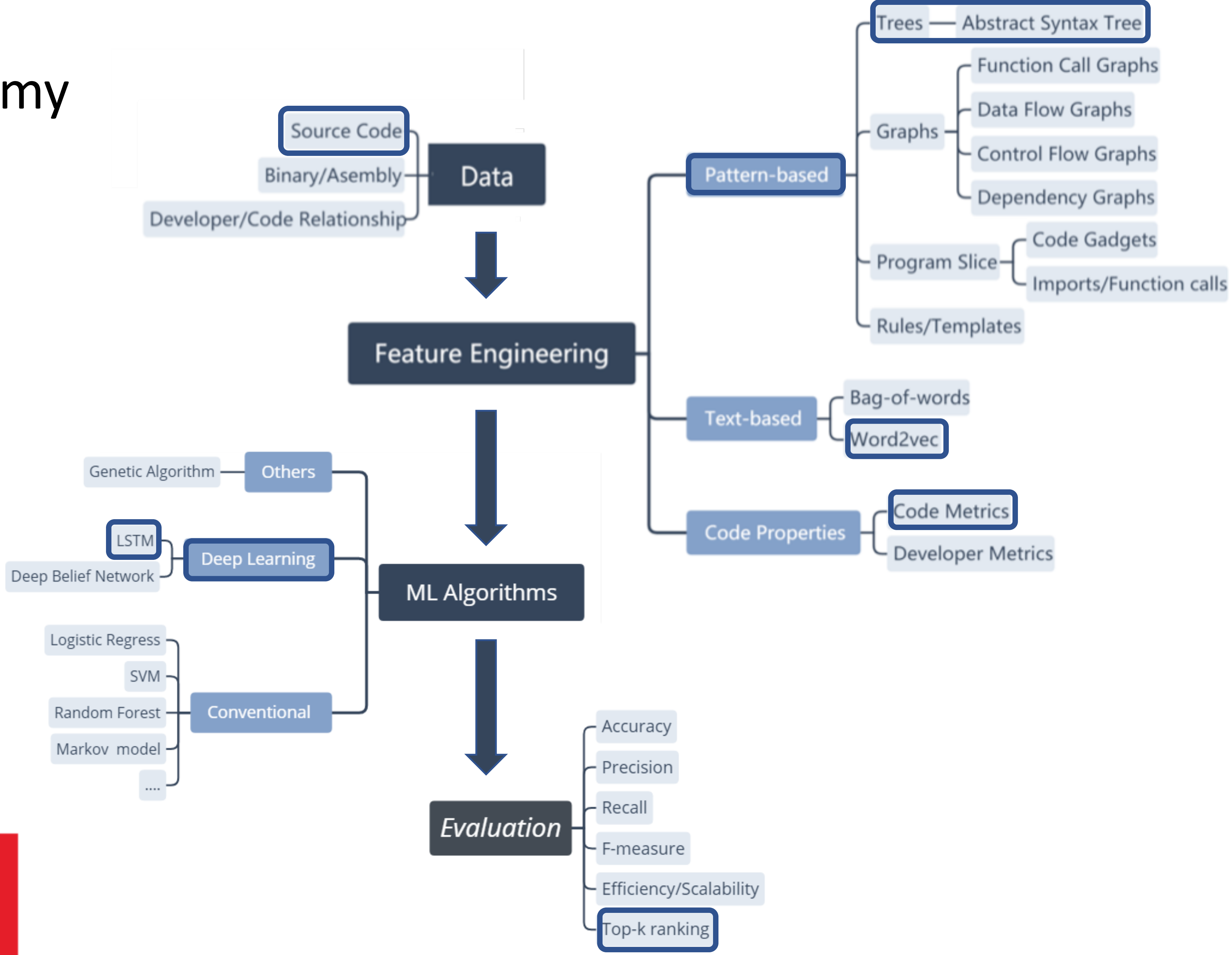


Our work

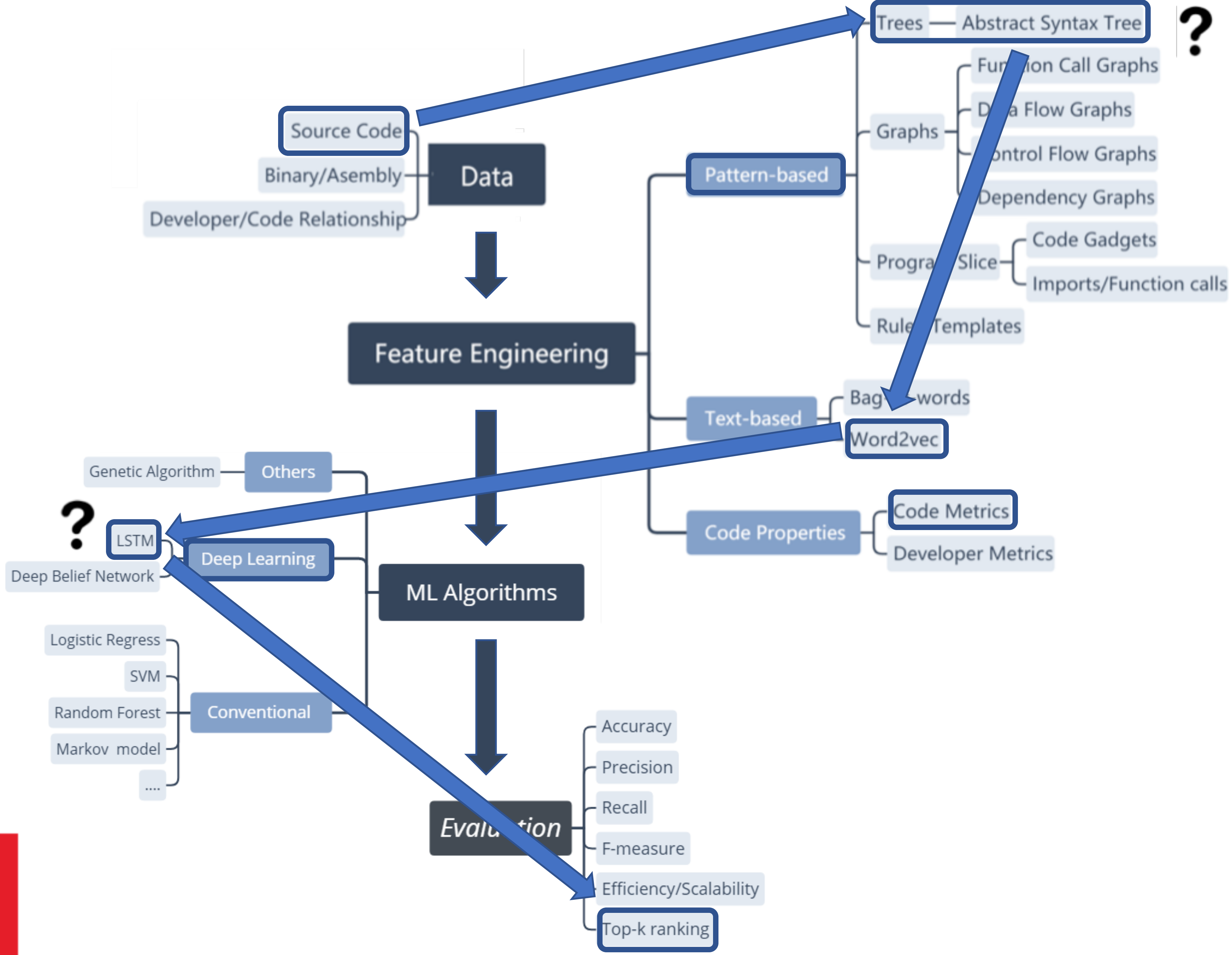
- Vulnerability Discovery with Function Representation Learning from Unlabeled Projects (accepted by CCS2017 Poster)
  - ✓ Function-level detection
  - ✓ Cross-project scenario
  - ✓ Representation learning with deep learning approach

SWINBURNE UNIVERSITY OF TECHNOLOGY

Taxonomy



SWINBURNE UNIVERSITY OF TECHNOLOGY



SWINBURNE UNIVERSITY OF TECHNOLOGY

Key Assumptions

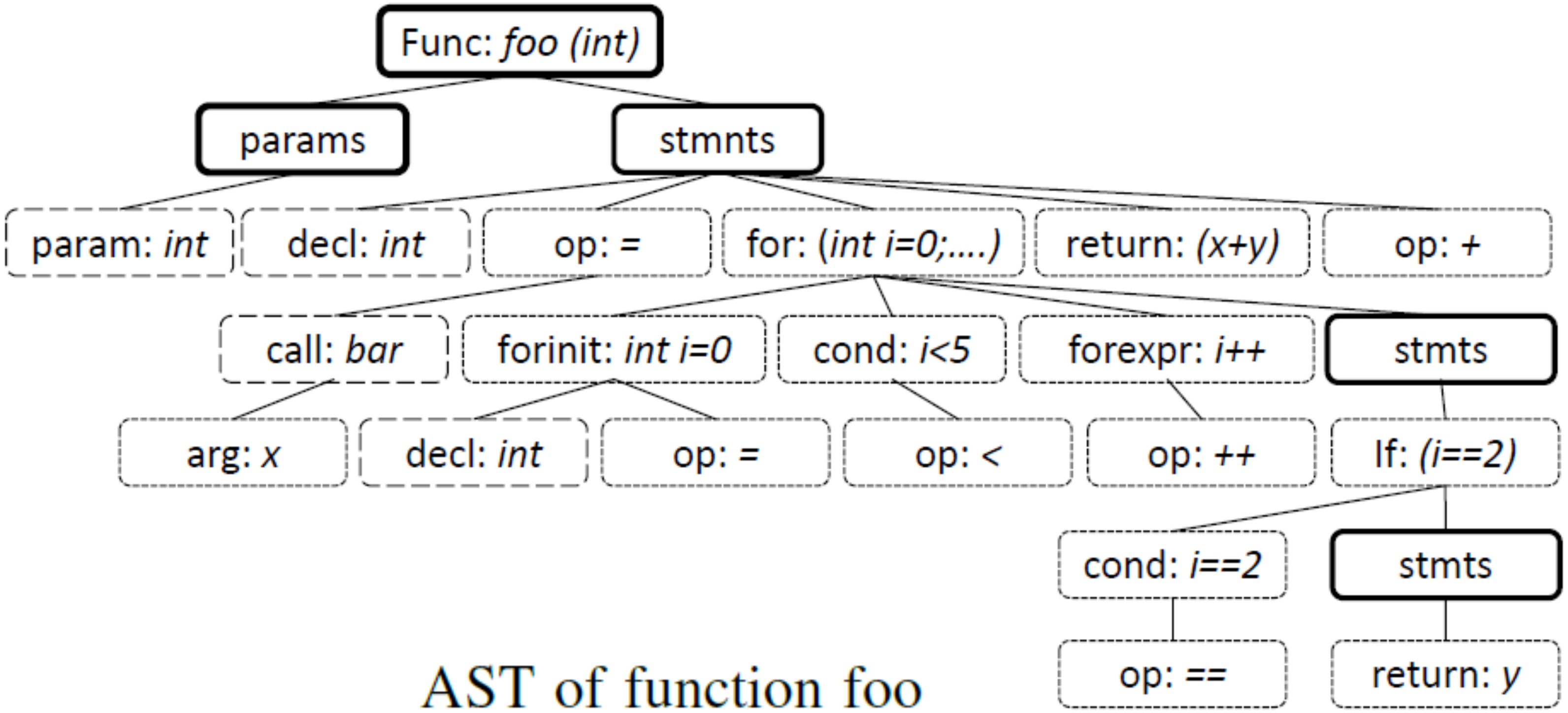
1. Vulnerable programming patterns are associated with many vulnerabilities, and these patterns can be revealed by analysing the program's ASTs

```
int foo(int x)
{
  int y = bar(x);

  for (int i = 0;
       i < 5;i++)
  {
    if (i == 2)
    return y;
  }

  return (x + y);
}
```

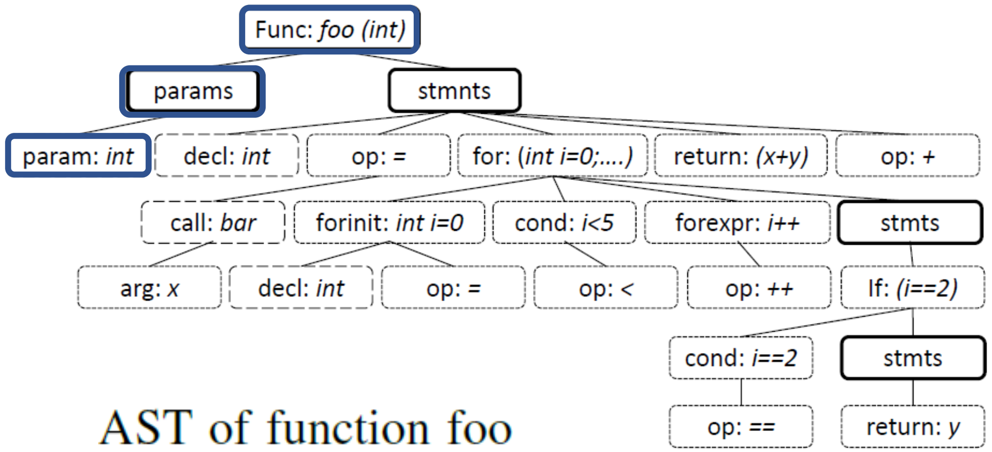
Function foo



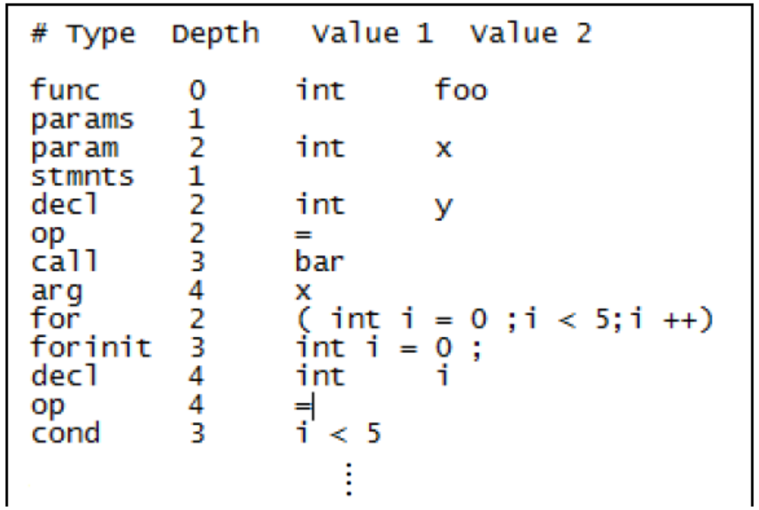
AST of function foo

Research Question

1. How do we covert ASTs to vectors acceptable by ML algorithms while preserving the **structural** information?



AST of function foo



# Type	Depth	Value 1	Value 2
func	0	int	foo
params	1		
param	2	int	x
stmnts	1		
decl	2	int	y
op	2	=	
call	3	bar	
arg	4	x	
for	2	(int i = 0 ;i < 5;i ++)	
forinit	3	int i = 0 ;	
decl	4	int	i
op	4	=	
cond	3	i < 5	
		⋮	

(c) A serialized AST of function foo

Depth-first Traversal

[*foo, int, params, param, int, x, stmnts, decl, int, y, op, =, call, bar, arg, x, for, int, i, … return , y*]

SWINBURNE UNIVERSITY OF TECHNOLOGY

Key Assumptions

2. The sequence of elements in the textual vectors **resembles** sequences of words in natural language.

[*foo, int, params, param, int, x, stmnts, decl, int, y, op, =, call, bar, arg, x, for, int, i, … return , y*]

AST: textual vector

Natural language: sentence.

[*Hi, everyone, my, name, is, Guanjun, Lin, I, am, from, China, …, I, can, speak, fluent,* Chinese *,thanks*]

SWINBURNE UNIVERSITY OF TECHNOLOGY

Research Question

2. How do we covert ASTs to vectors acceptable by ML algorithms while preserving the **syntactic & semantic** information?

Variable names

[*foo, int, params, param, int, x, stmnts, decl, int, y, op, =, call, bar, arg, x, for, int, i, … return , y*]

Function name

Data type

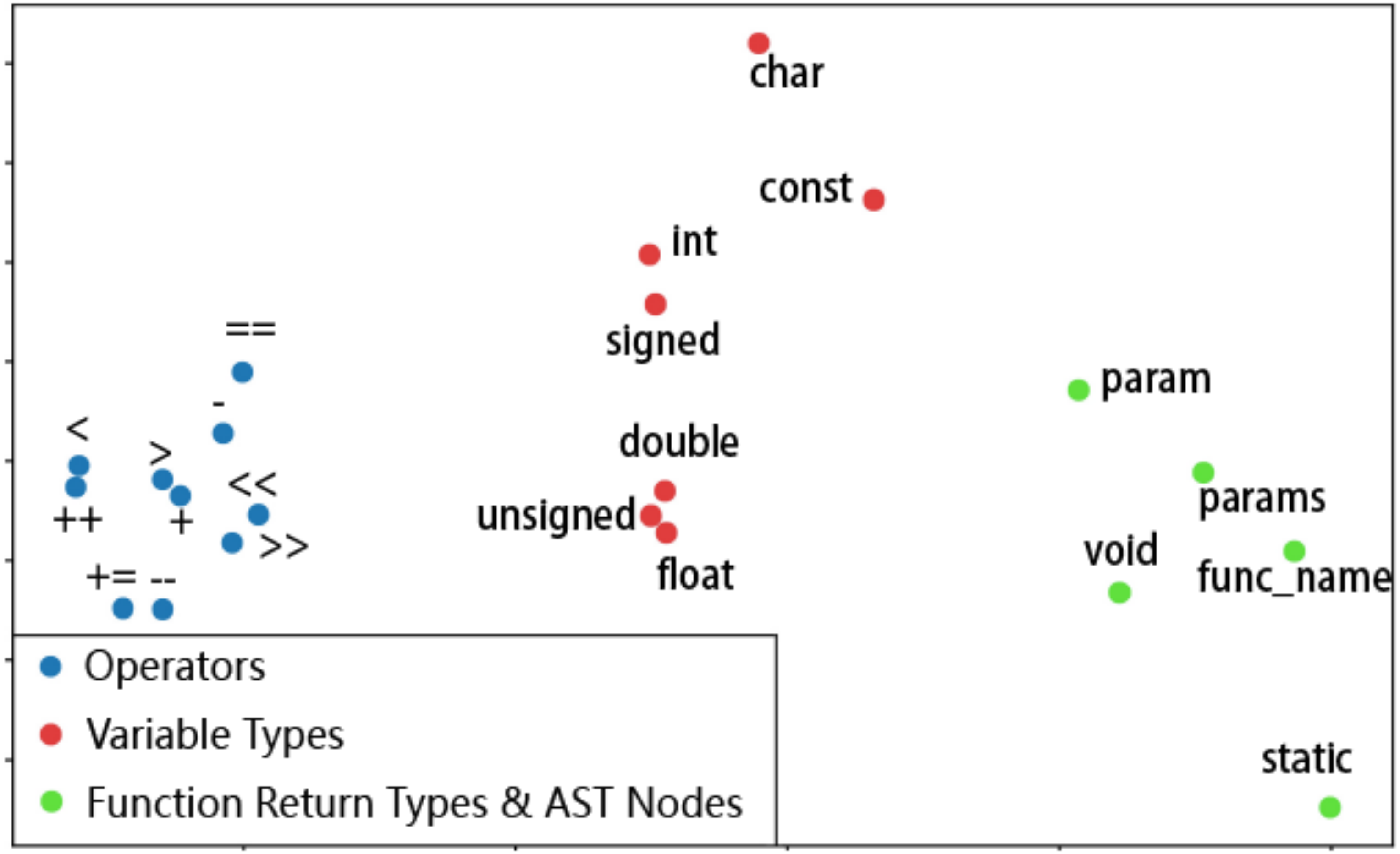
Variable declaration

Tokenization

[*25, 1, 7, 8, 16, 23, 45, 123, 569, 10, 1, 9, …., 20, 2*]

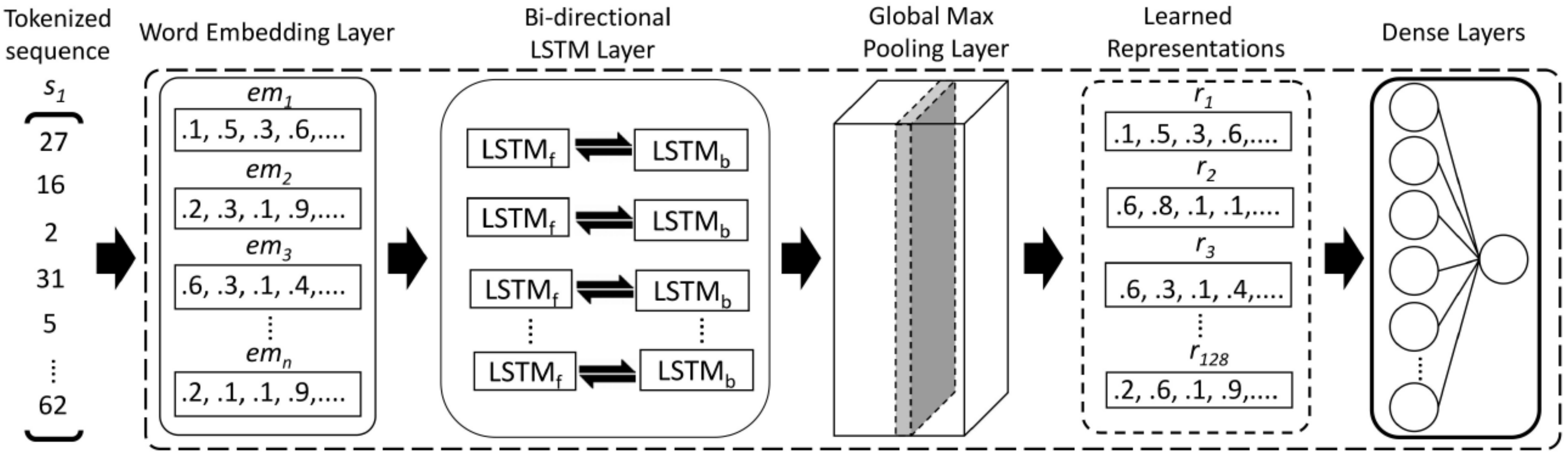
SWINBURNE UNIVERSITY OF TECHNOLOGY

Word2vec Embeddings



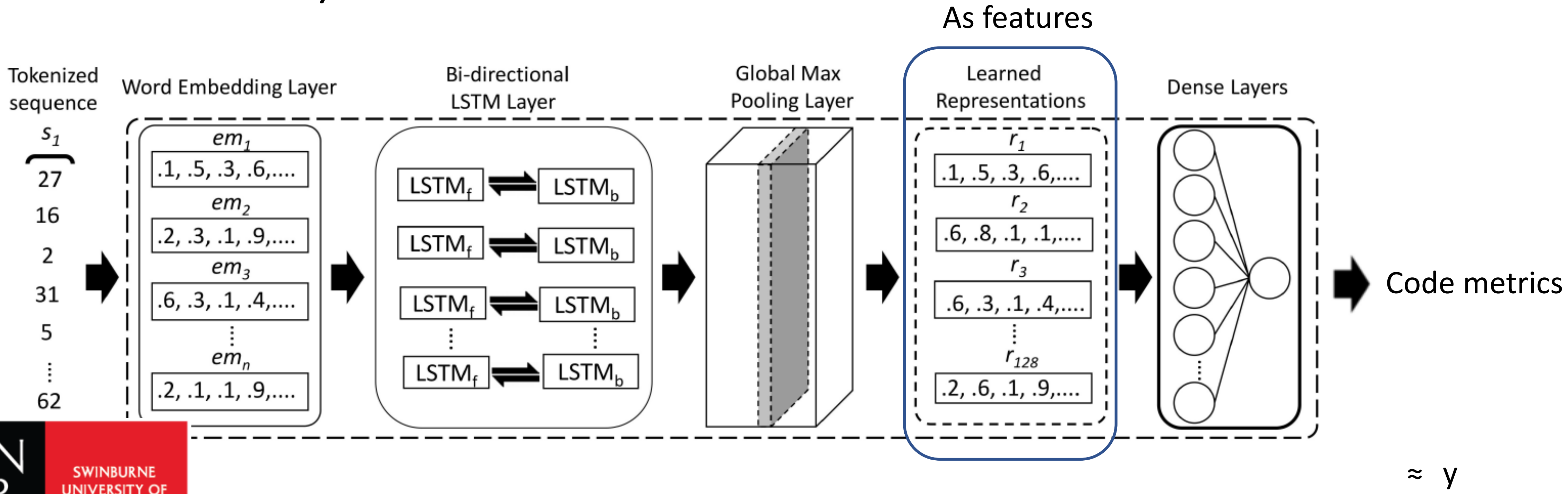
SWINBURNE UNIVERSITY OF TECHNOLOGY

Network Design



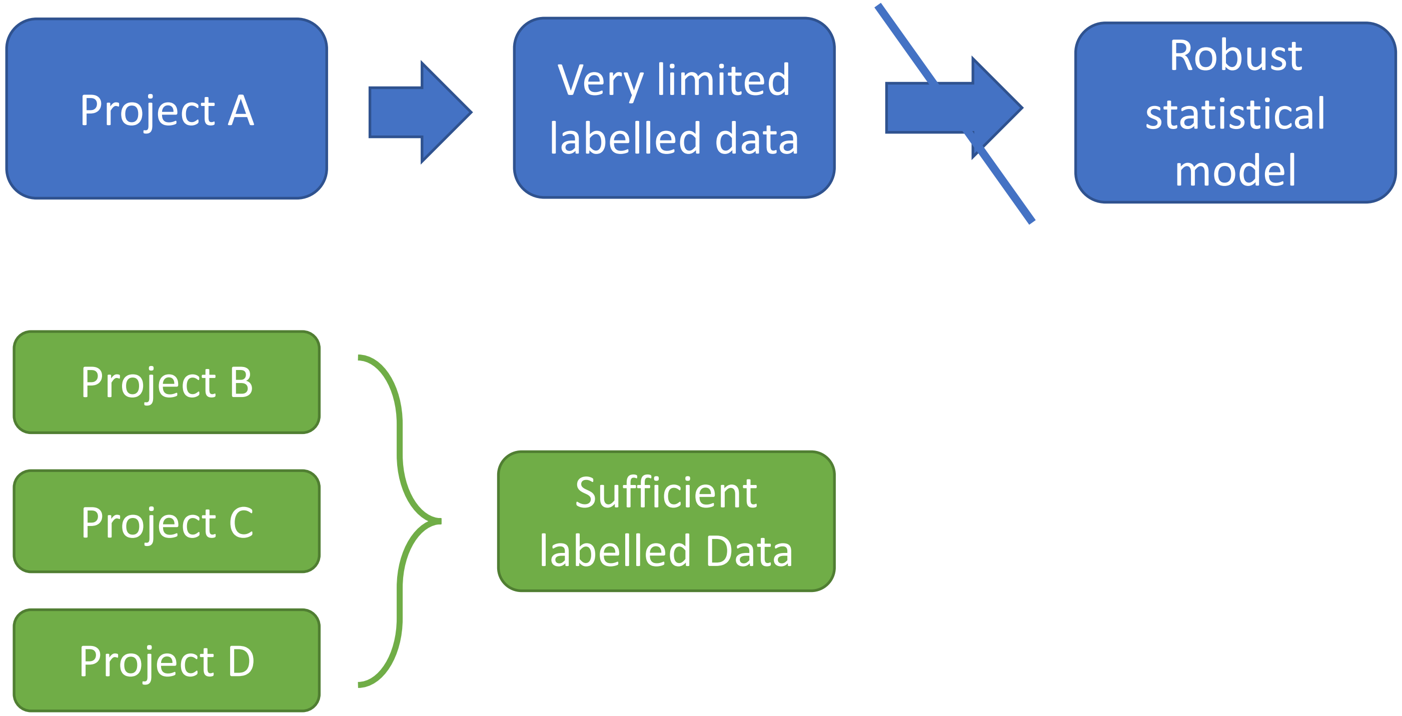
Our Work

1. Overcome the difficulty of obtaining manual labels by leveraging well-understood complexity metrics (used as a **proxy** as the substitute of data labels)



Our Work

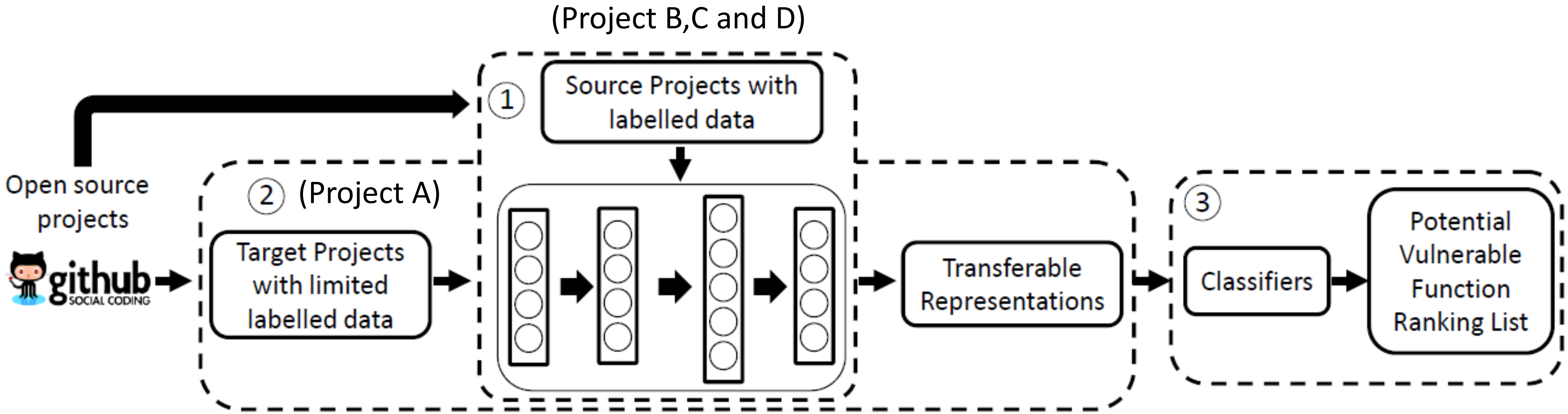
2. Overcome the insufficiency of vulnerable data of the inactive open source projects by leveraging transfer-learned representations.



SWINBURNE UNIVERSITY OF TECHNOLOGY

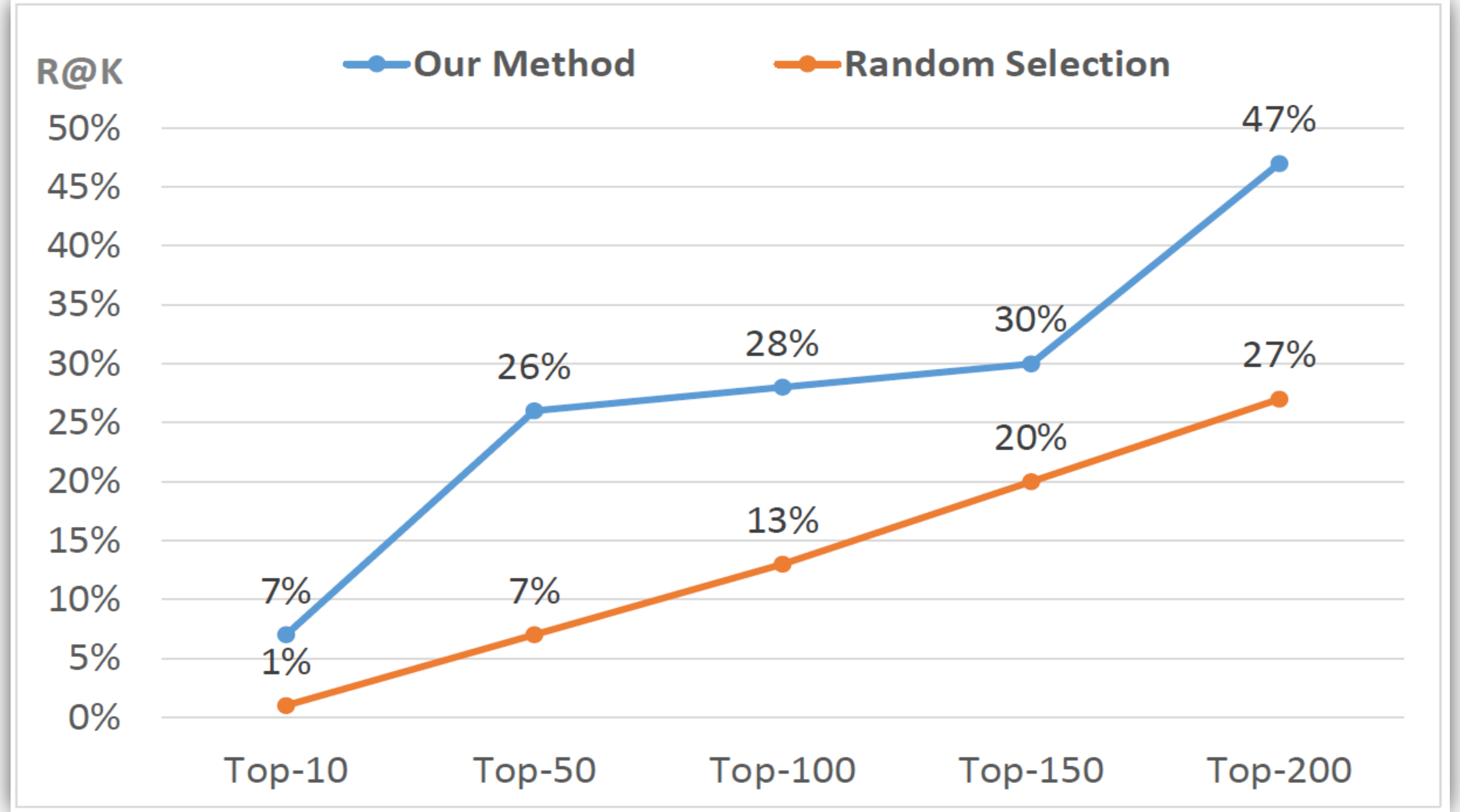
Our Work

2. Overcome the insufficiency of vulnerable data of the inactive open source projects by leveraging transfer-learned representations.



SWINBURNE UNIVERSITY OF TECHNOLOGY

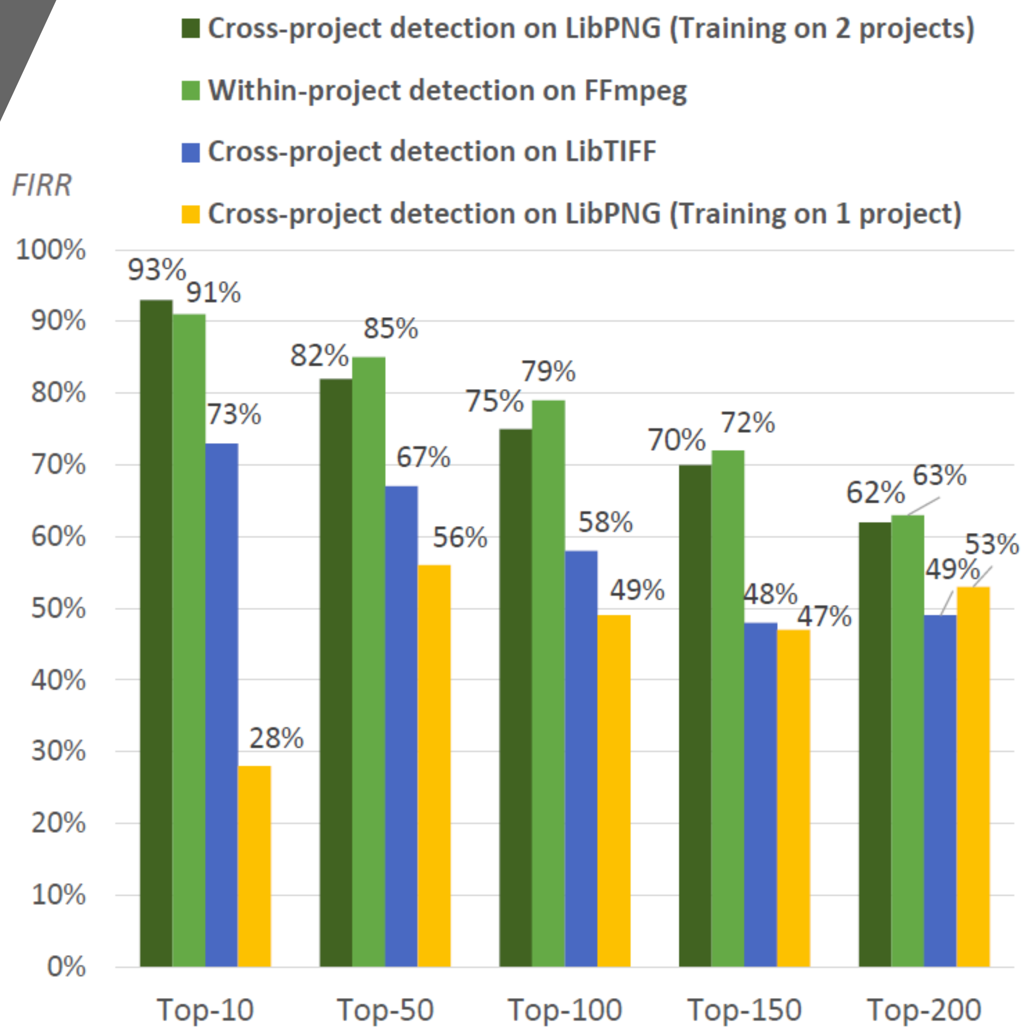
Results



SWINBURNE UNIVERSITY OF TECHNOLOGY

Efforts Saved

Human efforts saved for manually auditing potentially vulnerable functions with our method.



SWINBURNE UNIVERSITY OF TECHNOLOGY

Thank You – Q&A