VulSniper: 集中注意力击败细粒度漏洞

段旭 , 吴敬征, 纪守领, 芮志清, 罗天悦, 杨牧天, 武延军

VulSniper: Focus Your Attention to Shoot Fine-Grained Vulnerabilities. IJCAI-2019, 2019: 4665--4671.

联系方式 : 吴敬征 18910958184 jingzheng08@iscas.ac.cn

简介

有研究发现，少量代码差异就可以导致或修复一个漏洞，这导致漏洞和非漏洞程序之间的代码差异会非常小。它可能导致某些传统静态分析方法的准确性降低。我们将相似代码中的漏洞的准确识别定义为细粒度的漏洞检测问题。在本文中，我们提出了VulSniper，旨在更有效地检测细粒度的漏洞。

```
int foo(char *str, size_t n)
{
    char buf[BUF_SIZE], *ar;
    size_t len = strlen(str);

    if(len >= BUF_SIZE) return ERROR;
    memcpy(buf, str, len);

    ar = malloc(n);
    if(!ar) return ERROR;
}
```

(a) Non-vulnerable Program

```
int foo(char *str, size_t n)
{
    char buf[BUF_SIZE], *ar;
    size_t len = strlen(str);

    if(len >= 2*BUF_SIZE) return ERROR;
    memcpy(buf, str, len);

    ar = malloc(n);
    if(!ar) return ERROR;
}
```

(b) Vulnerable Program

图1：“缓冲区错误”漏洞的示例。这两个程序的代码差异较小，这表明了细粒度的必要性。

主要贡献:

1) 我们提出了一种将源代码编码为特征张量的方法，该方法能够有效地避免语义信息丢失。
2) 我们提出了一种用于细粒度漏洞检测的注意力神经网络模型，该模型可以有效地检测细粒度漏洞。
3) 我们在两个基准数据集上对VulSniper进行了评估，实验结果证明了VulSniper的有效性。

方法

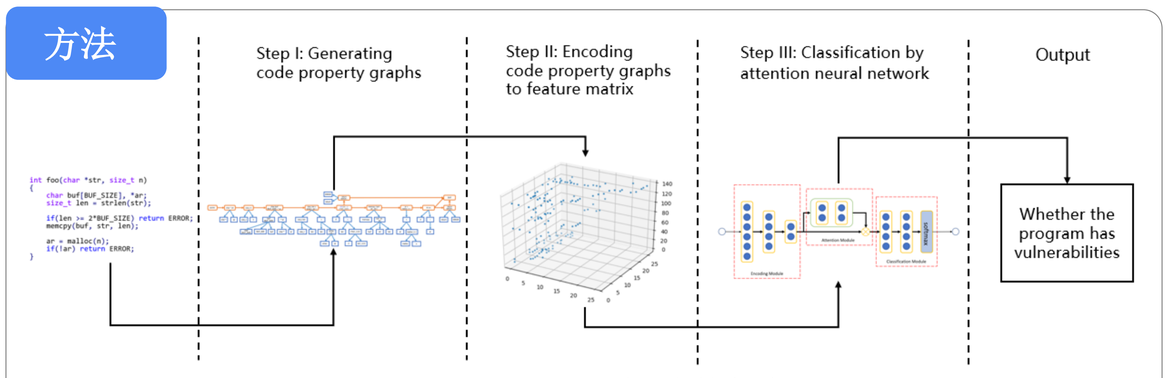

图2：VulSniper的总体框架

VulSniper通过以下步骤检测漏洞：

1）生成代码属性图：

对于程序而言，将代码视为纯文本序列不是一个好的选择，因为它具有更多的语义结构，这与自然语言不同。CPG包含足够的语义信息，可以有效避免语义信息的丢失。我们为图1（b）中的代码生成CPG，结果如图3所示。

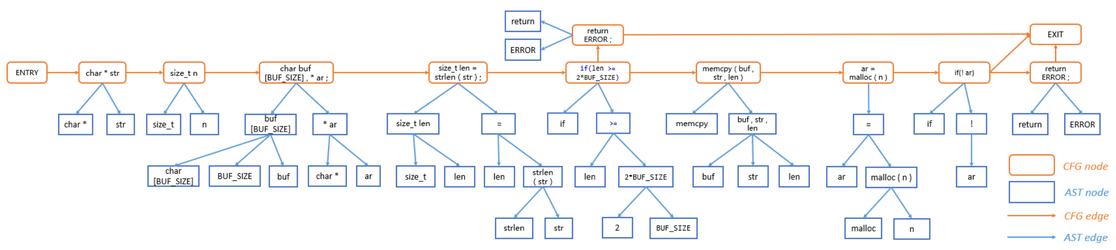

图3：示例程序的简化CPG

2）编码特征张量：

定义 1. 定义$T(G) \in R^{n\times n\times 144}$为一个特征张量，其中 G 是包含n个节点$\{v_1, v_2, \ldots, v_n\}$的代码属性图，并且$\forall t_{i,j,k} \in T(G)$ 满足:

$$t_{i,j,k} = \begin{cases} 1 & is\_match(relations(v_i, v_j), f(k)) \\ 0 & otherwise \end{cases}$$

本文使用$M(v_i, v_j)$ 表示144维特征向量$(t_{i,j,1}, t_{i,j,2}, \ldots, t_{i,j,144})$。其中$M(v_i, v_j)$ 可以根据不同的编码特征拆分为5个字段，分别为: $\{V_i, V_j, I_{ij}, AST_{ij}, CFG_{ij}\}$:

- V_i 和 V_j是19维向量，它们分别对节点v_i和v_j的数据类型，修饰符或特定标志进行编码。
- I_{ij}是29维向量，用于对v_i和v_j之间的运算符进行编码。
- AST_{ij}是57维向量，它编码AST节点之间的父子关系。
- CFG_{ij}是20维向量，用于编码CFG节点之间的邻接关系。

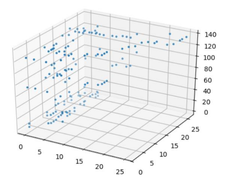

图4：特征张量

3）使用注意力神经网络进行检测：

我们的神经网络模型由三个模块组成：嵌入模块，注意力模块和分类模块。嵌入模块通过展平和映射将特征张量转换为由节点的特征向量组成的矩阵。注意模块将注意权重分配给不同的节点，其由自下而上和自上而下的结构实现。分类模块对特征进行整合并最终实现二分类。我们的神经网络模型的结构如图5(a)所示。

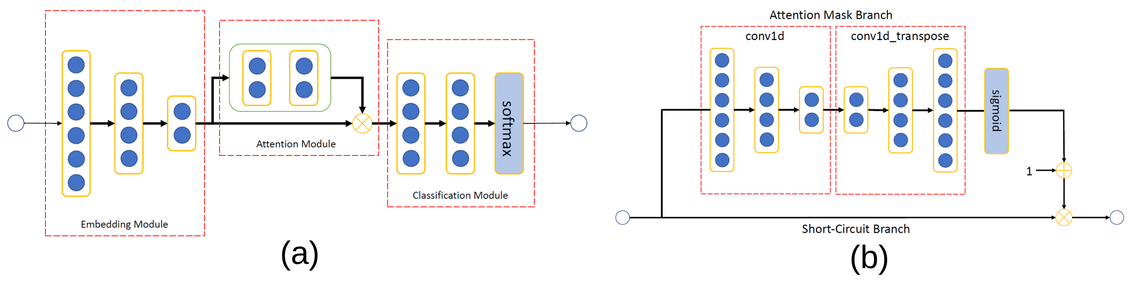

图5：(a) 神经网络模型的结构 (b) 注意模块的结构

如图5(b)所示，注意模块分为两个分支：注意掩模分支和短路分支。短路支路直接将输入作为输出。注意掩模分支通过自下而上和自上而下的结构来学习不同节点的注意权重。

实验

我们对SARD缓冲区错误数据集（CWE-119）和SARD资源管理错误数据集（CWE-399）进行了实验。SARD中的数据包含具有漏洞和不具有漏洞的版本，并且代码差异较小，它们可以有效地评估模型是否具有良好的细粒度漏洞检测能力。我们选择Flawfinder，RATS和DeepSim作为基线，并使用五个常用指标来评估它们在两个数据集上的性能。结果示于表1。

Dataset	Tools	FPR (%)	FNR (%)	TPR (%)	P (%)	F1 (%)
CWE-119	Flawfinder	56.6	44.8	55.2	39.9	46.3
	RATS	68.7	31.3	68.7	40.5	51.0
	DeepSim	16.1	41.6	58.4	71.6	64.4
	VulSniper	6.42	26.2	73.8	88.7	80.6
CWE-399	Flawfinder	40.7	58.4	41.6	34.1	37.4
	RATS	33.9	63.8	36.2	35.0	35.6
	DeepSim	7.30	52.2	47.8	77.2	59.1
	VulSniper	8.49	32.7	67.3	80.4	73.3

表1：两个数据集上不同工具的实验结果

图4中特征张量的注意力权重的可视化结果如图6所示。由于漏洞的类型是缓冲区溢出，因此注意力主要集中在内存操作API（例如memcpy和malloc）周围的区域。同时，BUF_SIZE和“2”也得到了足够的重视，这表明网络已经注意到导致该漏洞的关键因素。

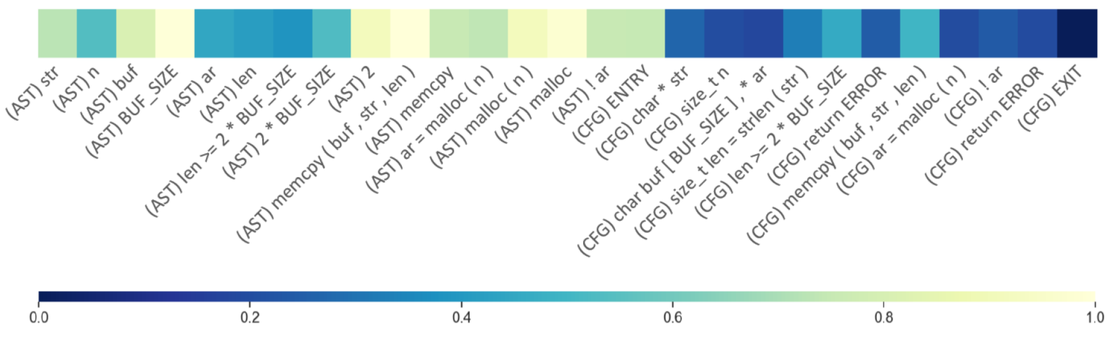

图6：注意权重可视化。亮色和深色分别代表较高和较低的注意权重