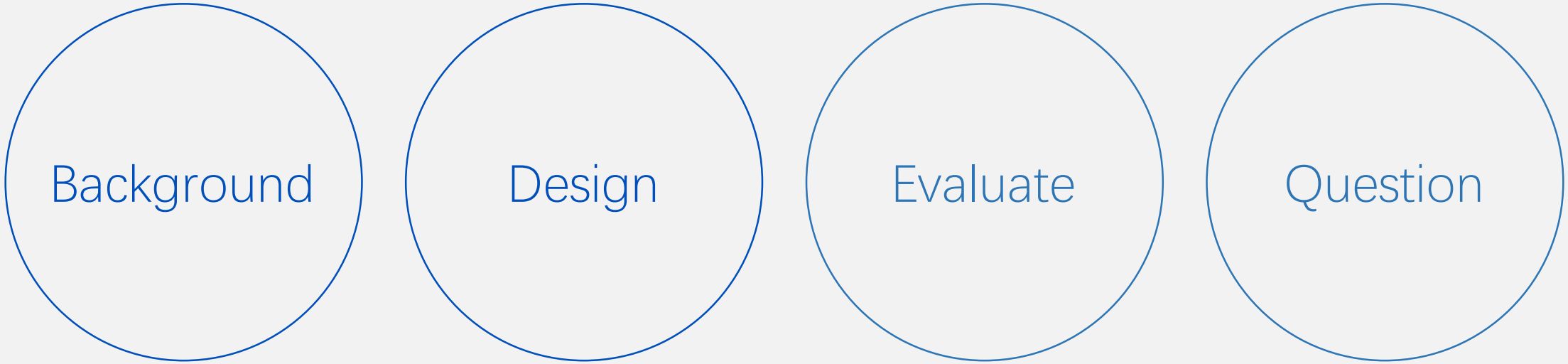
VulDeePecker :

A Deep Learning-Based System for Vulnerability Detection

Huazhong University of Science and Technology

NDSS - 2018

VulDeePecker : A Deep Learning-Based System for Vulnerability Detection



工作背景

依赖人工 ------ 现有方案依赖人类专家定义特征，而深度学习的特征学习更像一门艺术

自动化 ------ 可由网络防御自动化的趋势（DARPA 举办的 CGC 比赛）看出业界想要减轻人类专家的负担

深度学习 ------ 深度学习是为了处理与漏洞检测截然不同的问题

工作比较

VUDDY ------ 二者的漏报率高而误报率低，但只能检测与代码克隆引起的漏洞
非代码克隆引起的漏洞，误报会很高

VulPecker ------ 高误报的系统不能用，高漏报的系统没有用
漏报和误报之间往往相互矛盾

VulDeePecker

深度学习与漏洞检测

如何表示？

- 程序可以首先转换为中间表示，中间表示可以保留元素间的语法关系，之后再将中间表示转换为神经网络接受的向量
- Code Gadget 的概念受到代码重用攻击（Code-Reuse）的启发

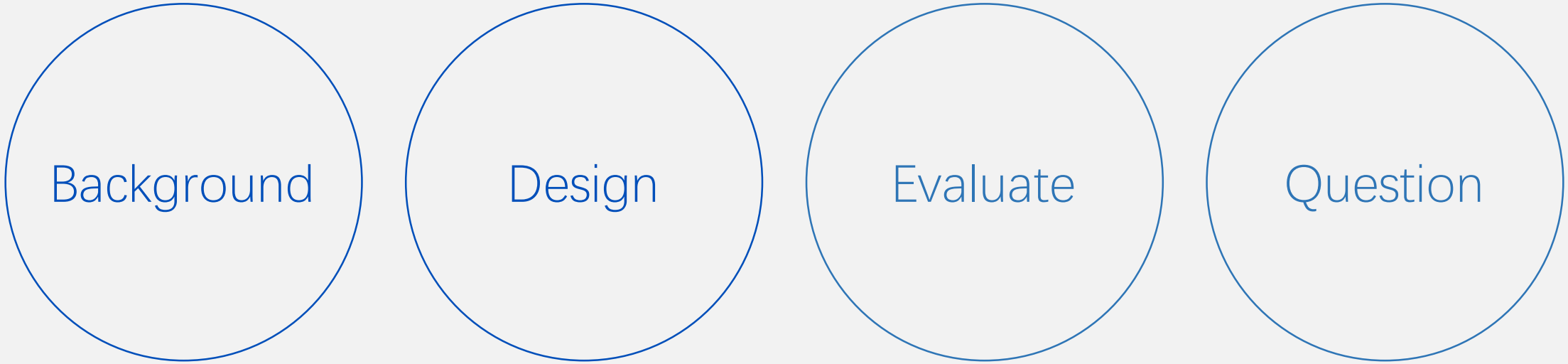
什么粒度？

- 为了识别漏洞的位置，粒度应比程序或函数更加细微

什么网络？

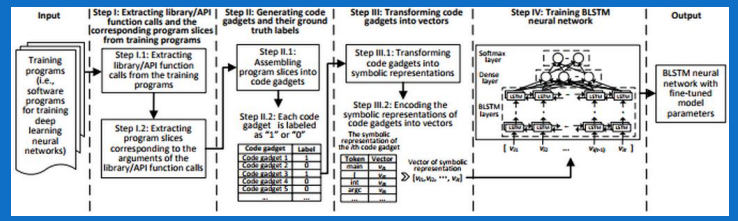
- 可以处理上下文的神经网络更合适
- RNN 已经用于程序分析领域，但其受梯度下降影响很大，LSTM 因为是单向的也不行，所以采用双向的 LSTM（BLSTM）

VulDeePecker : A Deep Learning-Based System for Vulnerability Detection



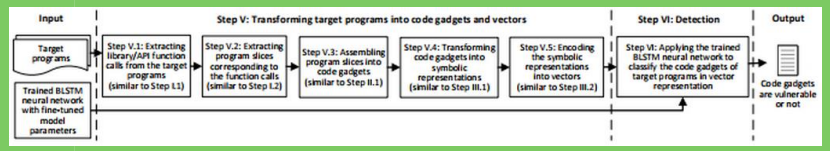
VulDeePecker

训练阶段



- 训练阶段会输入大量程序，有的有问题，有的没问题，输出的是脆弱模式，并编码成 BLSTM 神经网络

检测阶段

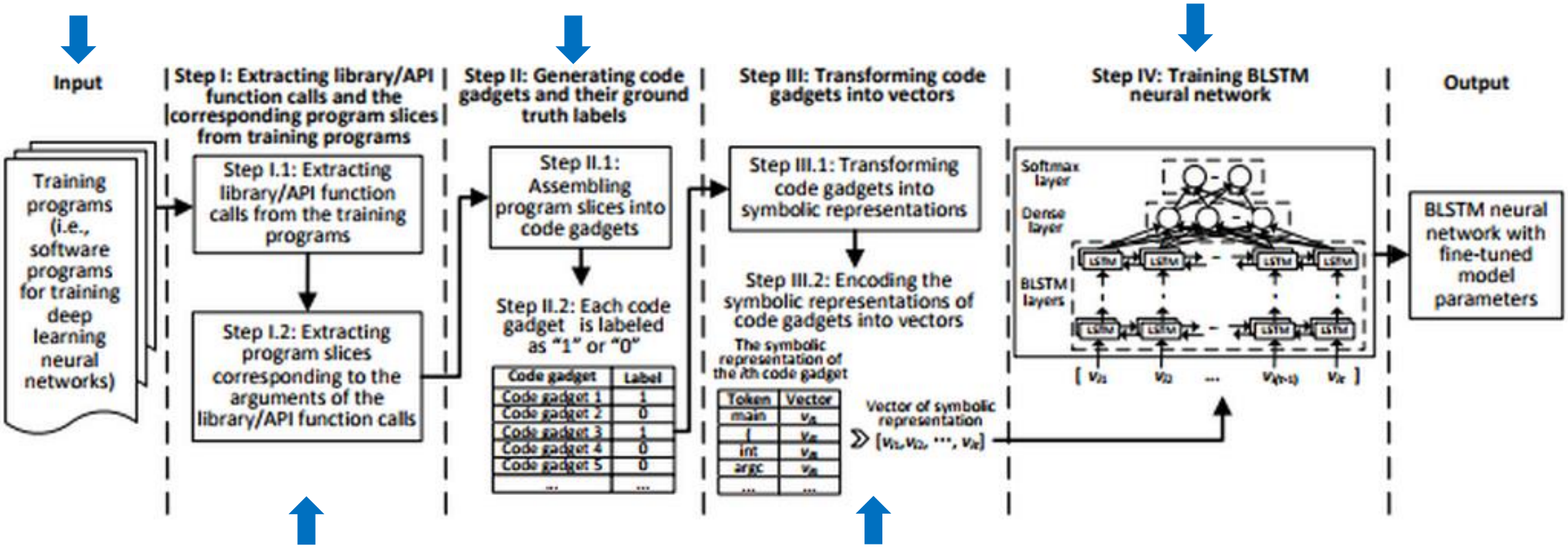


- 转换目标程序到代码段与向量，利用训练阶段的神经网络进行检测

程序切片代表那些与库/API 函数调用参数相关的程序的语句

汇集程序切片为代码片段 -> 为代码片段打标签

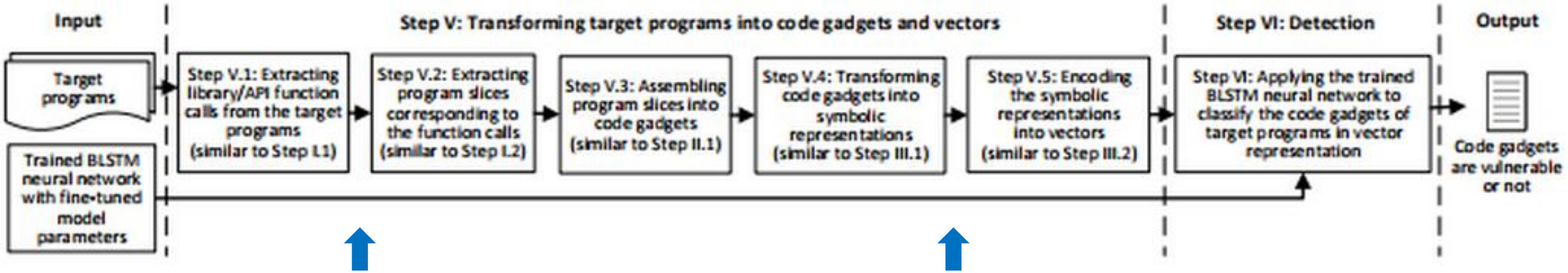
训练 BLSTM 神经网络是标准处理过程



在训练程序中提取库/API函数调用 -> 为函数调用的每个参数提取程序切片

转换代码片段为确定的符号代表 -> 编码符号表示的代码段为向量

提取库/API函数调用 -> 根据参数提取程序切片 -> 组成代码段 -> 转换为符号表示 -> 编码为向量



在训练程序中提取库/API函数调用 ->
为函数调用的每个参数提取程序切片

转换代码片段为确定的符号代表 ->
编码符号表示的代码段为向量

检测过程

提取库/API函数调用与程序切片

前向切片：接受外部输入的数据，受输入参数影响的语句是关键
后向切片：不接受任何外部输入，影响参数值的语句是关键

使用 Checkmarx 提取两种切片（数据依赖），程序切片可以超过函数边界
Checkmarx 用线性结构（链）代表程序切片，也可以用树，线性结构只能代表一个单独的切片，故调用通常对应多个切片

提取代码片段

属于相同、用户定义函数的语句到一个一起，保留语句在用户定义函数的出现顺序

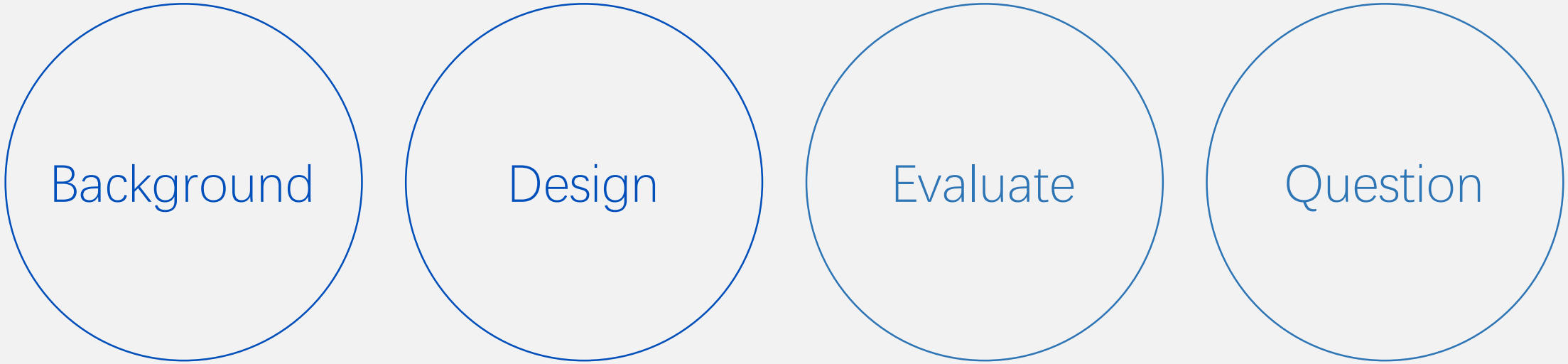
转换代码片段为向量

预处理：移除非 ASCII 字符与注释
映射用户定义变量到符号名称
映射函数到符号名称

通过词法分析把符号表示中的代码段转换成 Token 序列，再使用 word2vec 将 Token 序列转换成向量，首先将 Token 映射成一个整数再转成 fixedlength 向量

BLSTM 需要等长向量作为输入，所以引入定值n：
向量比n短，一个后向切片，在向量起始处填充0
向量比n短，多个后向切片，在向量结尾处填充0
向量比n长，一个后向切片，向量删除开头
向量比n长，多个后向切片，向量删除结尾

VulDeePecker : A Deep Learning-Based System for Vulnerability Detection

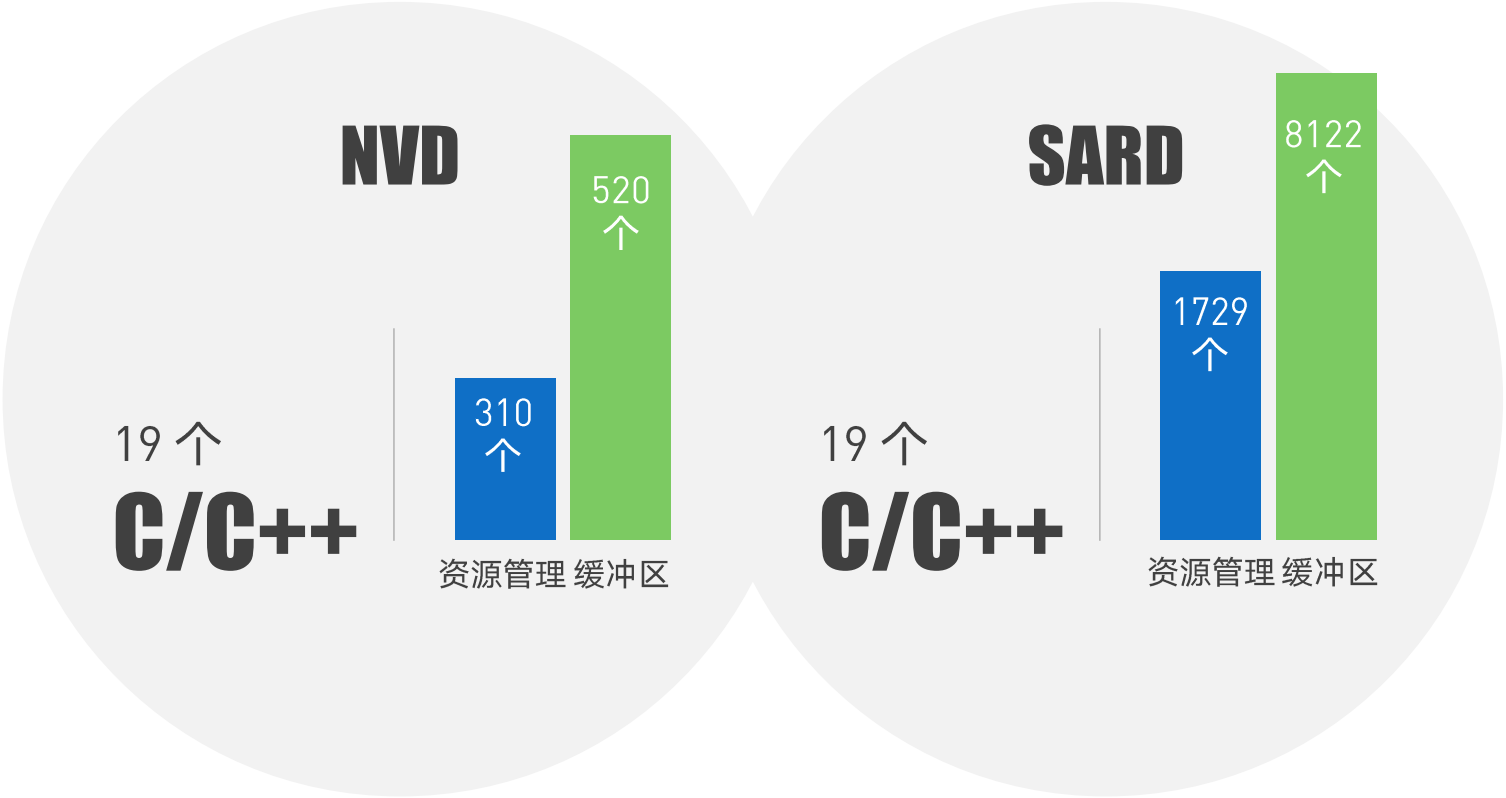


BLSTM 神经网络

Theano + Keras

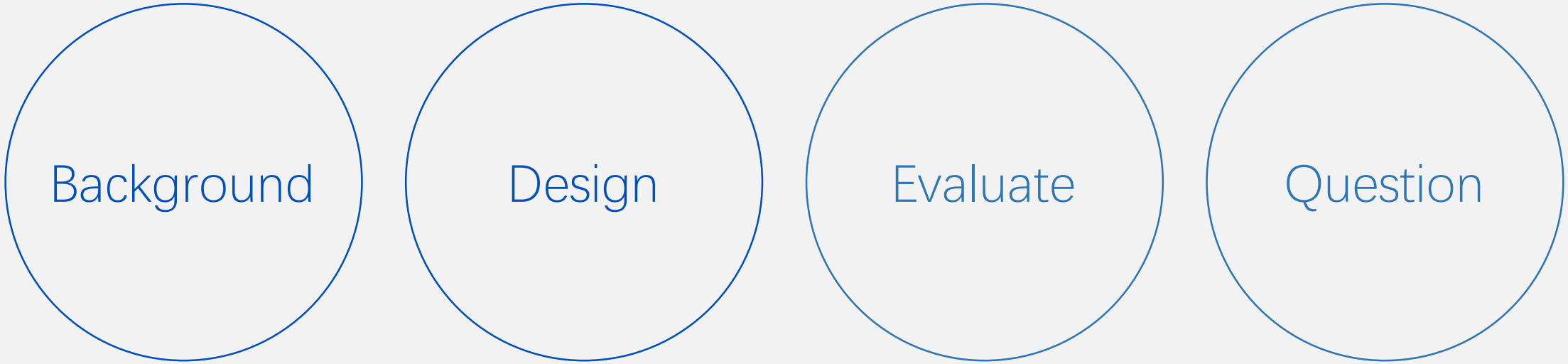
- Dropout – 0.5
- Batch Size – 64
- Number of epoch – 4
- Gradient Descent – Adamax
- Learning Rate – 1.0
- Hidden Node – 300
- Number of layer – 6
- 5 Fold Cross-Validation

训练测试数据集



- Linux Kernel
- Qemu
- Xen
- Gnutls
- Firefox
- Wireshark
- OpenSSL
- VLC

VulDeePecker : A Deep Learning-Based System for Vulnerability Detection



问题与思考

Question

01

量级敏感

该方案可以检测多类型的漏洞，但对调用的数量很敏感，如果调用量太大的情况下，神经网络很难提取模式

02

神经网络

利用神经网络，所以训练时间长，检测时间短

03

人工辅助

人类专家可用于选择调用来提高其有效性

04

数据流分析

利用数据流分析，也许可以帮助基于深度学习的漏洞检测系统

VulDeePecker :

A Deep Learning-Based System for Vulnerability Detection

Q&A

Huazhong University of Science and Technology

NDSS - 2018