Stack Overflow Considered Harmful?
The Impact of Copy&Paste on Android Application Security

Felix Fischer, Konstantin Böttinger, Huang Xiao, Christian Stransky*, Yasemin Acar*, Michael Backes*, Sascha Fahl*
Fraunhofer Institute for Applied and Integrated Security; *CISPA, Saarland University

出处：S&P 17'

Paper: [Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security]

1.Introduction

越来越多的软件开发人员在Stack Overflow这样的在线技术平台讨论技术问题。尤其是缺乏编程经验的程序员会在这样的社区提问的时候，会得到一些技术指导和其他人回答问题的时候提供的代码片段。普遍认为，大量的程序员在开发软件的时候，经常会从这样的平台上复制代码。所以作者认为这样的行为会导致一些不安全的代码，虽然这些代码绝大多数情况下会解决问题，但还是由于其中许多代码片段是不安全的。那么就会存在这样一个传播周期：

`社区提供代码片段->开发者复制粘贴代码->用户运行了包含这些代码的程序->攻击者利用这些程序中的漏洞`

这篇论文中，作者就这个问题做了研究，发现Stack Overflow上的1161份不安全的代码被开发者复制粘贴进了130万个Android应用程序中。

Contributions

作者从Stack Overflow上提取了有可能引起安全问题（security-related）的Android代码片段，然后与Google Play上的应用进行对比：

1. 作者从StackOverflow上提取了4019个安全相关的Android代码片段
2. 使用一系列静态代码分析技术去检测了130万个应用程序中是否包含这些代码片段
3. 发现15.4%的应用包含这些代码片段，这些应用中97.9%都至少包含一个不安全的代码片段
4. 设计并实现了一个自动化的流水线处理分析代码片段和Android应用
5. 所有的数据在https://www.aisec.fraunhofer.de/stackoverflow 上可见

2.Processing Pipeline Architecture

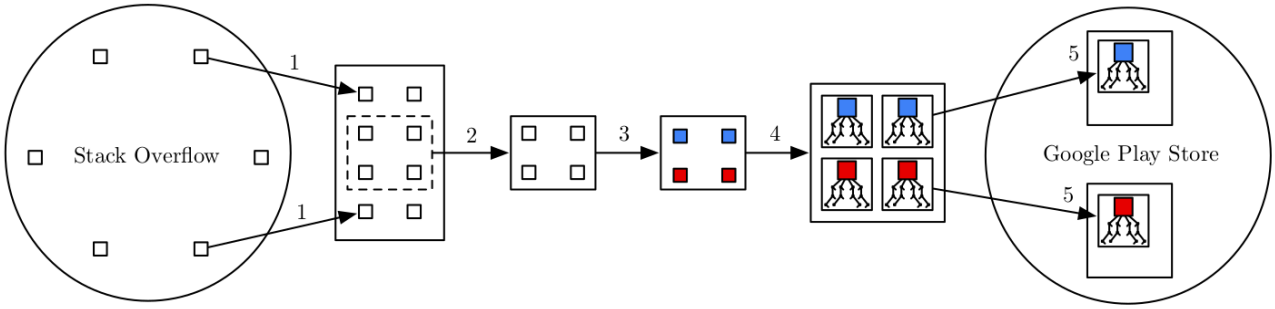

Fig. 1: Overall processing pipeline of code extraction (1), filtering (2), classification (3), program dependency graph generation (4), and clone detection (5).

所有的代码片段来源于Stack Overflow（左侧），并最终流入Google Play（右侧）。

作者首先抓取Stack Overflow并提取数据库中的每一个Android代码片段(1)。

然后根据制定的策略过滤那些安全相关的代码片段(2)。

定义安全和不安全的标签规则，并应用机器学习给每个代码片段贴上安全标签或不安全标签(3)。

接下来，作者生成每个标签代码片段(4)的抽象表示，以便在Google Play(5)中检测它们。

每一步都是完全自动化的，只有监督机器学习分类(3)的训练步骤需要手动标记训练集。

3.Code Extraction and Filtering

A. Security-related Code Snippets

作者从网上爬取了很多Android代码片段之后，定义了下列集合。如果代码中包含下列API的属于安全相关代码片段：

1. 密码学算法：Cryptography: Java Cryptography Architecture (JCA), Java Cryptography Extension (JCE)
2. 网络通信安全：Java Secure Socket Extension (JSSE), Java Generic Security Service (JGSS), Simple Authentication and Security Layer (SASL)
3. 公钥基础设施：X.509 and Certificate Revoca-tion Lists (CRL) in java.security.cert, Java certification path API, PKCS#11, OCSP
4. 认证和访问控制：Java Authentication and Authorization Service (JAAS)

除此之外，作者还认为使用下列安全lib的代码同样也属于安全相关代码片段：

1. BouncyCastle (BC)：开源并广泛使用的轻量级密码学lib

2. SpongyCastle (SC)：基于BC的重打包lib并多了其他功能
3. Apache的HttpClient中的TLS/SSL packages
4. keyczar,jasypt：简化开发人员对加密算法的使用

B. Finding Security-related Code Snippets on Stack Overflow

Stack Overflow上的代码片段被<code>标签包围，因此可以很容易地从附带的文本中分离出来并提取。 作者为了决定代码片段使用哪个API，我们需要代码片段中的代码元素的Fully Qualified Names（FQN）（即Java中的包名称）。 由于Partially Qualified Names（PQN）（即类和方法名称）不唯一，所以不同的API包含的类名（例如android.util.Base64，java.util.Base64）和方法名（例如java.security.Cipher.getInstance，java.security.Signature.getInstance）会使用相同的名称。 FQN允许我们区分非唯一的类和方法名称。

因为在Stack Overflow上发布的代码片段通常是不完整的或者错误的。 由于消除部分Java程序的歧义是一个不确定的问题，所以作者使用一个名为JavaBaker 的工具，以及之前提到的安全相关的lib的范围，可以确定代码片段中的安全相关的类型引用，方法调用或字段访问属于哪个security library。 这部分一共提取了3834个安全相关的代码片段，2474个在提问中，1360个在回答中。

4.Code Labeling

根据之前的工作，已经提取到了安全相关的代码片段，接下来就是对这些安全相关的代码片段进行分类。作者首先提供标签定义和分类规则，然后手动标记一小部分提取的代码片段当做训练样本去训练支持向量机(SVM)。之后再使用机器学习的方法去判断剩余的安全相关的代码片段是否安全。作者手动标记的1360个代码片段全部来自于Stack Overflow中的回答帖子。因为作者认为答案中的代码更可能被开发者复制和粘贴，因为它们是为了解决一个给定的问题给出的回答。 问题中的代码片段不包含在训练集中，因为它们可能会引入不可预测的噪音，从而影响训练结果分类器。

A. Security Labels

Secure:

- 包含最新和强对称密码算法的片段，足够强的密钥用于RSA或椭圆曲线密码，安全的随机数生成
- 包含不符合最佳实践的代码片段，但不会导致轻易被利用的漏洞
- 包含代码段的安全性取决于开发人员的输入，例如对称加密算法的密钥大小取决于开发者的配置而不是代码

Insecure:

- 使用过时的加密算法或静态初始化向量和对称加密的密钥，非对称加密的弱RSA密钥，不安全的随机数生成或不安全的SSL

B. Labeling Rules

这部分定义了代码片段中特定参数的安全性。作者提供了判断这些参数在代码中是安全还是不安全的度量标准。针对每一大类API的使用都列举了判断标准。

SSL/TLS

Parameter	Secure	Insecure
Hostname Verifier	browser compatible, strict	allow all hosts [17]
Trust Manager	default, secure pinning	trust all [2], bad pinning [18], [17], validity only
Version	>=TLSv1.1 [12]	<TLSv1.1 [19], [12], [20], [21]
Cipher Suite	DHE_RSA, ECDHE AES>=128, GCM SHA>=256 [12]	RC4,3DES, AES-CBC MD5, MD2 [12], [22]
OnReceived-SSLError	cancel	proceed

TABLE II: Secure and insecure TLS parameters.

Symmetric Cryptography

Parameter	Secure	Insecure
Cipher/Mode	AES/GCM [12] AES/CFB [12] AES/CBC*	RC2 [23], RC4 [24], DES [23], 3DES [25], AES/ECB [3], AES/CBC** [22] Blowfish [26], [27]
Key	provider generated	static [3], bad derivation [3]
Initialization Vector (IV)	provider generated	zeroed [3], static [3], bad derivation [3],
Password Based Encryption (PBE)	>=1k iterations [13], >=64-bit salt [13], non-static salt [13]	<1k iterations [13], <64-bit salt [13] static salt [3]

TABLE III: Secure and insecure symmetric cryptography parameters.

Asymmetric Cryptography

Parameter	Secure	Insecure
Cipher/Mode	RSA RSA/ECB RSA/None	
Padding	PKCS1*, PKCS8, OAEPWithSHA-256 AndMGF1Padding,	PKCS1**
Key	RSA >= 2048 bit ECC >= 224 bit	RSA < 2048 bit [14] ECC < 224 bit [15]

TABLE IV: Secure and insecure asymmetric cryptography parameters.

One Way Hash Functions

Parameter	Secure	Insecure
PBKDF	[PBKDF2](Hmac) >=SHA224 [29]	[PBKDF2](Hmac) MD2, MD5 [29]
Digital Signature	>SHA1	MD2, MD5
Credentials	>SHA1	MD2, MD5

TABLE V: Secure and insecure hash function parameters

Random Number Generation

Parameter	Secure	Insecure
Type	SecureRandom	Random
Seeding	nextBytes, nextBytes->setSeed	setSeed->nextBytes, setSeed with static values [3]

5.Code Classification

一共有3834个安全相关的代码片段，2474个在提问中，1360个在回答中。

这部分其实就是用从回答中提取的1360个已经分好安全/不安全的代码片段当做机器学习的样本训练SVM（支持向量机），然后去识别剩下的从提问中提取的2474个代码片段。

其中有1161 (30.28%) 个不安全的代码片段，有2673 (69.72%)个安全的代码片段。

6.PDG Generation And Code Detection

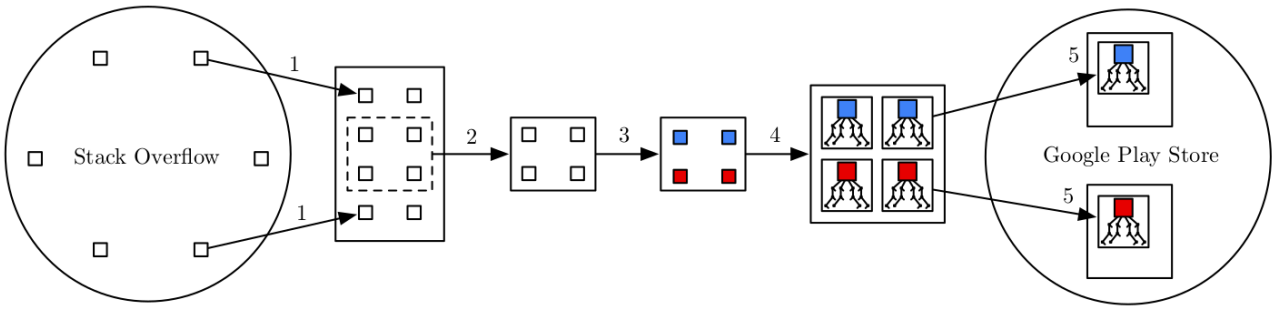

Fig. 1: Overall processing pipeline of code extraction (1), filtering (2), classification (3), program dependency graph generation (4), and clone detection (5).

经过之前的的流程，作者现在已经得到了提取之后又经过安全/非安全判定的Android代码片段(1-3)，接下来就是通过这些代码去识别Google Play中的应用是否包含这些代码。

因为代码片段是Java代码，而Android应用程序是DEX二进制文件，所以作者把这两类Code都转化成相同的IR表示，然后再进行相似性检验。

A. Code Snippet Compiling

因为编译的过程需要完整的代码，但是Stack Overflow上的代码大多都只是代码片段。这些代码缺失了很多class信息和外部lib名称，没办法像开发Android应用一样直接编译。为了克服这个问题，作者首先使用了Partial Program Analysis（PPA）。PPA专门设计用于从Java程序的源代码创建完整的类型化抽象语法树（AST）。

同样为了能够把二进制文件（DEX）转化成相同的IR，作者首先使用enjarify把应用中的DEX文件转化成JVM Bytecode，然后再通过WALA生成IR。

对于Stack Overflow中的代码片段，作者通过整合PPA来修改WALA，从而能够生成相同的IR。最后使用这种方法作者成功处理了1293(85.2%)个answer中的代码片段和1668(66.6%)个question中的代码片段。其他失败的代码片段是因为语法错误而不能被WALA识别。

B. Code Snippets in Apps

作者根据修改了的Crussel等人通过代码生成的程序依赖图（PDG）来做代码相似性检验。原先的Crussel等人为每种方法创建PDG，并将PDG中独立子块的基本代码特征用于重用检测。而语义块是指一个方法的PDG中可能包含的多个独立子块。代码相似性的判定就取决于所比较的代码中相似的语义块数量。这种方式对于控制流程被简单修改和非恶意代码的插入有很强的稳健型。

作者认为，这种方式不适合本文中大型Android应用程序中代码片段的相似性检测。因为从Stack Overflow上提取的代码片段非常小，因此会导致较小的PDG，不同的代码可能会导致相同的PDG。所以作者使用了更严格的方法，增加了比较语义块的常量和方法名称。因为常量是初始化Android安全相关API的关键。另外，作者还比较语义块中的属于预先定义的安全lib集合API方法名称。这使我们能够区分与代码的安全相关的部分。

对于PDG的生成，作者使用Crussel提供的嵌入算法为每个语义块分配一个语义向量，语义向量中存储关于节点和边缘的信息。WALA IR提供的每种指令类型在向量中都有两个对应的字段。一个字段存储节点，另一个存储边缘信息。语义块中的每种指令类型（例如，invokevirtual，getfield，new或return）的节点数存储在向量中的指令类型的相关节点字段中。

判断两个语法向量是否相似可以使用Jaccard相似度：用两个集合中不同元素占所有元素的比例来衡量两个集合的相似度。

$$J_s(X,Y) = \frac{\sum_i min(X_i, Y_i)}{\sum_i max(X_i, Y_i)}$$

作者最后使用修改之后的Jaccard相似度来确定X语法向量（代码片段）是否包含在Y（App）中

$$\frac{|X \wedge Y|}{|X|}$$

当且仅当以下条件适用于片段中的每个方法时，

- 对于所有给定的语义块，我们可以找到语义块满足Jaccard相似性，并包含在一个单一的方法包含在给定应用程序的调用图中。
- 方法名称集完全包含在相同的内容中方法。
- 常量集完全包含在同一个方法中。
- 它们属于同一个（嵌套的）类。

7.Evaluation

A. Evaluation of Code Extraction and Filtering

一共从Stack Overflow上的Android标签下的提问中提取了2474个不同的安全相关的代码片段，从回答中提取了1360个安全相关的代码片段。

java.security相关的有70.7%（java.security.KeyStore – 44.9%, java.security.MessageDigest – 30.4%）

Android’s cryptographic相关的有31.9%

javax.net.ssl相关的有28.9%

jasypt and keyzcar相关的有0.3%

Namespace	Snippets	Namespace	Snippets
javax.crypto	1,286	android.security	5
Cipher	1,088	com.sun.security	5
KeyGenerator	207	gnu.crypto	47
spec.SecretKeySpec	701	java.security	2,841
spec.PBEKeySpec	69	javax.security	44
spec.DESedeKeySpec	6	javax.xml.crypto	3
spec.DESKeySpec	21	org.bouncycastle	48
spec.IvParameterSpec	338	org.spongycastle	44
spec.RC2ParameterSpec	1	org.jasypt	11
Mac	85	org.apache.http.conn.ssl	241
Sealed	8	AllowAllHostnameVerifier	184
javax.net.ssl	1,166	StrictHostnameVerifier	51
TrustManager	545	BrowserCompatHostnameVerifier	8
HostnameVerifier	200	TrustSelfSignedStrategy	1
SSLSocket	533	SSLSocketFactory	105
org.keyczar	2		

TABLE VII: Snippet counts per library.

B. Evaluation of Code Classification

一共提取了3834个安全相关的代码片段，其中有1161 (30.28%) 个不安全的代码片段，有2673 (69.72%)个安全的代码片段。

C. Evaluation of Code Detection

一共检测了1,305,820个免费下载的APP，其中有200,672 (15.4%)抄了Stack Overflow上的代码。绝大多数的应用程序包含一个不安全的代码片段：196,403（15%）个应用程序至少包含一个。作者发现包含安全片段的应用为506,922（38.82%）个。

1. TLS：通过复制和粘贴的不安全代码段，183,268（14.03%）个应用程序受到不安全的TLS处理的影响。 所有应用程序中只有441（0.03%）包含与TLS相关的安全代码段。 对于182,659（13.98%）个应用程序中带有不安全TLS片段的大多数应用程序，它们的代码片段与Stack Overflow上的问题代码片段相匹配。
2. 对称密码：在21,239（1.62%）的应用程序中，对称密码的不安全代码片段数量为第二大。 19,452（1.48%）的与对称加密相关的代码片段的应用程序包含了一个安全片段。此类之中复制粘贴的不安全片段数量最多的是ECB模式下的AES（可在18,000个应用中找到）。
3. 非对称加密：作者发现只有114（0.01%）个应用程序包含与非对称加密相关的不安全代码片段，698（0.05%）个应用程序包含安全非对称加密相关代码段。

4. 安全随机数生成：8,228（0.63%）个应用包含与随机数生成相关的不安全代码段，而4,100（0.31%）个应用包含安全片段。

5. 散列：对于散列函数，包含来自Stack Overflow的代码片段大多数都是安全的：有4012（0.3%）是安全的，14个应用程序不安全。

6. 签名：15个应用程序包含与安全签名相关的代码段，而在此类别的应用程序中未找到不安全的代码段。 所有这些代码片段都可以在Stack Overflow的问题中找到。

7. 安全无关代码：在应用程序中检测到的某些代码段无法分配到上述其中任何一个类别，因为它们与第三部分中描述的安全分类无关。 498,046（38.1%）个应用程序包含一个与安全无关的片段，因此被分类为安全。

8. 敏感应用类别：使用不安全的复制和粘贴代码片段的敏感应用数量最多的有14,944个商业应用，4,707个购物应用和4,243个金融应用。

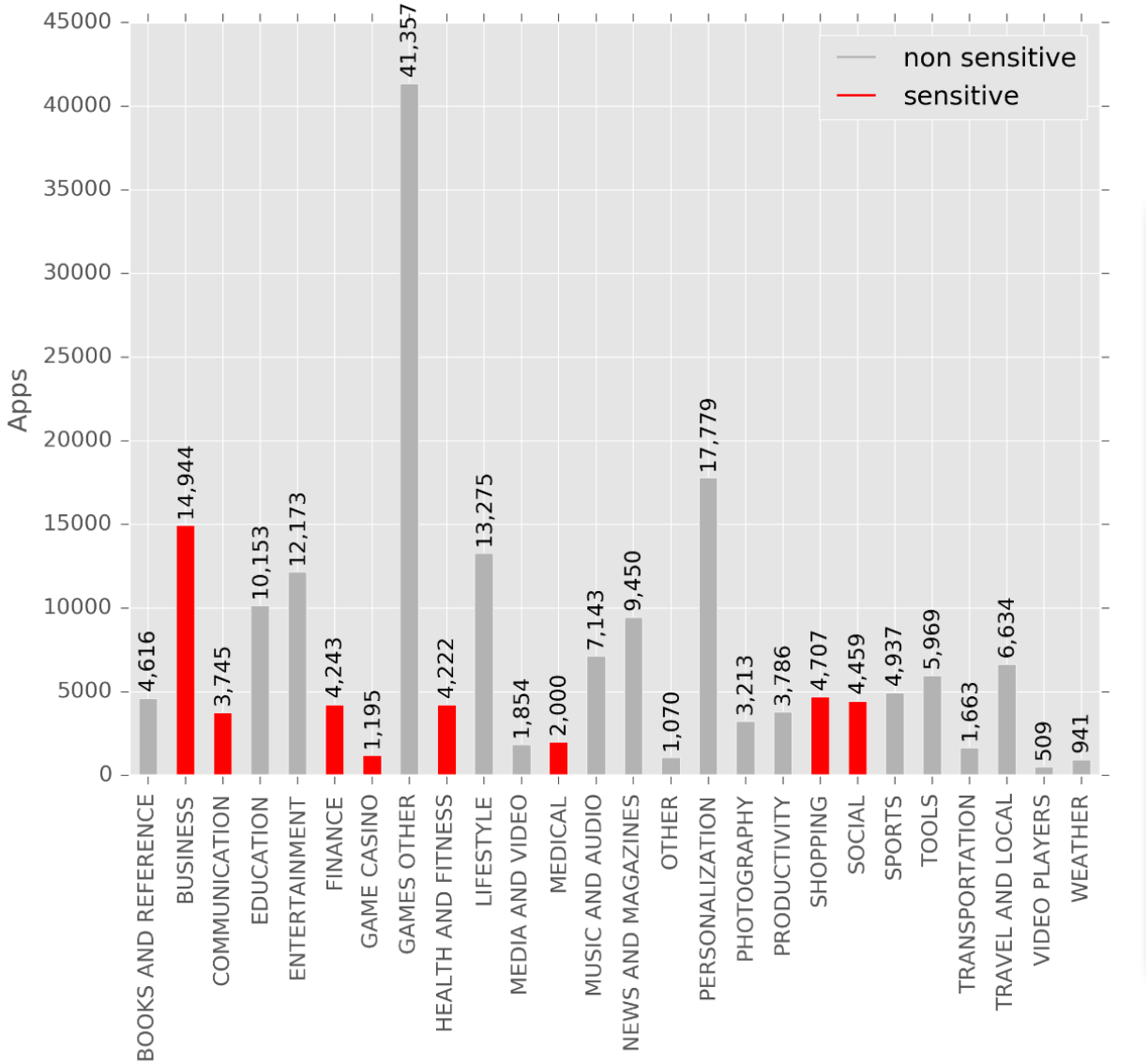

Fig. 5: Distributions of insecure snippets found in Apps

9. 下载次数：作者还提供了包含不安全片段的应用的下载次数

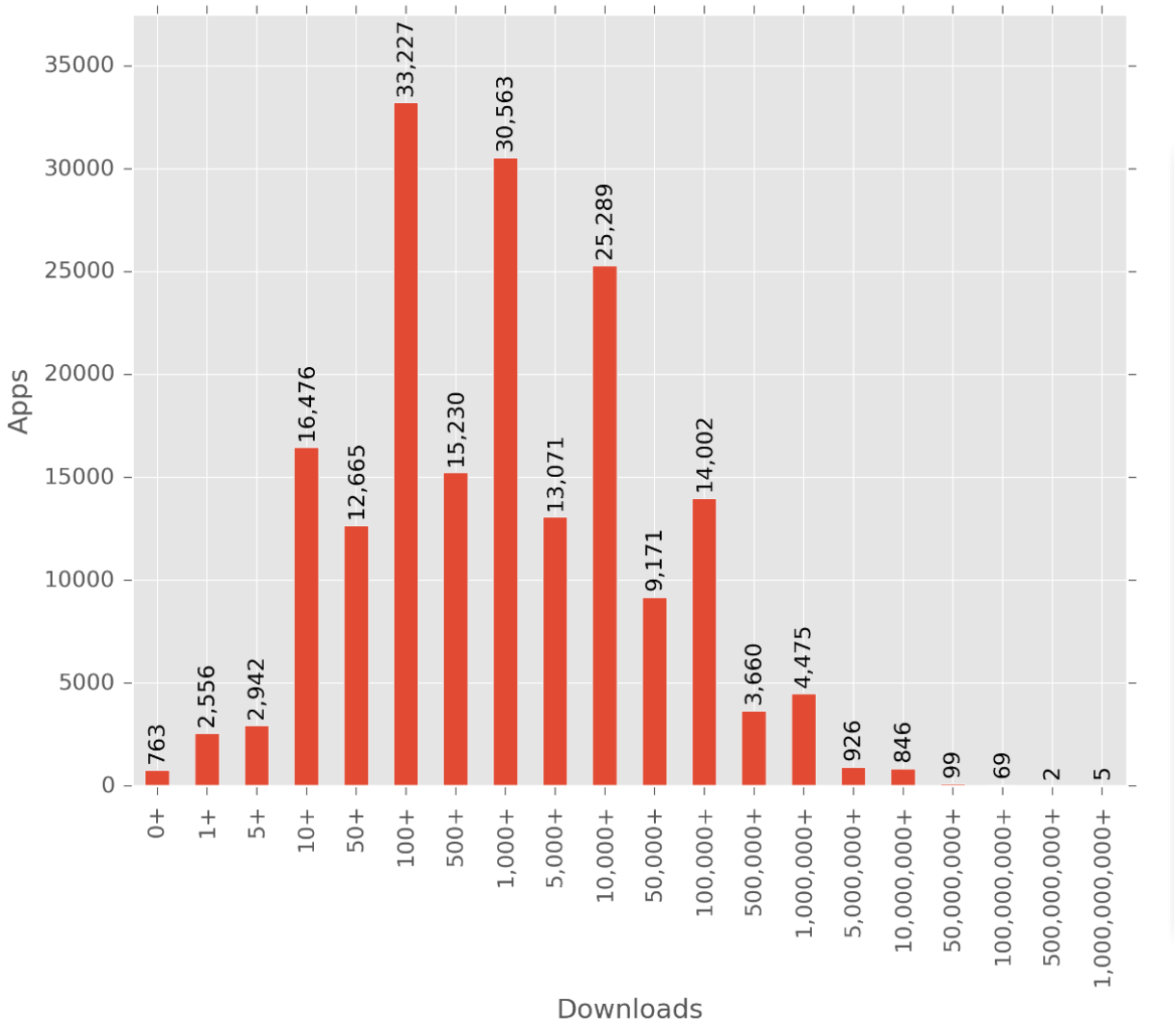

Fig. 6: Download counts for Apps with insecure snippets