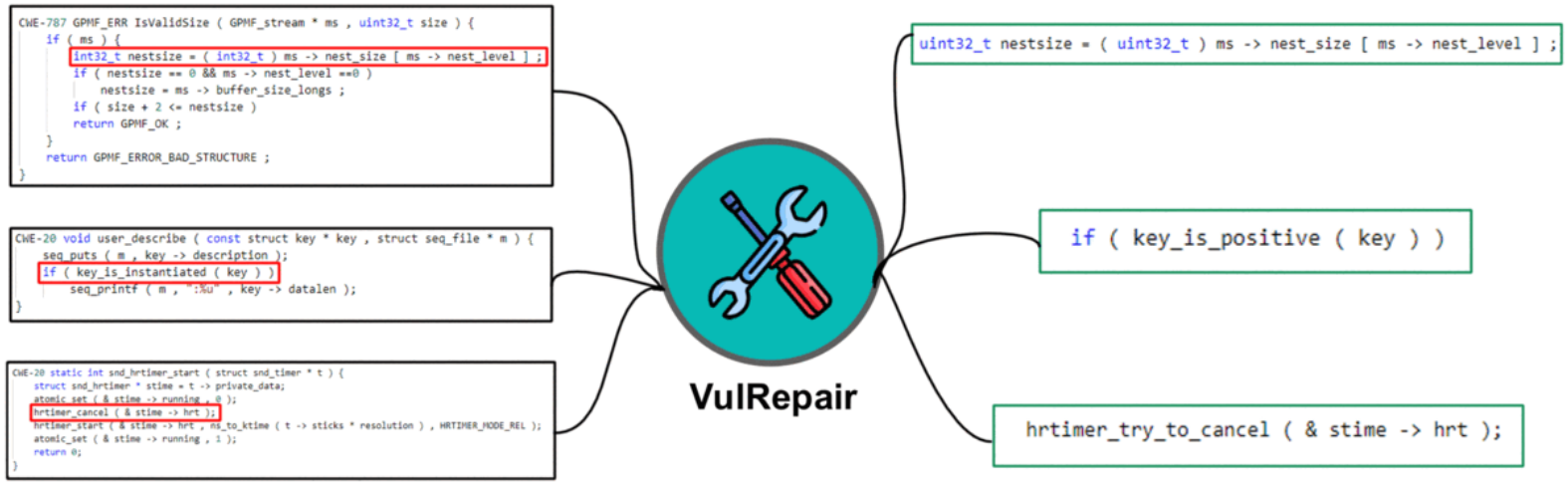
VulRepair: A T5-based Automated Software Vulnerability Repair




Michael Fu
MONASH University


Kla Tantithamthavorn
MONASH University


Trung Le
MONASH University


Van Nguyễn
MONASH University


Dinh Phung
MONASH University

http://chakkrit.com @klainfo

Accepted at The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) 2022

Cybercrimes Are Costly

Vulnerabilities are security flaws in code that attackers can exploit to harm organisations and communities.

According to the National Vulnerability Database, the software vulnerabilities discovered every year have skyrocketed from 4k in 2011 to 20k in 2021.

The global cost of cybercrime is also estimated to reach

$10.5 trillion USD by 2025 – up from $3 trillion in 2015.

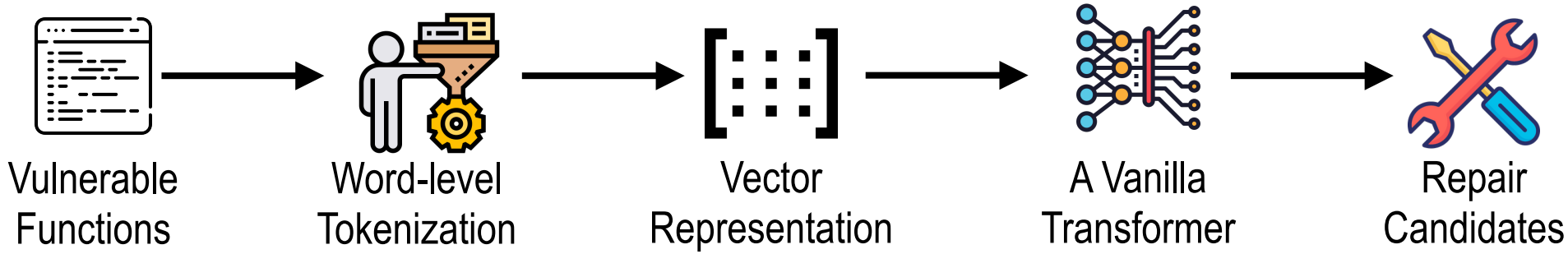
AI-Powered Vulnerability Solutions

Vulnerability Detection
(e.g., VulDeePecker, Devign)

Vulnerability Localization
(e.g., LineVul, LineVD)

Security analysts still have to spend effort on manually fixing and repairing vulnerabilities

NMT-based Vulnerability Repair (VRepair, Chen et al)



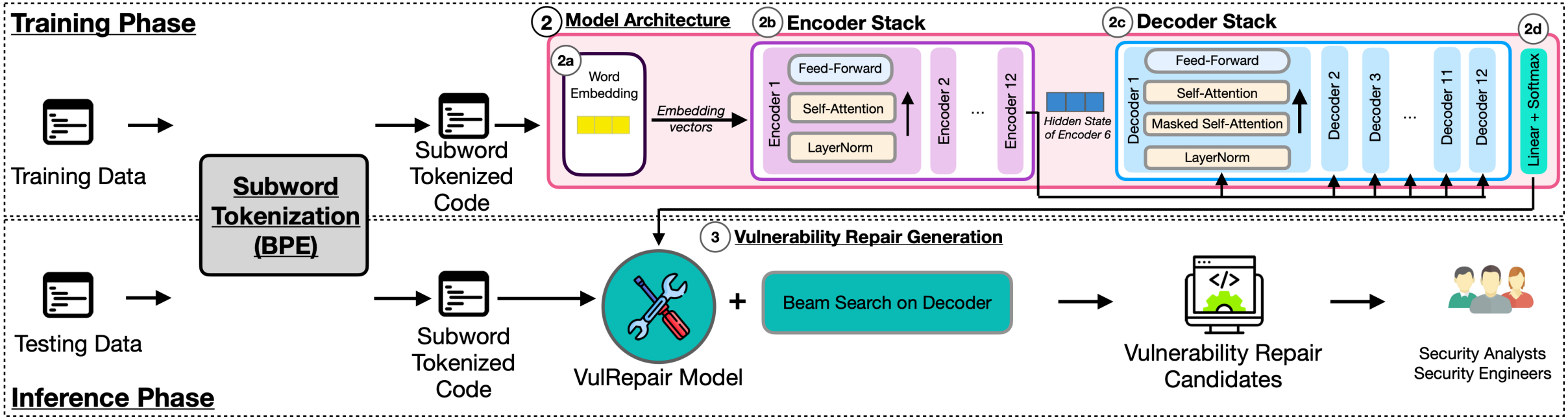
1. *Leverages a small bug-fix corpus of 23k functions for model **pre-training**, limiting its ability to generate optimal vector representation.*

2. *Leverages a **word-level tokenization**, limiting its ability to generate new tokens that never appear in a vulnerable function.*

3. *Leverages a **Vanilla Transformer**, limiting its ability to learn the relative position information of code tokens.*

Zimin Chen, Steve Kommrusch, and Martin Monperrus, Neural Transfer Learning for Repairing Security Vulnerabilities in C Code, IEEE Transactions on Software Engineering (TSE), 2021.

VulRepair: A T5-based Vulnerability Repair



- *Pre-trained on large code base -> Effectively generate more meaningful vector representation.*
- *BPE subword tokenisation -> Effectively generate unknown code tokens.*
- *Relative positional encoding -> Effectively capture the location of each token.*

Research Questions & Experimental Setup

RQ1 *What is the accuracy of our VulRepair for generating software vulnerability repairs?*

RQ2 *What is the benefit of using a pre-training component for vulnerability repairs?*

RQ3 *What is the benefit of using BPE tokenization for vulnerability repairs?*

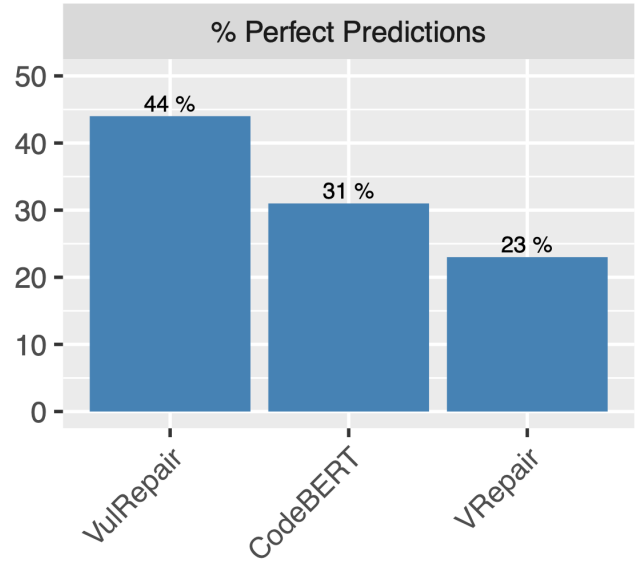
RQ4 *What are the contributions of the components of our VulRepair?*

Datasets: *CVE-Fixes and Big-Vul (a total of 8K pairs)*

Split: *Same as Chen et al, 70% for training, 10% for validation, and 20% for testing*

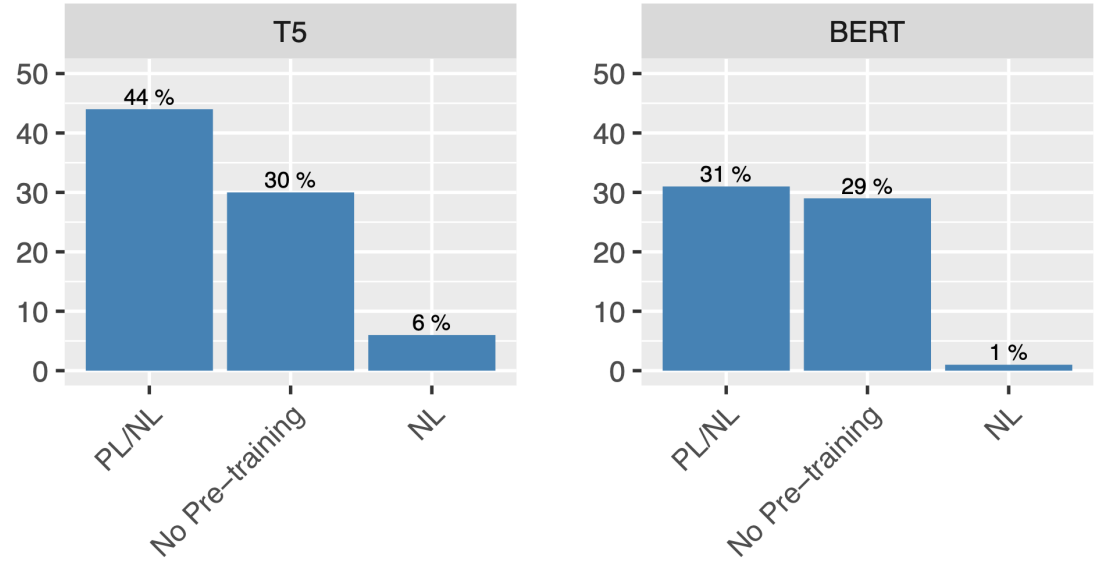
Baselines: *CodeBERT and VRepair (Chen et al)*

RQ1 *What is the accuracy of our VulRepair for generating vulnerability repairs?*



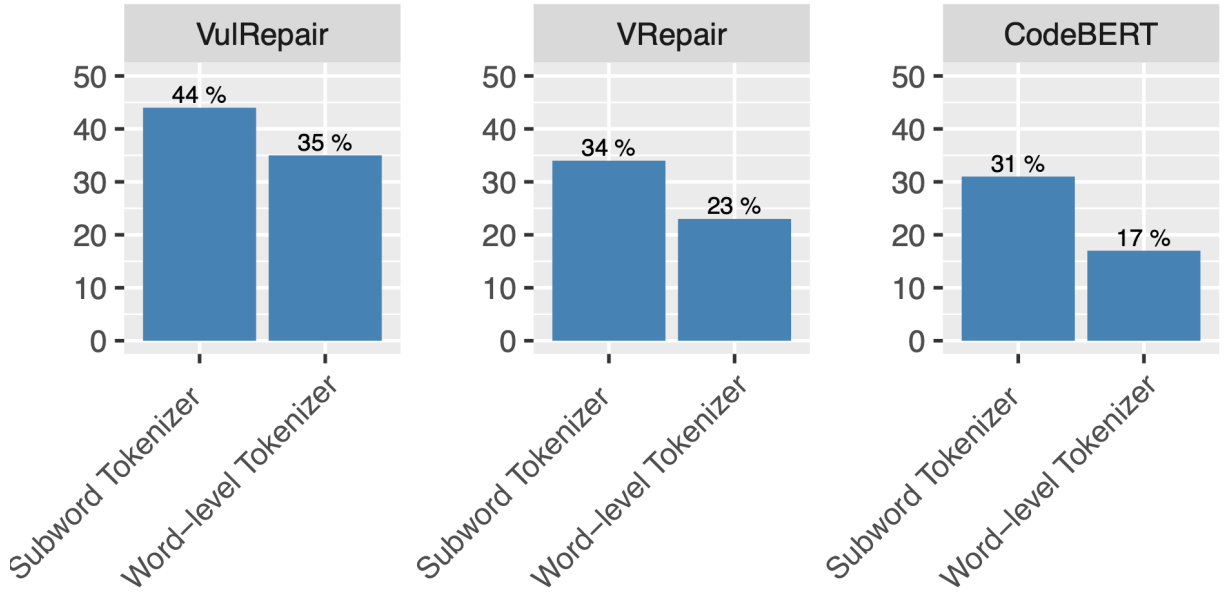
Our VulRepair achieves a Perfect Prediction of 44%, which is 13%-21% more accurate than the baseline approaches.

RQ2 *What is the benefit of using a pre-training component for vulnerability repairs?*



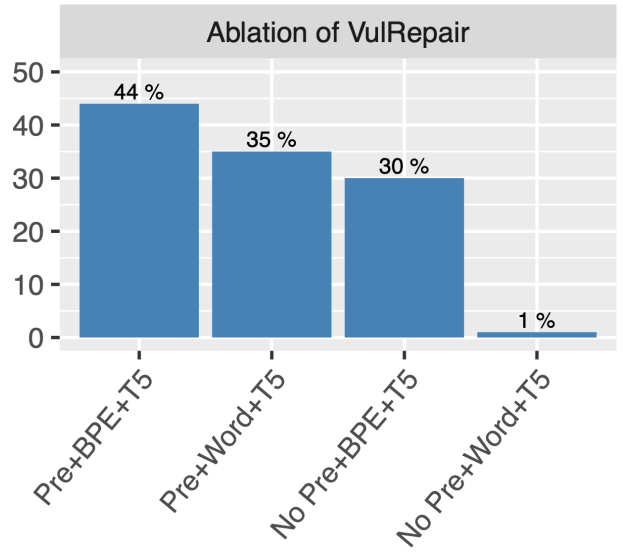
The PL/NL-based pre-training corpus improves the percentage of perfect predictions by 30%-38% for vulnerability repair approaches.

RQ3 *What is the benefit of using BPE tokenization for vulnerability repairs?*



BPE improves the percentage of perfect predictions by 9%-14% for vulnerability repair approaches.

RQ4 *What are the contributions of the components of our VulRepair?*



The pre-training component of our VulRepair is the most important component.

VulRepair can accurately repair as many as 745 out of 1,706 real-world well-known vulnerabilities (e.g., Use After Free, Improper Input Validation, OS Command Injection)

Q1 *What types of CWEs that our VulRepair can correctly repair?*

Rank	CWE Type	Name	%PP	Proportion
1	CWE-787	Out-of-bounds Write	30%	16/53
2	CWE-79	Cross-site Scripting	0%	0/1
3	CWE-125	Out-of-bounds Read	32%	54/170
4	CWE-20	Improper Input Validation	45%	68/152
5	CWE-78	OS Command Injection	33%	1/3
6	CWE-89	SQL Injection	20%	1/5
7	CWE-416	Use After Free	53%	29/55
8	CWE-22	Path Traversal	25%	2/8
9	CWE-352	Cross-Site Request Forgery	0%	0/2
10	CWE-434	Dangerous File Type	-	-
		TOTAL	38%	171/449

Our VulRepair can correctly repair 38% of the vulnerable functions affected by the Top-10 most dangerous CWEs, but cannot accurately repair for some types of rare vulnerabilities.

To handle rare vulnerabilities in the dataset

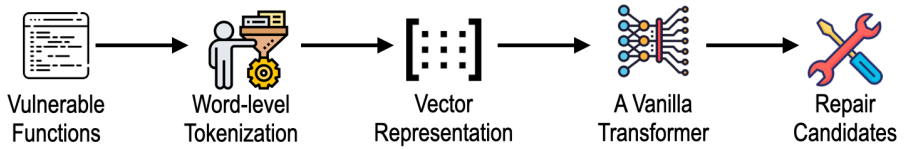
Q2 *How Do the Function Lengths and Repair Lengths Impact the Accuracy of Our VulRepair?*

		Function Lengths (#Tokens)					
		0-100	101-200	201-300	301-400	401-500	500+
Repair Lengths (#Repair Tokens)	0-10	77%	64%	75%	76%	67%	32%
	11-20	63%	56%	59%	43%	33%	32%
	21-30	50%	55%	56%	65%	56%	33%
	31-40	48%	53%	57%	42%	56%	15%
	41-50	54%	61%	53%	45%	20%	30%
	50+	48%	24%	32%	28%	16%	6%

The accuracy of our VulRepair depends on the size of the vulnerable functions and its difficulty to repair.

To handle difficult & complex repairs

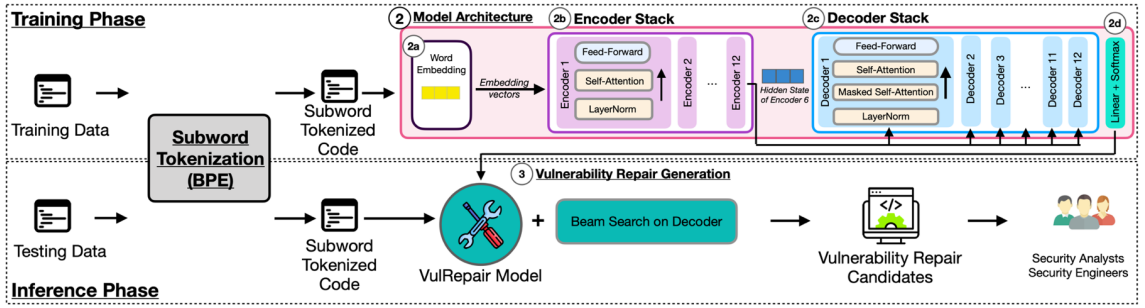
NMT-based Vulnerability Repair (VRepair, Chen et al)



1. *Leverages a small bug-fix corpus of 23k functions for model **pre-training**, limiting its ability to generate optimal vector representation.*
2. *Leverages a **word-level tokenization**, limiting its ability to generate new tokens that never appear in a vulnerable function.*
3. *Leverages a **Vanilla Transformer**, limiting its ability to learn the relative position information of code tokens.*

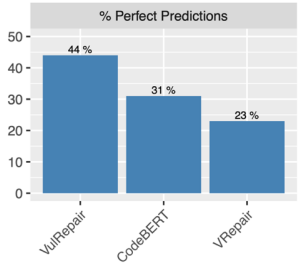
Zimin Chen, Steve Kommrusch, and Martin Monperrus, Neural Transfer Learning for Repairing Security Vulnerabilities in C Code, IEEE Transactions on Software Engineering (TSE), 2021.

VulRepair: A T5-based Vulnerability Repair



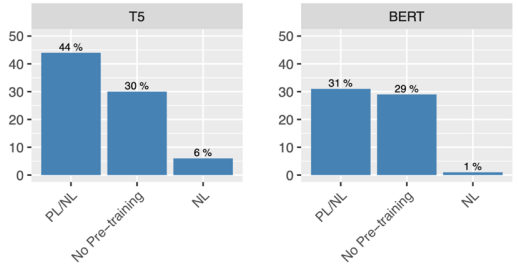
- *Pre-trained on large code base -> Effectively generate more meaningful vector representation.*
- *BPE subword tokenisation -> Effectively generate unknown code tokens.*
- *Relative positional encoding -> Effectively capture the location of each token.*

RQ1 *What is the accuracy of our VulRepair for generating vulnerability repairs?*



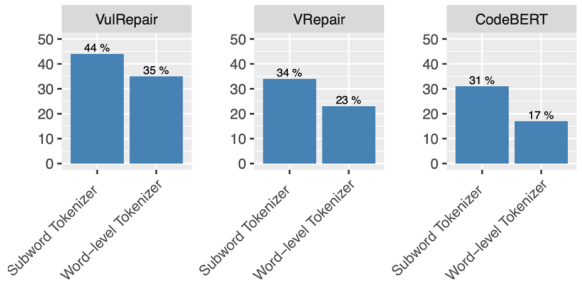
Our VulRepair achieves a Perfect Prediction of 44%, which is 13%-21% more accurate than the baseline approaches.

RQ2 *What is the benefit of using a pre-training component for vulnerability repairs?*



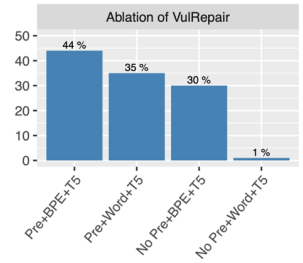
The PL/NL-based pre-training corpus improves the percentage of perfect predictions by 30%-38% for vulnerability repair approaches.

RQ3 *What is the benefit of using BPE tokenization for vulnerability repairs?*



BPE improves the percentage of perfect predictions by 9%-14% for vulnerability repair approaches.

RQ4 *What are the contributions of the components of our VulRepair?*



The pre-training component of our VulRepair is the most important component.

VulRepair can accurately repair as many as 745 out of 1,706 real-world well-known vulnerabilities (e.g., Use After Free, Improper Input Validation, OS Command Injection)