Domain Knowledge enhanced vulnerability Detection in Source Code

Rosmael Zidane Lekeufack Foulefack and Alessandro Marchetto

Department of Information Engineering and Computer Science, University of Trento, Trento, Italy

(rz.lekeufack,alessandro.marchetto)@unitn.it

ICT DAYS

Vuln. Detection in source code

Detecting software vulnerabilities in source code:

- Traditional static analyzers frequently produce a large number of false alarms
- Deep-learning (DL) methods have shown their effectiveness
- Sequential-based DL methods (RNNs, LTSM, Transformer): source code as a flat text
- Graph-based DL methods (GNN): graphs (AST, ...) as intermediate representation of the source code

Previous study

- No statement-level detection capability
- No knowledge about the reasons underlying the vulnerability
- The only element used as the source for features is the source code.

Domain Knowledge

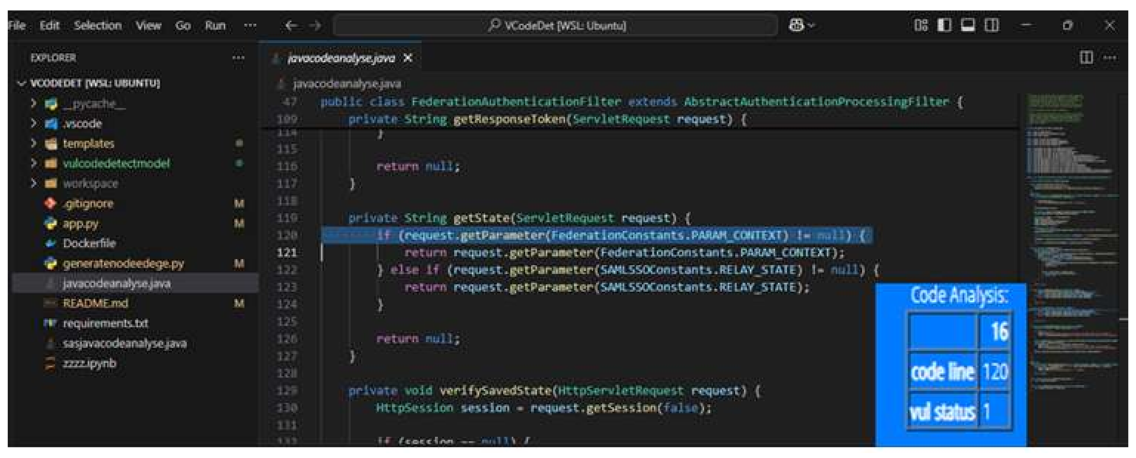
MITRE system: an extract of a CVE description example

CVE-ID: CVE-2020-10221	
Vulnerability name	rConfig OS Command Injection Vulnerability
Affected software	rConfig - Network Configuration Management
Date added	11/03/2021
Required Action	Apply updates per vendor instructions
References to Advisories, Solutions, and Tools	Third Party Exploit: https://engindemirbilek.github.io/rconfig-3.93-rce ...
Description	lib/ajaxHandlers/ajaxAddTemplate.php in rConfig through 3.94 allows remote attackers to execute arbitrary OS commands via shell metacharacters in the fileName POST parameter.
CVSS	Base Score: 8.8 HIGH
Weakness Enumeration	CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Affected Software Configurations	cpe:2.3:a:rconfig:rconfig:3.9.4:*:*:*:*:*:*:*

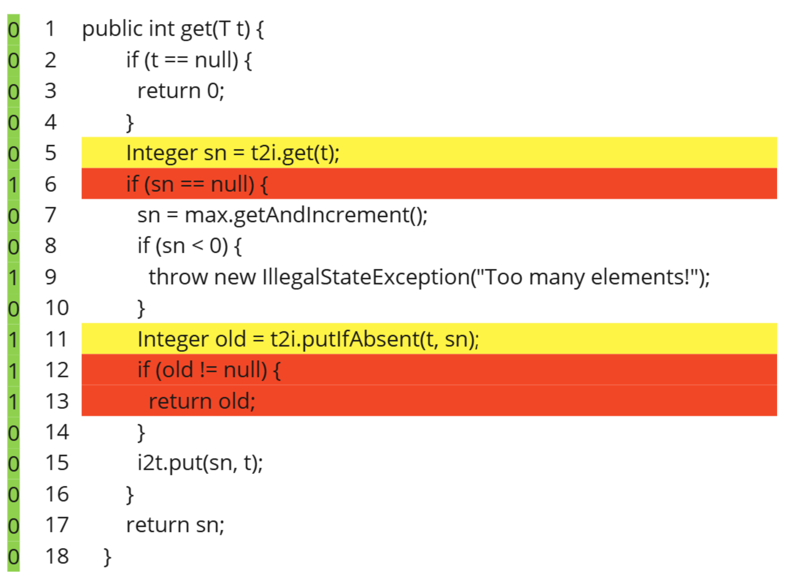
MITRE system: MITRE system: an extract of a CWE description example

CWE-ID: CWE-78	
Weakness Name	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Description	The product constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.
Scope	Confidentiality, Integrity, Availability, Non-Repudiation
Technical Impact	Execute Unauthorized Code or Commands; DoS: Crash, Exit, or Restart; Read Application Data; Modify Application Data; Hide Activities
Relationships: ChildOf	CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection') CWE-74 Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')
Likelihood Of Exploit	High
Examples of code	This code intends to take the name of a user and list the contents of that user's home directory. It is subject to the first variant of OS command injection. `$userName = $_POST["user"];` `$command = 'ls -l /home/' . $userName;` `system($command);` The `$userName` variable is not checked, an attacker could set the `$userName` variable to an arbitrary OS command such as: `;rm -rf /` which results in $command: `ls -l /home/;rm -rf /`. In Unix, semi-colon is a command separator, the OS first executes ls and then rm.

Tool

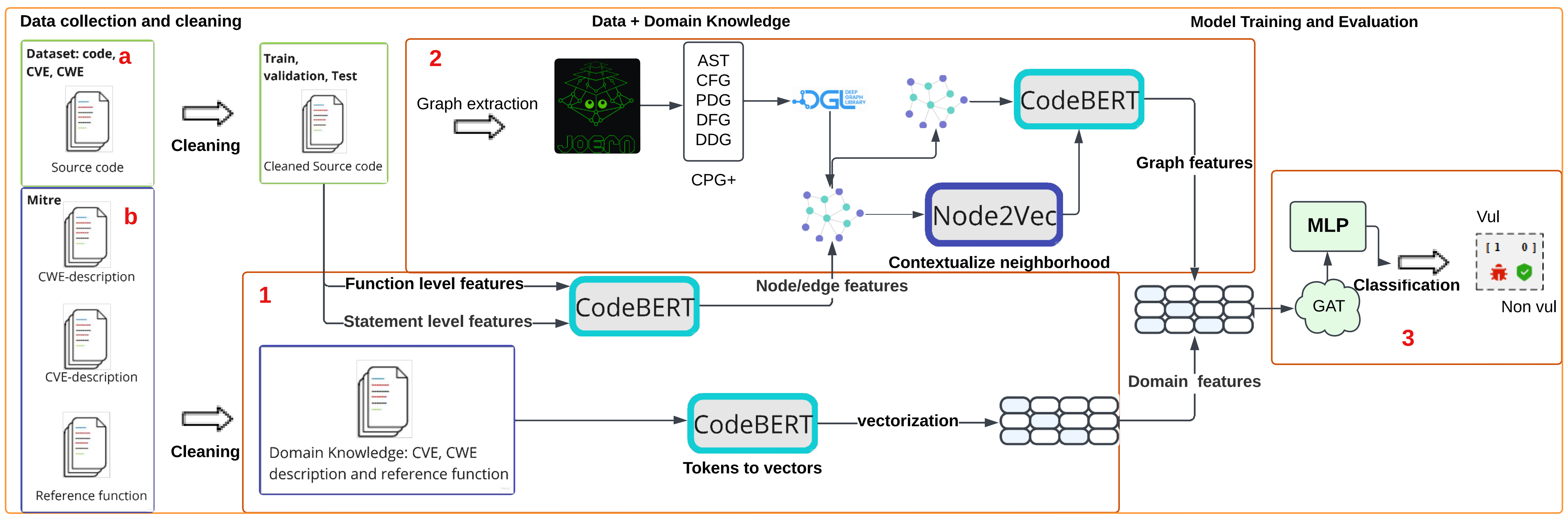


https://github.com/RosmaelZidane/VVulDet



Statement-level vulnerability detection output.

Approach Overview



(a) and **(b)** Collection of domain knowledge information and source code from public benchmarks and datasets of vulnerable code repositories and online resources (MITRE CVEs and CWEs)
1. **Feature Extraction:** from domain knowledge information and source code to vector-based representation by CodeBERT
2. **Graph Construction:** a multi-graph representation (CPG, AST, DDG, ...) is extracted from the source code and embeddings derived by means of Node2Vec and CodeBERT
3. **Classification:** A GAT+MLP model is used for feature learning and making predictions.

Empirical Results

Dataset information

		Function			Statement		
Dataset	Language	Total	Vulnerable	Ratio(%)	Total	Vulnerable	Ratio(%)
ProjectKB	Java	20155	2420	13.64	91052	1826	2.04
Megavul	Java	41949	2433	6.15	377083	3759	1.01
CVEFixes	Python	29597	3305	12.57	646109	4128	0.643
Big-Vul	C/C++	188636	10900	6.33	1016135	24103	2.43

SOTA comparison at Statement-level

Model	Prec	F1-score	Rec	RAUC	PRAUC
LineVD	0.271	0.360	0.533	**0.913**	0.642
IVDetect	0.238	0.176	0.140	0.463	0.520
VVulDet	**0.495**	**0.417**	**0.612**	0.86	**0.725**

SOTA comparison at Function-level

Model	Prec	F1-score	Rec	RAUC	PRAUC
IVDetect	0.093	0.160	0.590	**0.512**	0.106
MVulD	0.183	0.272	0.531	—	0.231
Devign	0.090	0.157	**0.617**	0.420	0.112
VVulDet	**0.245**	**0.331**	0.551	0.5	**0.304**

Performance (macro) metrics of VVulDet in different programming languages

		Statement-level					Function-level				
Dataset	Language	Prec	F1-score	Rec	MCC	PRAUC	Prec	F1-score	Rec	MCC	PRAUC
ProjectKB	Java	0.791	0.736	0.666	0.5	**0.86**	0.484	0.492	0.5	0.0	**0.515**
Megavul	Java	**0.795**	**0.824**	**0.862**	**0.653**	0.702	**0.557**	**0.500**	0.501	**0.17**	0.268
CVEFixes	Python	0.763	0.786	0.813	0.574	0.610	0.498	0.499	0.5	0.02	0.507
Big-Vul	C/C++	0.495	0.417	0.612	0.01	0.725	0.245	0.331	**0.551**	0.0	0.304

Scientific achievements: Paper

1. *Enhanced Graph Neural Networks for Vulnerability Detection in Java via Advanced Subgraph Construction IFIP International Conference on Testing Software and Systems (2024).*
2. *Enhancing Vulnerability Detection with Domain Knowledge: a Comparison of Different Mechanisms IFIP International Conference on Testing Software and Systems (2024).*
3. *Towards a Knowledge Graph based approach for vulnerable code weaknesses identification IFIP International Conference on Testing Software and Systems (2024).*
4. *Incorporating Domain Knowledge into GNNs for Advanced Vulnerability Detection in Java Proceedings of the 6th ACM/IEEE International Conference on Automation of Software Test (AST 2025)*

Demo and Video

Statement-level vulnerability detection example.

UNIVERSITÀ DI TRENTO — Dipartimento di Ingegneria e Scienza dell'Informazione — SEC4AI4SEC — SERICS SECURITY AND RIGHTS IN THE CYBERSPACE