Automated Software Vulnerability Detection with Deep Learning for Natural Language Processing

Noah Ziems (*Research Scientist*)
Shaoen Wu (*Endowed Chair Professor*)

School of Information Technology
Illinois State University

CENTER FOR CYBERSECURITY RESEARCH AND EDUCATION
Illinois State University

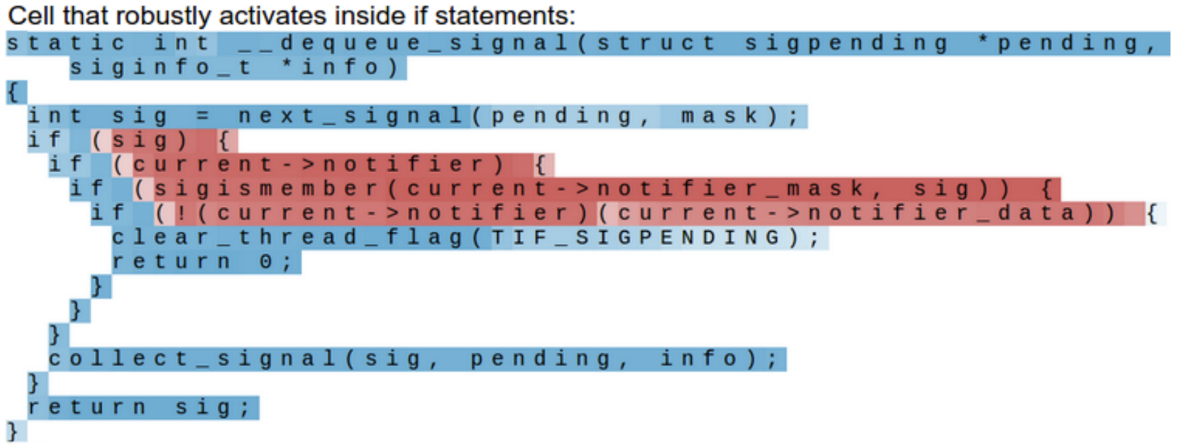
Literature Background

Traditional Methods for Detecting Security Vulnerabilities

- Static Analysis
  - Pre-written set of rules
  - Does not execute code
  - **Prone to false positives**
- Dynamic Analysis
  - Unit Tests
  - Written by programmer
  - Executes code
  - **Vulnerabilities must be anticipated**

Deep Learning Natural Language Processing for Code

- Very good at finding patterns in text
- Previous Work on Code
  - Karpathy et al.(2015)
  - Lachaux et al.(2020)
  - Rozerie et al.(2021)
  - OpenAI Codex(2021)


Cell that robustly activates inside if statements:

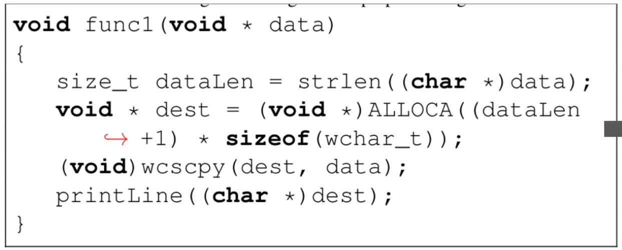
```
static int __dequeue_signal(struct sigpending *pending,
   siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

Software Vulnerability Detection with DL-NLP

Objectives

- General
  - Warn coders of security vulnerabilities in C++/Java
  - Specific Line
- Technical Details
  - State of the Art Transformer Language Model(DOBF)
  - Pretraining on C++/Java
    - Open Source Github Projects
  - Custom tokenizer using clang python library(C++)
  - function-wise evaluation

In Summary



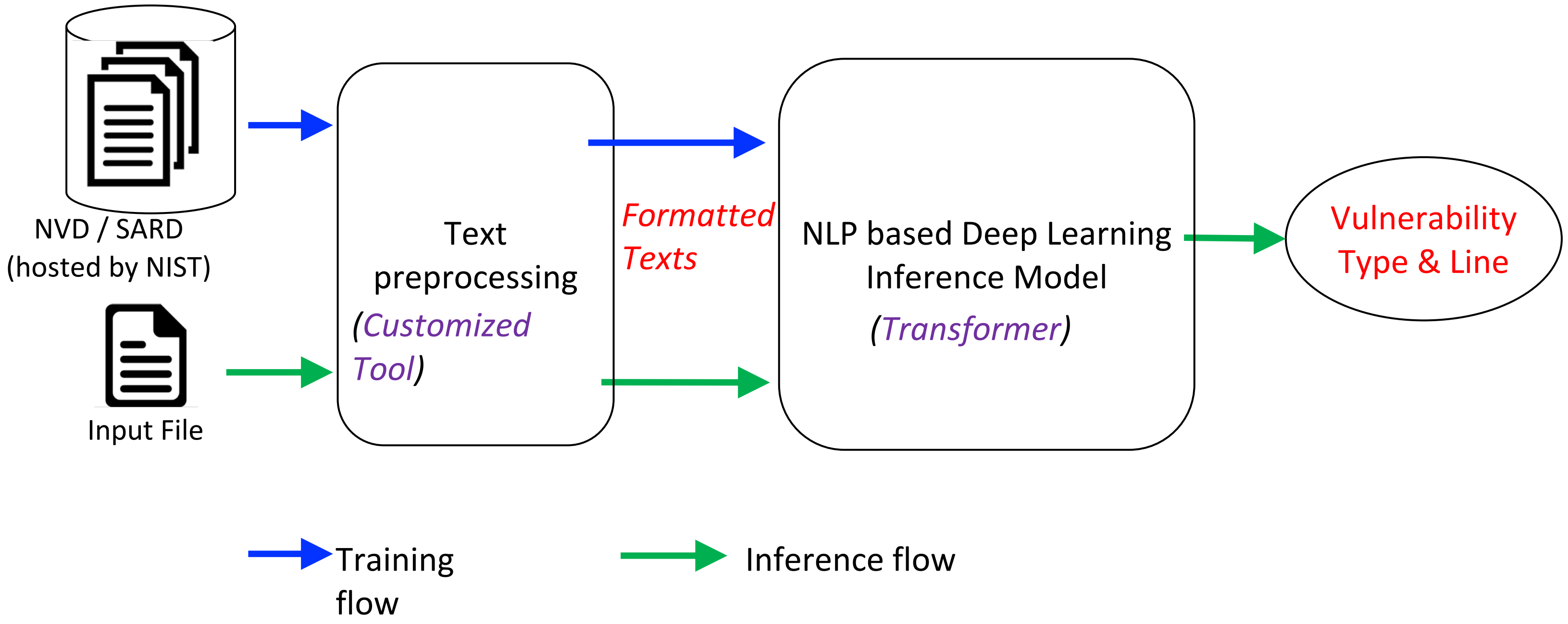
```
void func1(void * data)
{
   size_t dataLen = strlen((char *)data);
   void * dest = (void *)ALLOCA((dataLen
       ↪ +1) * sizeof(wchar_t));
   (void)wcscpy(dest, data);
   printLine((char *)dest);
}
```

Real Time

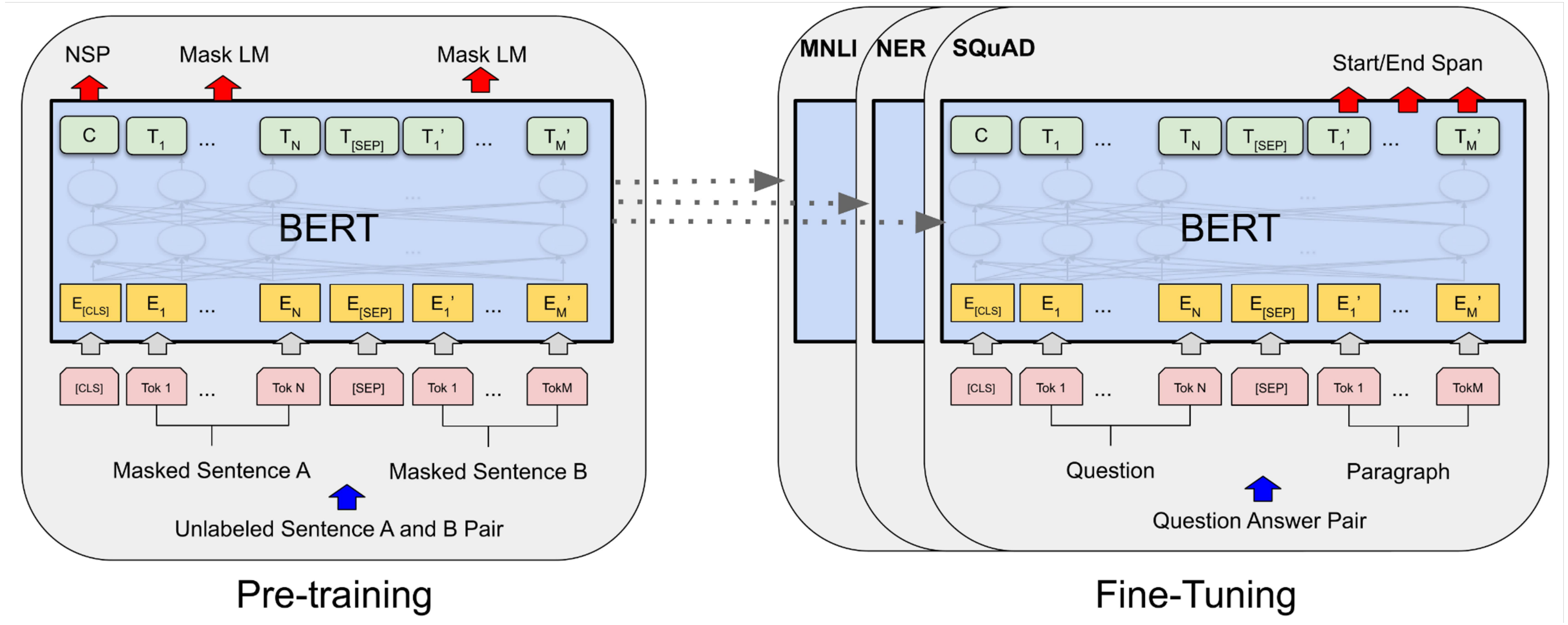
- 76% Chance of CWE-476
- Line 3 in func1
- NULL Pointer Dereference

Programmer
Fixes

Dataset

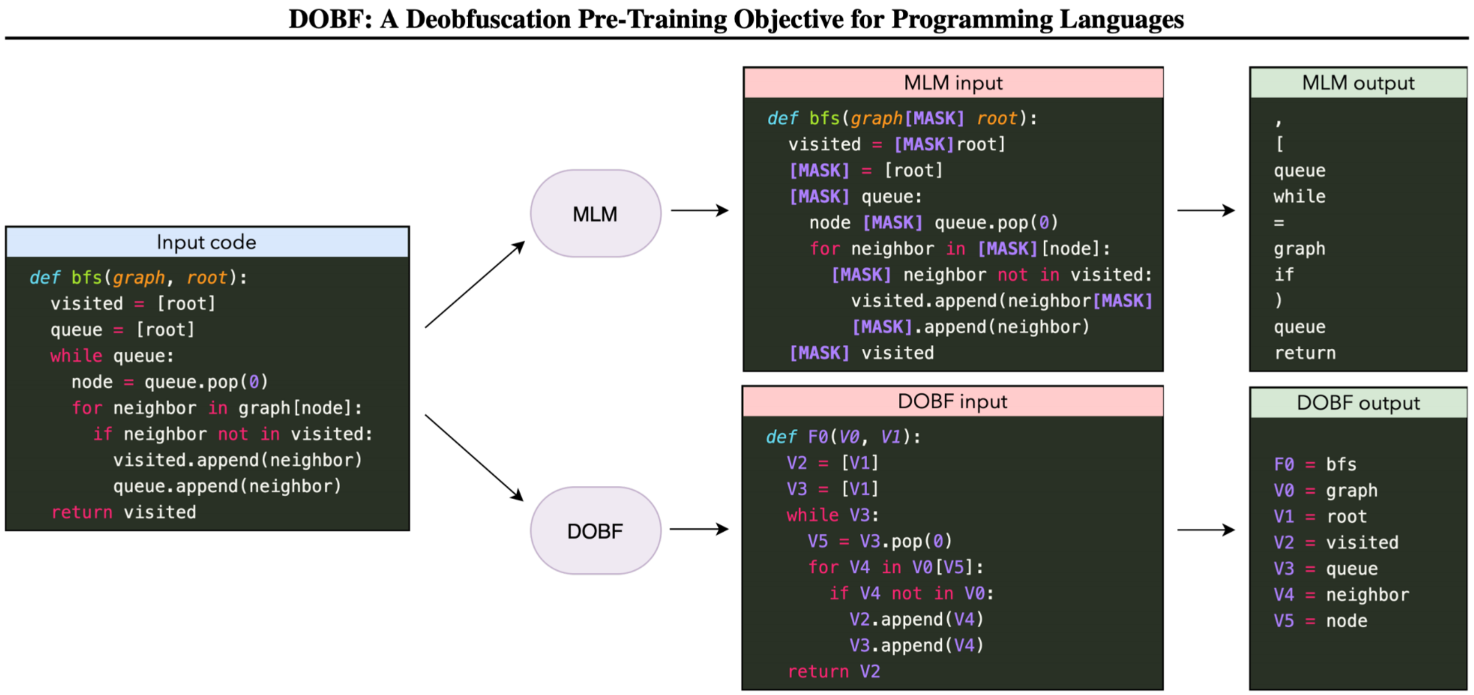

NVD / SARD
(hosted by NIST)
Input File
Text preprocessing
(Customized Tool)
Formatted Texts
NLP based Deep Learning Inference Model
(Transformer)
Vulnerability Type & Line
Training flow
Inference flow

Pre-Training Intuition



Source: Devlin et. al (2019)

Pre-Training & Custom Tokenization



DOBF: A Deobfuscation Pre-Training Objective for Programming Languages
MLM
DOBF

Input code

```
def bfs(graph, root):
  visited = [root]
  queue = [root]
  while queue:
    node = queue.pop(0)
    for neighbor in graph[node]:
      if neighbor not in visited:
        visited.append(neighbor)
        queue.append(neighbor)
  return visited
```

MLM input

```
def bfs(graph[MASK] root):
  visited = [MASK]root]
  [MASK] = [root]
  [MASK] queue:
    node [MASK] queue.pop(0)
    for neighbor in [MASK][node]:
      [MASK] neighbor not in visited:
        visited.append(neighbor[MASK]
        [MASK].append(neighbor)
  [MASK] visited
```

MLM output

```
,
[
queue
while
=
graph
if
)
queue
return
```

DOBF input

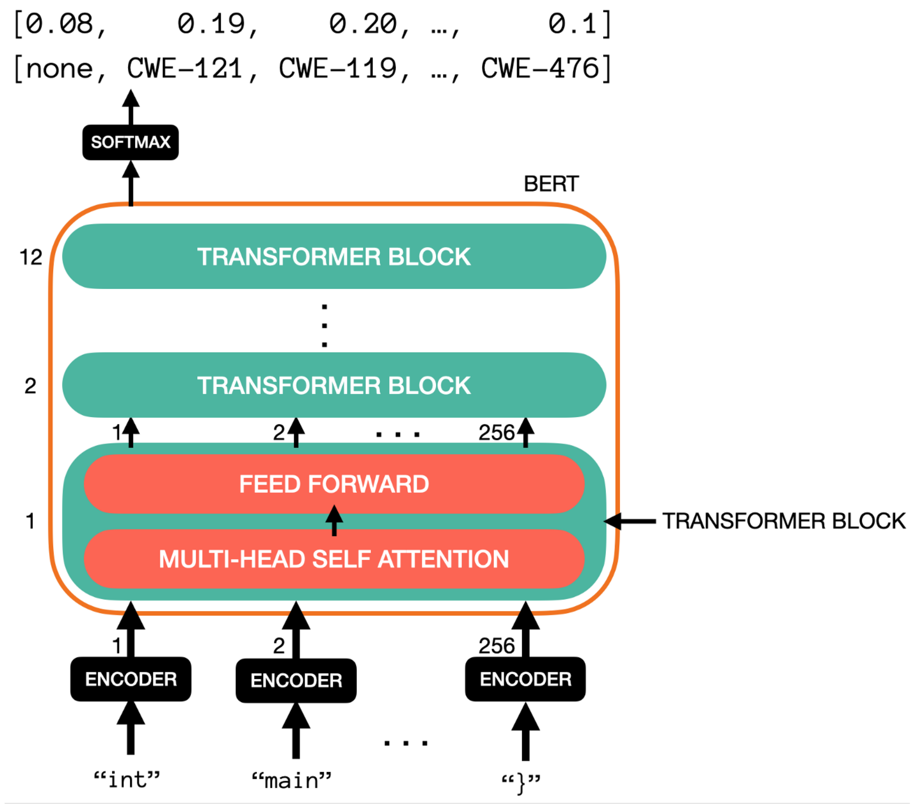
```
def F0(V0, V1):
  V2 = [V1]
  V3 = [V1]
  while V3:
    V5 = V3.pop(0)
    for V4 in V0[V5]:
      if V4 not in V0:
        V2.append(V4)
        V3.append(V4)
  return V2
```

DOBF output

```
F0 = bfs
V0 = graph
V1 = root
V2 = visited
V3 = queue
V4 = neighbor
V5 = node
```

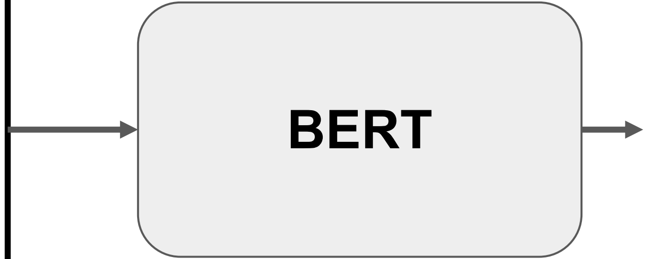
Source: Roziere et. al (2021)

Architecture Diagram



Detection Results (C & C++)

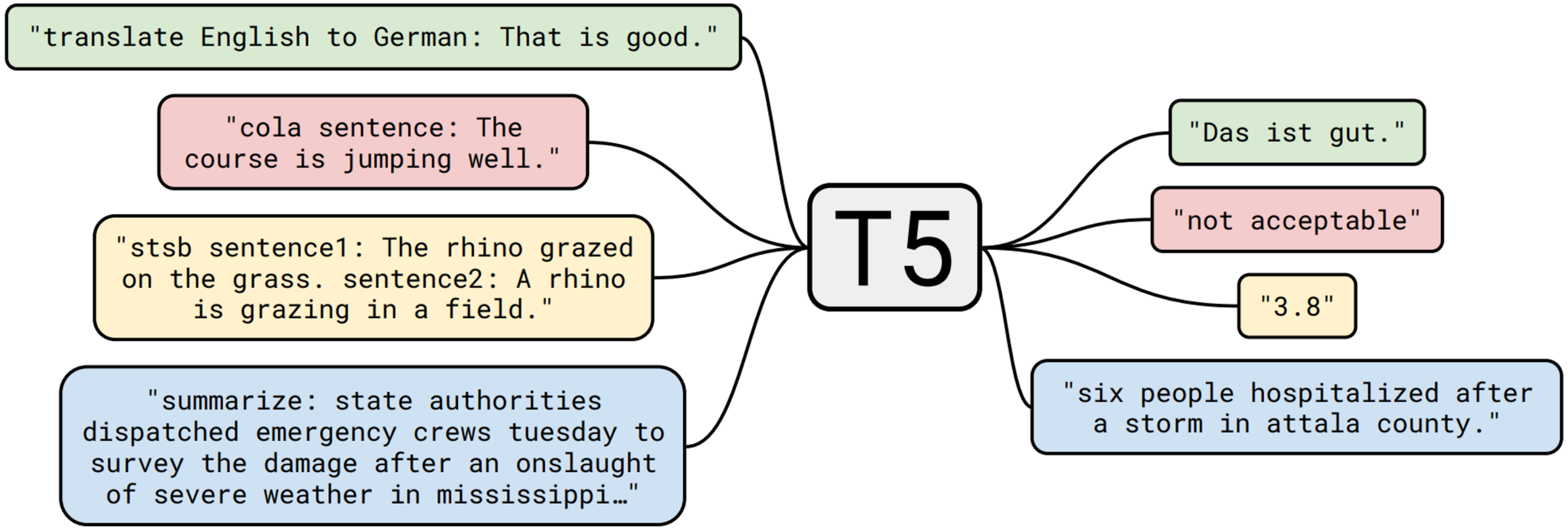
```
void func()
{
    switch(6)
    {
    case 6:
    {
        HCRYPTPROV hCryptProv;
        HCRYPTHASH hHash;
        FILE *pFile = NULL;
        char
password[PASSWORD_INPUT_SIZE];
        UCHAR savedHash[SHA1_SUM_SIZE],
calcHash[SHA1_SUM_SIZE];
        DWORD hashSize;
        char *replace;
        size_t i;
        pFile = fopen("password.txt", "r");
        if (pFile == NULL)
        {
            exit(1);
        }
...
```



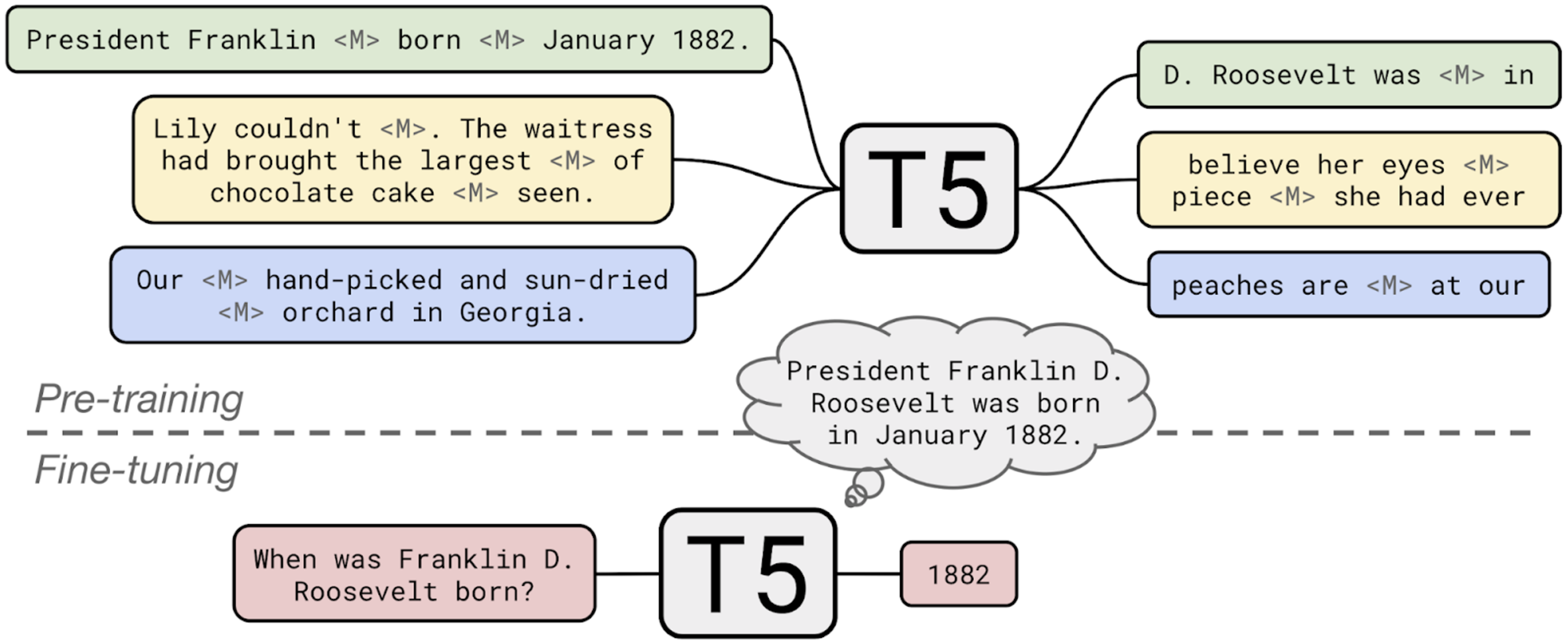
CWE 328: Reversible One-Way Hash

Detection Metrics(Very Good)
SARD: 93%
Draper VDISC: 98% MultiClass(5)
(https://osf.io/d45bw/)

Description Generation with Google's T5 Architecture

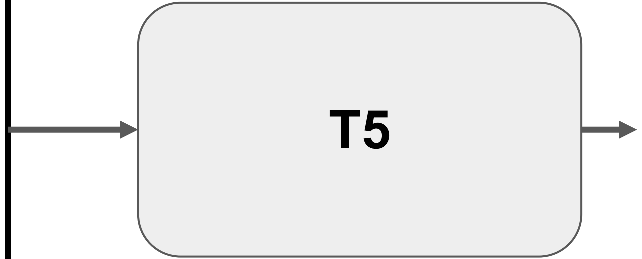

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Description Generation with Google's T5 Architecture


President Franklin <M> born <M> January 1882.
Lily couldn't <M>. The waitress had brought the largest <M> of chocolate cake <M> seen.
Our <M> hand-picked and sun-dried <M> orchard in Georgia.
T5
D. Roosevelt was <M> in
believe her eyes <M> piece <M> she had ever
peaches are <M> at our
President Franklin D. Roosevelt was born in January 1882.
Pre-training
Fine-tuning
When was Franklin D. Roosevelt born?
T5
1882

Generation Results: Actual Output from Model

```
void func()
{
    switch(6)
    {
    case 6:
    {
        HCRYPTPROV hCryptProv;
        HCRYPTHASH hHash;
        FILE *pFile = NULL;
        char
password[PASSWORD_INPUT_SIZE];
        UCHAR savedHash[SHA1_SUM_SIZE],
calcHash[SHA1_SUM_SIZE];
        DWORD hashSize;
        char *replace;
        size_t i;
        pFile = fopen("password.txt", "r");
        if (pFile == NULL)
        {
            exit(1);
        }
...
```



FLAW: Use a reversible hash (SHA1)
Flaw Located ~line 24

Generation Metrics(Very Good)
Rouge: 0.5942
Bleu: 0.4718

Thanks!

Questions