Поиск уязвимостей в программном коде при помощи методов машинного обучения на примере парадокса Лебланка.

Легерова Валерия
Евгеньевна, МГУ 2024
germiona1404@yandex.ru

https://github.com/Wilgvalz/fittzel

1) Представление целых чисел в памяти компьютера

Существует несколько способов представления целых чисел в памяти компьютера.

(-) unsigned int a;

(-) a=0;

(-) a=a-1;

Typical Sizes and Ranges for Integer Types on 32-Bit Platforms

Type	Width (in Bits)	Minimum Value	Maximum Value
signed char	8	-128	127
unsigned char	8	0	255
short	16	-32,768	32,767
unsigned short	16	0	65,535
Int	32	-2,147,483,648	2,147,483,647
unsigned int	32	0	4,294,967,295
long	32	-2,147,483,648	2,147,483,647
unsigned long	32	0	4,294,967,295
long long	64	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long long	64	0	18,446,744,073,709,551,615

```
      1110 0000 0000 0000 0000 0000 0010 0000
+     0010 0000 0000 0000 0000 0000 0010 0000
=   1 0000 0000 0000 0000 0000 0000 0100 0000
```

2) ***Парадокс Лебланка***

Рассмотрим число (0x80000000)16 (или (-2147483648)10). Это число является нижней границей допустимых значений типа int. Попробуем получить из

него отрицательное, используя правила представления отрицательных чисел в дополнительном коде:

1000 0000 0000 0000 0000 0000 0000 0000 ---> 0111 1111 1111 1111 1111 1111 1111 1111 ---> 1000 0000 0000 0000 0000 0000 0000 0000

Число не изменилось!

Ошибка состоит в том, что происходит попытка записи значения по индексу большого по модулю отрицательного числа. Указатель как бы проскакивает массив и пытается записать какое-то значение в невыделенную для этого область памяти.

```c
#include <stdio.h>
int hashbank(int index_e, int value_e);
int hashbank(int index, int value) {
        int bank[4096];
        if (index < 0) {
                index = -index;
        }
        if (index >= 4096) return -1;

        bank[index] = value;
        return 0;
}
int main() {
        int a,b;
        a = 0x80000000;
        b = 0x414243;
        printf("%d", hashbank(a,b));
}
```

```
macpro:~ lera$ gcc -o leblanc_example leblanc_example.c -fsanitize=address
macpro:~ lera$ ./leblanc_example
80000000-2147483648,0
AddressSanitizer:DEADLYSIGNAL
=================================================================
==2661==ERROR: AddressSanitizer: SEGV on unknown address 0x7ffcea2e0880 (pc
 0x00010591bc34 bp 0x7ffeea2e4a40 sp 0x7ffeea2e0860 T0)
==2661==The signal is caused by a WRITE memory access.
    #0 0x10591bc33 in hashbank (leblanc_example:x86_64+0x100000c33)
    #1 0x10591bd14 in main (leblanc_example:x86_64+0x100000d14)
    #2 0x7fff598c43d4 in start (libdyld.dylib:x86_64+0x163d4)
```

- 3) **Semgrep**

```yaml
rules:
  - id: leblanc
    metadata:
      author: project Fitzel
      references:
        - https://github.com
      confidence: LOW
    message: >-
        Possible Leblancian paradox detected
    severity: INFO
    languages:
      - c
      - cpp
    pattern-either:
      - pattern: |
                if ($X < 0) {
                   $X = -$X;
                }
      - pattern: |
                if (<... $X < 0 ...>) {
                    ...
                    $X = -$X;
                    ...
                }
      - pattern: |
                if (<... $X <= 0 ...>) {
                    ...
                    $X = -$X;
                    ...
                }
      - pattern: |
                if ($Z < 0) $Z = -$Z;
      - pattern: |
                if ($Z <= 0) $Z = -$Z;
      - pattern: |
                if (<... $Z < 0 ...>) $Z = -$Z;
      - pattern: |
                $Y = $X < 0 ? -$X : $X;
      - pattern: |
                $Y = $X <= 0 ? -$X : $X;

```

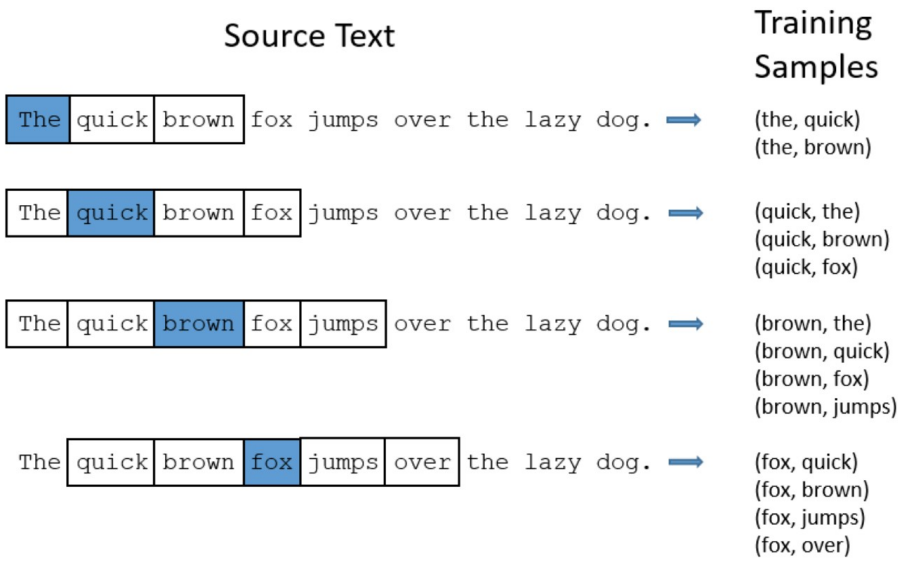
```
linux/tools/testing/selftests/powerpc/pmu/ebb/instruction_count_test.c
  ❯ leblanc
      Possible Leblancian paradox detected

       58┆ if (difference < 0)
       59┆   difference = -difference;

 linux/tools/testing/selftests/timers/adjtick.c
  ❯ leblanc
      Possible Leblancian paradox detected

       39┆ if (val < 0)
       40┆   val = -val;

 linux/tools/testing/selftests/timers/raw_skew.c
  ❯ leblanc
      Possible Leblancian paradox detected

       41┆ if (val < 0)
       42┆   val = -val;
```

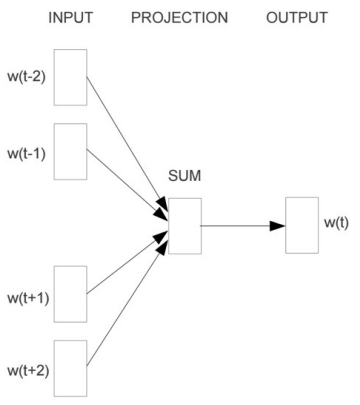
Semgrep results

```
┌──────────────┐
│ Scan Summary │
└──────────────┘
Some files were skipped or only partially analyzed.
  Partially scanned: 25446 files only partially analyzed due to parsing or inter
nal Semgrep errors
  Scan skipped: 98 files larger than 1.0 MB, 319 files matching .semgrepignore p
atterns
  For a full list of skipped files, run semgrep with the --verbose flag.

Ran 1 rule on 58195 files: 194 findings.
macpro:dist lera$
```

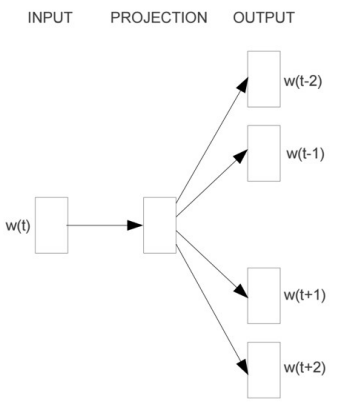
4) Word2Wec

Модель берет на вход текстовый корпус, а на выходе сопоставляет каждому отдельному слову вектор, основываясь на контекстной близости слов друг к другу

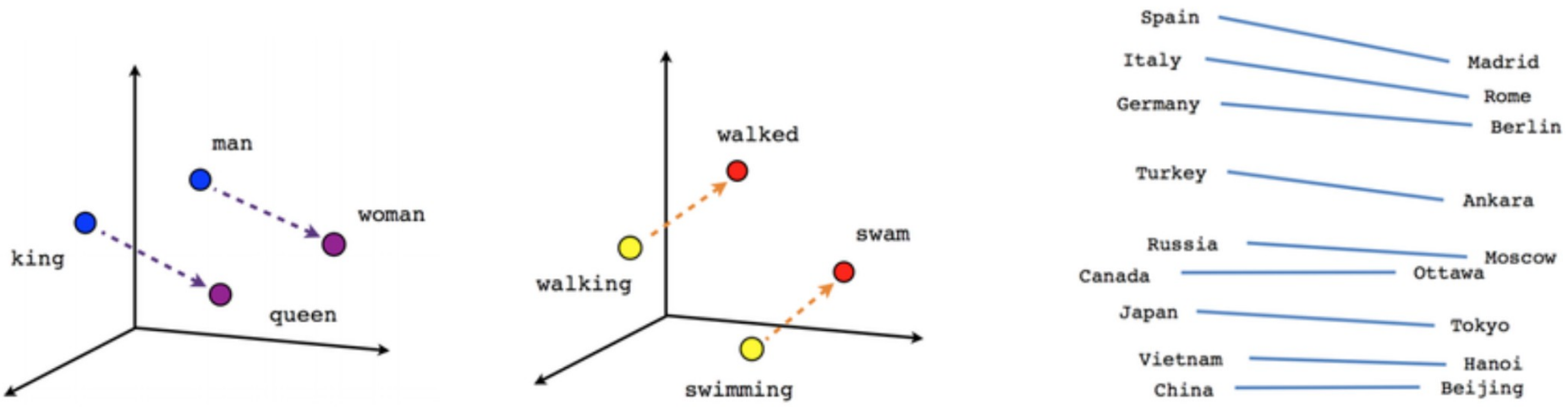




CBOW



Skip-gram



3: Visualization of semantic relationships, e.g. male-female, verb tense and even country-capital relationships between words (Mikolov et al., 2013b).

3.1) Копируем строки с N -5 до N + 5 в отдельный файл. Это будет пример из датасета содержащий уязвимость.
3.2) Копируем строки с N + 10 до N + 20 в файл. Это будет пример из датасета не содержащий уязвимость.
3.3) Копируем строки с N – 3 до N + 7 в файл. Это будет пример из валидационного датасета содержащий уязвимость.
3.4) Копируем строки с N + 12 до N + 22 в файл. Это будет пример из валидационного датасета не содержащий уязвимость.

6) Сбор датасата

Train dataset, vulnerable example:

```
WORD32 id = in_data[band];
if (id != 0) {
   id_sign = 0;
   if (id < 0) {
      id = -id;
      id_sign = 1;
   }
}
```

Validation dataset, vulnerable example:

```
if (id != 0) {
   id_sign = 0;
   if (id < 0) {
      id = -id;
      id_sign = 1;
   }
}
huff_bits += ixheaace_write_bits(pstr_bit_buf, p_huff_tab[id].value, p_huff_tab[id].length);
if (id != 0) {
   huff_bits += ixheaace_write_bits(pstr_bit_buf, id_sign, 1);
}
```

Validation dataset, not vulnerable example:

```
double sin_half_angle = std::sqrt(1.0 - cos_half_angle * cos_half_angle);
if (sin_half_angle < kEpsilon) {
    return *this;
}
double half_angle = std::acos(cos_half_angle);
double scaleA = std::sin((1 - t) * half_angle) / sin_half_angle;
```

TRain dataset, not vulnerable sample:

```
if (cos_half_angle > 1)
    cos_half_angle = 1;
double sin_half_angle = std::sqrt(1.0 - cos_half_angle * cos_half_angle);
if (sin_half_angle < kEpsilon) {
    return *this;
}
double half_angle = std::acos(cos_half_angle);
```

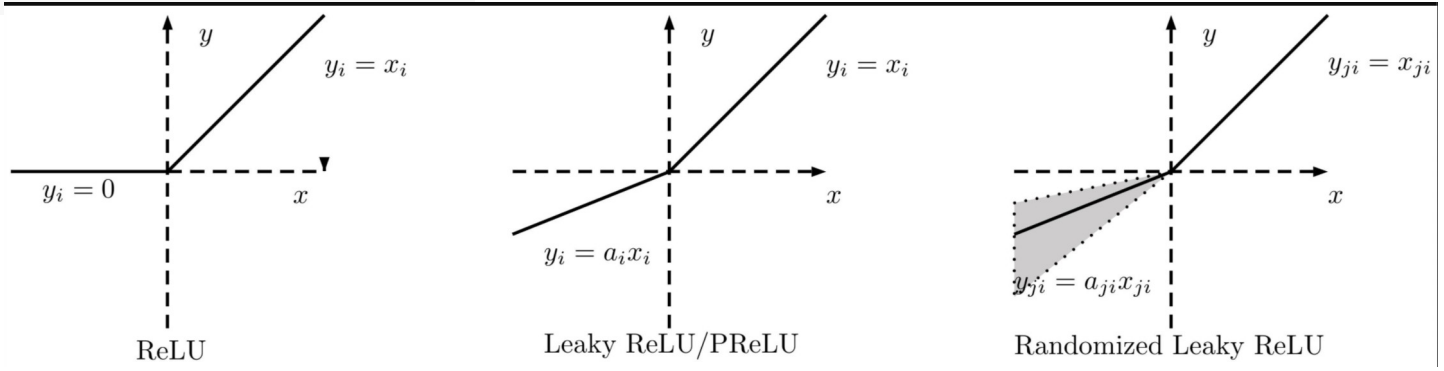
7) Реализация нейронной сети

Модель содержала 2 линейных слоя, между которыми нелинейная функция активации LeakyReLU

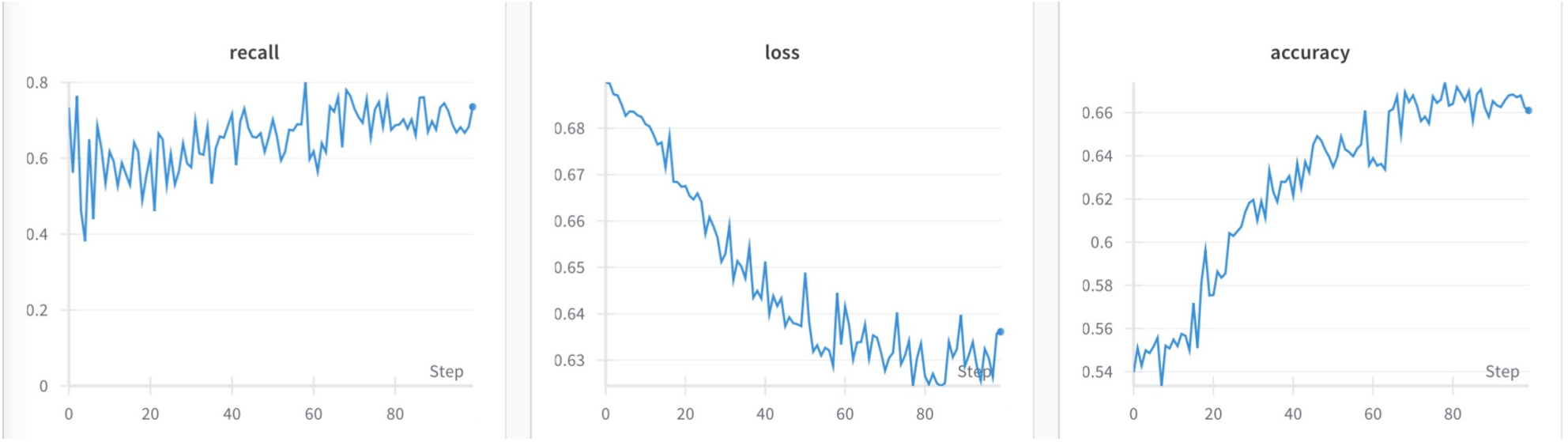
```
class MLP_Model(nn.Module):
    def __init__(self):
        super().__init__()

        self.fc1 = nn.Linear(30*100, 100)
        self.fc2 = nn.Linear(100, 2)
        self.relu = nn.LeakyReLU()
```

CLASS torch.nn.Linear(*in_features*, *out_features*, *bias=True*, *device=None*, *dtype=None*) [SOURCE]

Applies a linear transformation to the incoming data: $y = xA^T + b$.



По результатам ее работы на валидационной выборке были построены графики loss, recall и accuracy:



max recall: 0.78
max accuracy: 0.674

Как видно, соблюдается тенденция типичного поведения нармально обученной модели.

Проект android/external/lua/

```
static int num2straux (char *buff, int sz, lua_Number x) {
   ...
    int e;
    lua_Number m = l_mathop(frexp)(x, &e);  /* 'x' fraction and exponent */
    int n = 0;  /* character count */
    if (m < 0) {  /* is number negative? */
      buff[n++] = '-';  /* add sign */
      m = -m;  /* make it positive */
    }
    ...
  ...
}
```

Видна попытка сделать переменную 'm' неотрицательной.

Проект – android/external/libaom/third_party/libyuv

```
// Copy a plane of data
LIBYUV_API
void CopyPlane(const uint8_t* src_y,
               int src_stride_y,
               uint8_t* dst_y,
               int dst_stride_y,
               int width,
               int height) {
  int y;
  void (*CopyRow)(const uint8_t* src, uint8_t* dst, int width) = CopyRow_C;
  // Negative height means invert the image.
  if (height < 0) {
    height = -height;
    dst_y = dst_y + (height - 1) * dst_stride_y;
    dst_stride_y = -dst_stride_y;
  }
  // Coalesce rows.
  if (src_stride_y == width && dst_stride_y == width) {
    width *= height;
    height = 1;
    src_stride_y = dst_stride_y = 0;
  }
  // Nothing to do.
  if (src_y == dst_y && src_stride_y == dst_stride_y) {
    return;
  }
  // Copy plane
  for (y = 0; y < height; ++y) {
    CopyRow(src_y, dst_y, width);
    src_y += src_stride_y;
    dst_y += dst_stride_y;
  }
}
```

Здесь если переменная 'height' будет отрицательной, то указатель 'dst_y' после операции:
"dst_y = dst_y + (height -1) * dst_stride_y" будет указывать в область слева от dst_y,
так как смещение отрицательное.
Далее в цикле 'for' произойдет запись в границы буфера dst_y.
Это беглый анализ, для более подробного анализа нужно время.

Спасибо за внимание!