Scalable Graph-based Bug Search for Firmware Images

出处：CCS'16
作者：Qian Feng, Rundong Zhou, Chengcheng Xu, Yao Cheng, Brian Testa, and Heng Yin
单位：雪城大学 & 加州大学河滨分校

`Paper Report`

Abstract && Introduction

- 物联网设备爆炸性增长，这些设备的漏洞查找充满挑战
- 其中一个是漏洞查找的效率问题，这篇文章提出高效的查找方案

Methodology

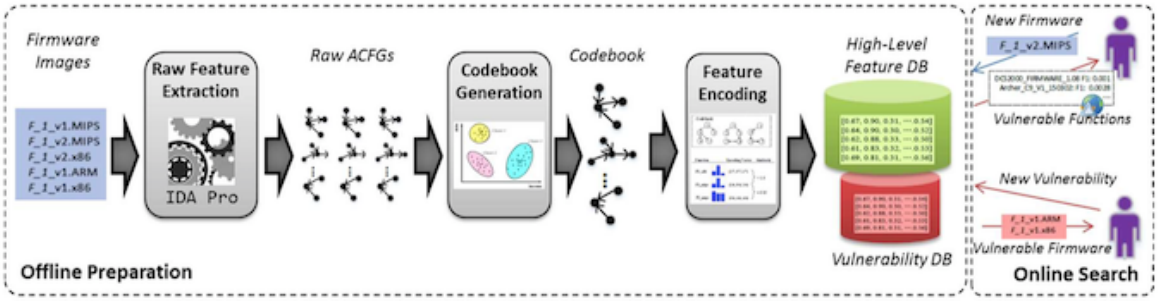


Figure 1: **The approach overview**

Table 1: Basic-block level features used in `Genius`.

Type	Feature Name	Weight (α)
Statistical Features	String Constants	10.82
	Numeric Constants	14.47
	No. of Transfer Instructions	6.54
	No. of Calls	66.22
	No. of Instructions	41.37
	No. of Arithmetic Instructions	55.65
Structural Features	No. of offspring	198.67
	Betweeness	30.66

- ACFG是将CFG抽象以后形成的Attribute向量
- 用到的特征分两类，数据类的是discovRE的内容，结构类的是文章新提出的
- *offspring*是子节点数量

> *Betweenness centrality* is an indicator of a node's centrality in a network. It is equal to the number of shortest paths from all vertices to all others that pass through that node.

Codebook Generation

- Raw Feature Similarity
  - 用IDA得到CFG，然后转成ACFG
  - 用bipartite graph matching计算两个ACFG的相似度值
    - 结果是两个ACFG匹配的成本（cost），类似修改距离
    - 每对基本块的匹配成本和discovRE一致：$\text{cost}(v,\hat{v}) = \frac{\sum_i \alpha_i |a_i - \hat{a}_i|}{\sum_i \alpha_i \max(a_i, \hat{a}_i)}$

    - a是特征值，α是权重
    - 两个ACFG相似度值：$\kappa(g_1, g_2) = 1 - \frac{\text{cost}(g_1, g_2)}{\max(\text{cost}(g_1, \Phi), \text{cost}(\Phi, g_2))}$,
    - Φ是空图
- Clustering
    - 将一个程序的函数分成n个部分，每个部分的中心点是c：$\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$
    - codebook即c的集合

Feature Encoding

> Feature Encoding q: $\mathbb{G} \rightarrow \mathbb{R}^n$ over the codebook $\boldsymbol{C = c_1, \ldots, c_n}$
> For a given graph g_i, $NN(g_i)$ is the nearest centroid neighbors in the codebook:
> $NN(g_i) = argmax_{c_j \in C}(g_i, c_j)$

- 实际中，取一些最近的点
- 然后有：$q(g_i) = \sum_{g_i : NN(g_i) = c_j} [\mathbb{1}(1 = j)\kappa(g_i, c_1), ..., \mathbb{1}(n = j)\kappa(g_i, c_n)]^T$,

Online Search

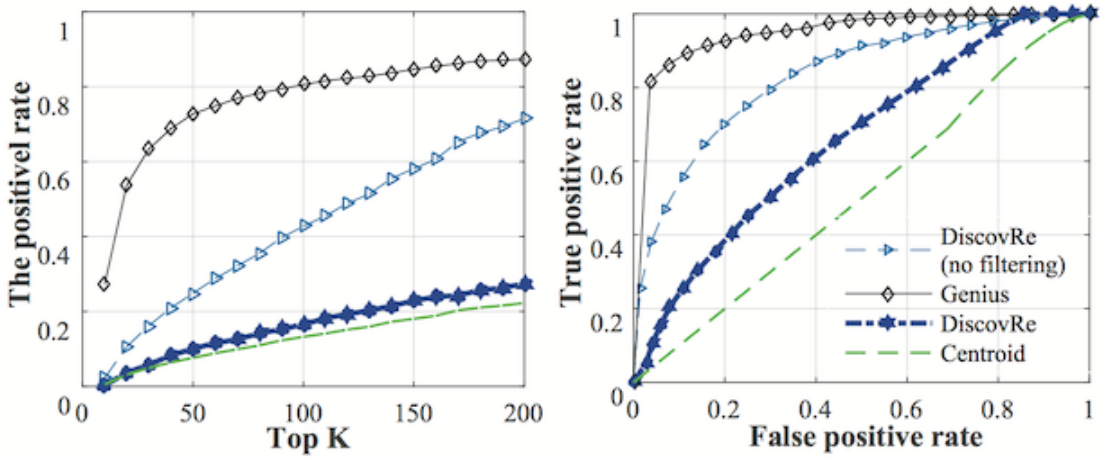
- 用LSH提速

Experiments

Data Preparation

- Baseline evaluation: 和SP'15一样的数据集
- Public dataset: DD-WRT and ReadyNAS
- Firmare image dataset: 从已有的工作中提取固件镜像，总共从26个vendors得到8126个镜像
- Vulnerability dataset: 作者自定义的漏洞库

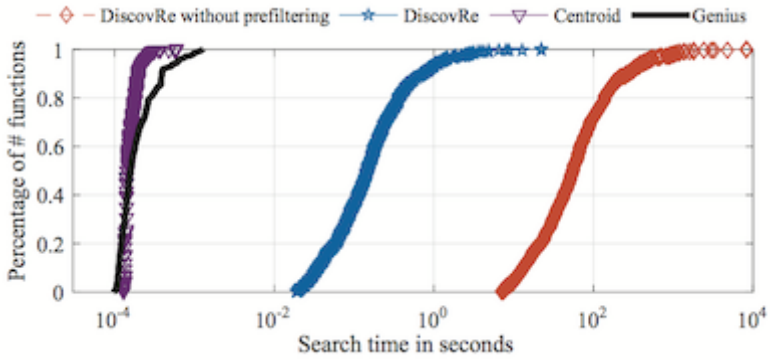
Cross-Platform Baseline Comparison



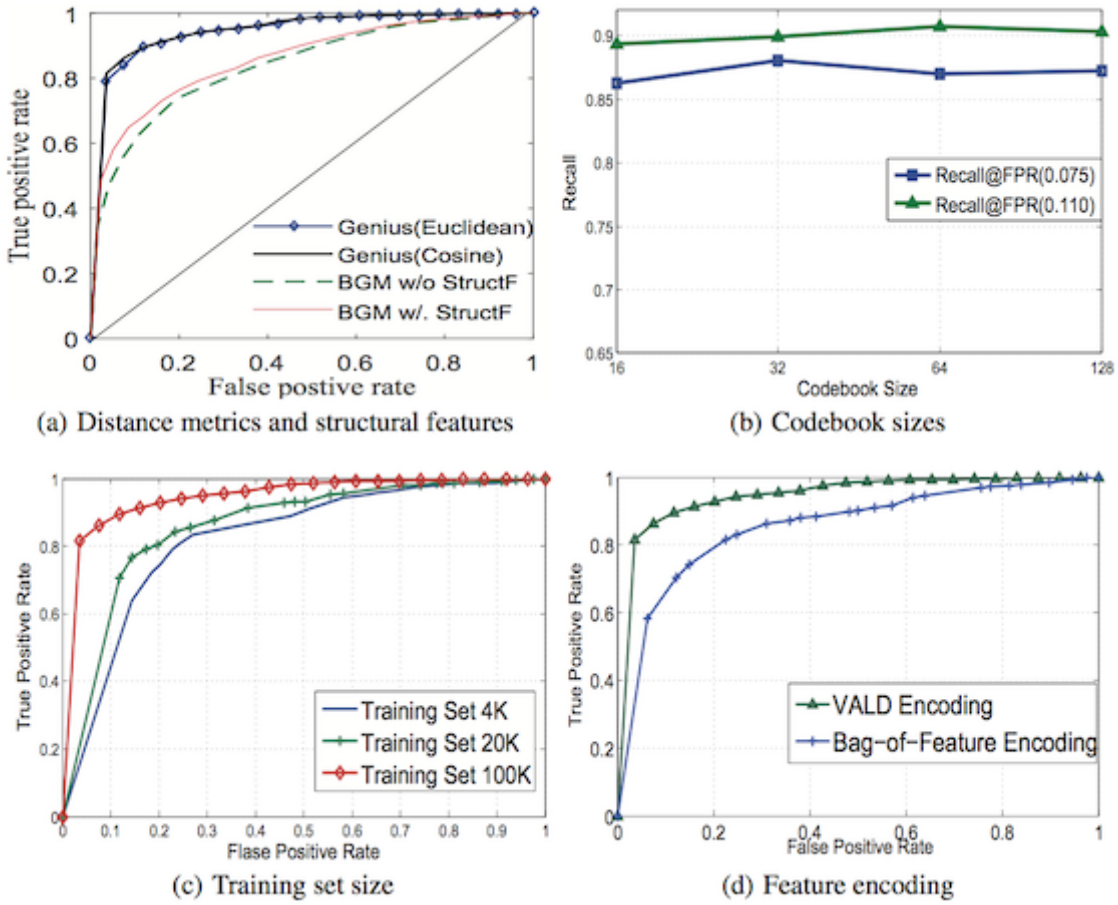
a) Recall rates across different threshold K　　b) ROC curves for different approaches

- Genius ranks 27% functions at top 1, whereas discovRe is 0.5%

Firmware Image	Binaries	Basic Blocks	Preparation Time in Minutes				
			Multi-MH	Multi-k-MH	discovRE	Genius	Centroid
DD-WRT r21676 (MIPS)	143 (142)	329,220	616	9,419	2.1	4.9	3.2
ReadyNAS v6.1.6 (ARM)	1,510 (1,463)	2,927,857	5,475	83,766	54.1	89.7	69.6



Parameter Studies



(a) Distance metrics and structural features

(b) Codebook sizes

(c) Training set size

(d) Feature encoding

Bug Search at Scale

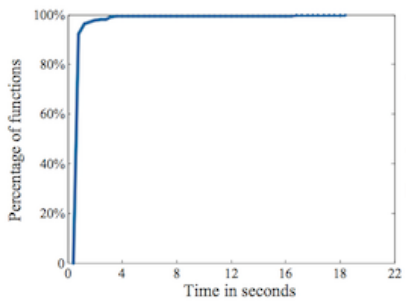


Figure 7: The CDF of preparation time over#1 million functions

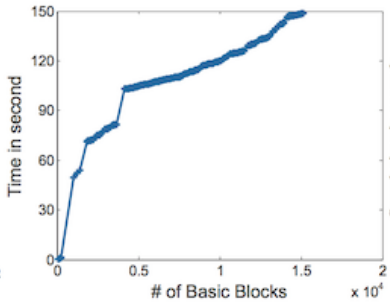


Figure 8: The preparation time cross different size of CFG

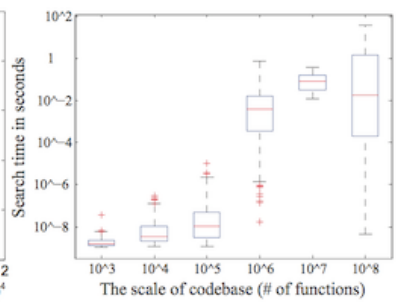


Figure 9: The search time crossscales of fimware codebases(# of funcitons

Case Study

DIR-810L_REVB_FIRMWARE_2.03B02			DIR-810L_REVB_FIRMWARE_2.02.B01		
CVE	**Patched**	**Vulnerability Type**	**CVE**	**Patched**	**Vulnerability Type**
CVE-2016-0703	No	Allows man-in-the-middle attack	CVE-2015-0206	No	Memory consumption
CVE-2015-1790	No	NULL pointer dereference	CVE-2014-0160	Yes	Heartbleed
CVE-2015-1791	Yes	Double free	CVE-2015-0289	No	NULL pointer dereference
CVE-2015-0289	No	NULL pointer dereference	CVE-2016-0797	No	Heap memory corruption
CVE-2014-8275	No	Missing sanitation check	CVE-2016-0798	No	Memory consumption
CVE-2015-0209	No	Use-after-free	CVE-2014-3513	No	Memory consumption
CVE-2015-3195	No	Mishandles errors	CVE-2014-3508	No	Information leakage
#	#	#	CVE-2015-0206	No	Memory consumption
#	#	#	CVE-2014-8275	No	Missing sanitation check