检测假冒小程序：微信案例研究

Detect Counterfeit Mini-apps: A Case Study on WeChat

Xuanfa Deng, Miao Zhang, Xinqi Dong, and Xin Hu. 2024. In Proceedings of the ACM Workshop on Secure and Trustworthy Superapps (SaTS '24), October 14–18, 2024, Salt Lake City, UT, USA. ACM, New York, NY, USA, 10 pages. https: //doi.org/10.1145/3689941.3695773

MiniCAT：解释并检测小程序中的跨页面请求伪造漏洞

MiniCAT: Understanding and Detecting Cross-Page Request Forgery Vulnerabilities in Mini-Programs.

Zidong Zhang, Qinsheng Hou, Lingyun Ying, Wenrui Diao, Yacong Gu, Rui Li, Shanqing Guo, and Haixin Duan. 2024. In Proceedings of the 2024 ACM SIGSAC Conference on Computer and Com munications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA. ACM, NewYork, NY, USA, 17 pages. https://doi.org/10.1145/365864 4.3670294

小程序简介

App-in-app is a new and trending mobile computing paradigm in which native app-like software modules, called sub-apps, are hosted by popular mobile apps such as Wechat, Baidu, TikTok and Chrome, to enrich the host app's functionalities and to form an "all-in-one app" ecosystem. Sub-apps access system resources through the host, and their functionalities come close to regular mobile apps (taking photos, recording voices, banking, shopping, etc.).

https://dl.acm.org/doi/10.1145/3372297.3417255

子应用程序即小程序是一种新的移动计算模式，也是一种趋势。由微信、百度、嘀嗒和 Chrome 浏览器等流行的移动应用程序托管，以丰富主应用程序的功能，形成 “一体化应用程序 ”生态系统。小程序通过主机访问系统资源，其功能接近于普通手机应用程序（拍照、录音、银行、购物等）。

小程序简介

特点

轻量快捷

多场景融合

功能多样化

高效传播

性能限制

现状

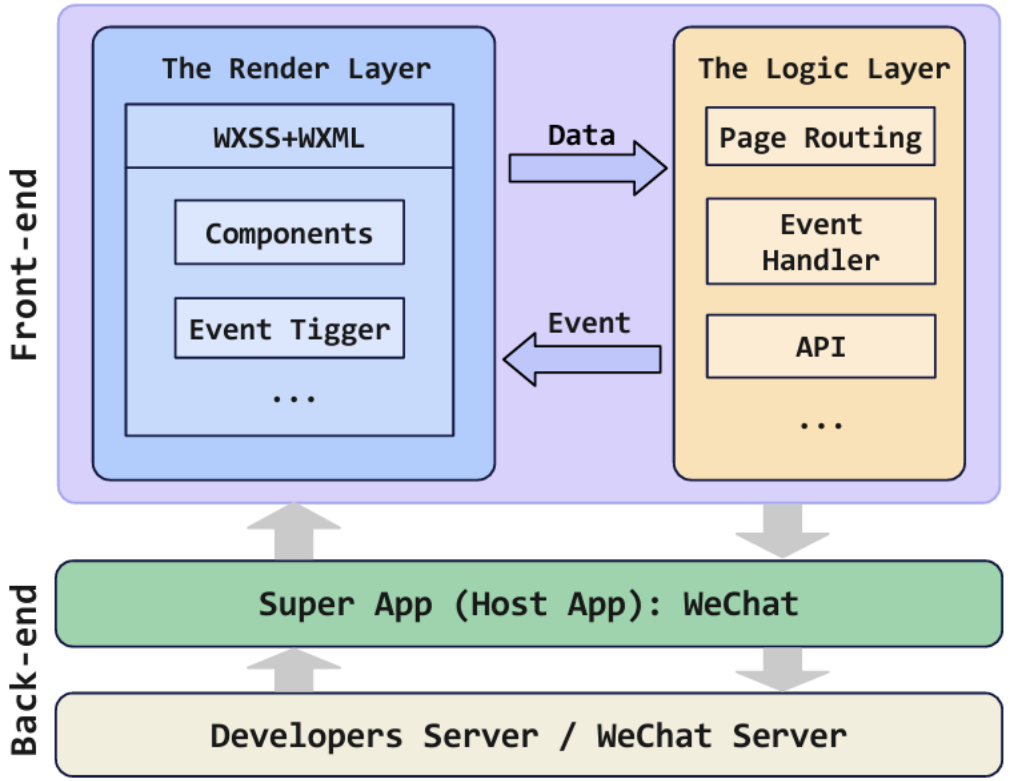
用户规模庞大

商业化成熟

功能多样化

政府与企业广泛采用

小程序简介



微信小程序框架图

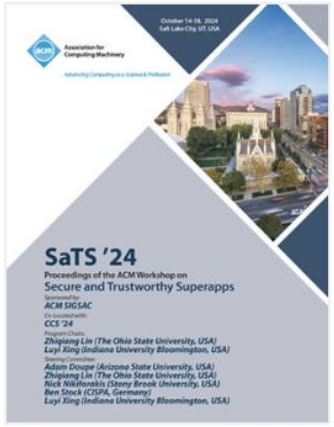
https://developers.weixin.qq.com/miniprogram/dev/framework/

检测假冒小程序：微信案例研究

Detect Counterfeit Mini-apps: A Case Study on WeChat

Xuanfa Deng, Miao Zhang, Xinqi Dong, and Xin Hu. 2024. In Proceedings of the ACM Workshop on Secure and Trustworthy Superapps (SaTS '24), October 14–18, 2024, Salt Lake City, UT, USA. ACM, New York, NY, USA, 10 pages. https: //doi.org/10.1145/3689941.3695773

Published In



SaTS '24: Proceedings of the ACM Workshop on Secure and Trustworthy Superapps

November 2024 | 28 pages

ISBN: 9798400712371

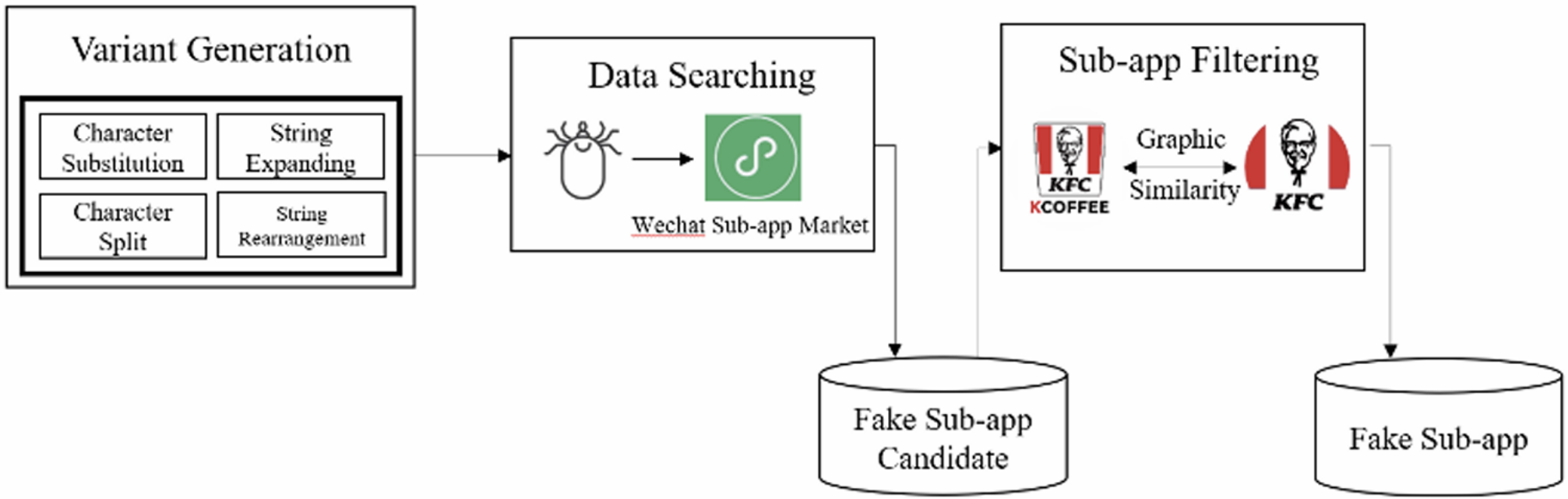
DOI: 10.1145/3689941

Program Chairs: Zhiqiang Lin, Luyi Xing

什么是假冒小程序?

(a) Legitimate "KFC"

MiniWukong



检测流程图

对抗示例生成

1、相似字替换

$$S(a,b) = 1 - \frac{\text{EditDistance}\ (a,b)}{\text{MaxLen}\ (a,b)} \tag{1}$$

$$M_{K,T} = \alpha S\left(K_o, T_o\right) + \beta S\left(K_p, T_p\right) + \gamma S\left(K_c, T_c\right) + \delta S\left(K_f, T_f\right) \tag{2}$$

对抗示例生成

2、拼音替换
用相应的拼音替换原始小程序名称中的原始汉字

3、序列交换
打乱原小程序名称中的汉字顺序来创建一个新的小程序名称

4、同义替换
如：“外卖”（取出）、“甄选”（选定）、“在线"（联机）等

5、左右拆分
如果单独检查一个字符的左部或右部，每个部分都可以组成一个独立的字符
如：“卧“可以拆分成”臣”和“卜“两个独立的字符

数据收集

一、利用专门设计的网络爬虫搜索微信小程序市场中生成的假冒小程序名称

二、采用精确比对法。我们将生成的仿冒小程序名称与搜索结果中的小程序名称进行精确匹配比较。只有当小程序名称与我们生成的变体完全匹配时，我们才会将相应的应用标记为假冒小程序候选。

数据过滤

美团酒店

美团机票

美团贷款

所有者相同

DiDi

DiDa

图标相似

有效性证明

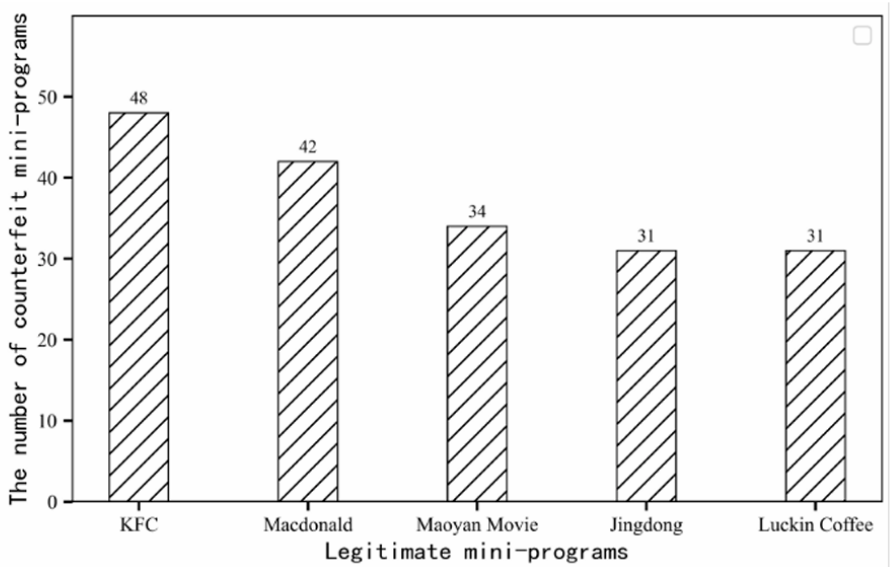
Attacked Models	Attack Methods	Classification Accuracy	Model Perplexity	Visual Similarity	Modification Ratio
LSTM	Before	0.952	22.9	/	/
	WordHanding	0.747	30.1	0.874	0.136
	CWordAttack	0.634	37.5	0.891	0.152
	Our(0.25,0.25,0.25,0.25)	0.655	35.7	0.904	0.167
	Our(0.4,0.1,0.25,0.25)	0.627	37.2	0.909	0.152
	Our(0.1,0.4,0.25,0.25)	0.679	33.2	0.894	0.166
	Our(0.4,0.3,0.15,0.15)	0.620	37.7	0.908	0.163
BERT	Before	0.961	22.9	/	/
	WordHanding	0.791	29.2	0.870	0.145
	CWordAttack	0.773	36.8	0.842	0.177
	Our(0.25,0.25,0.25,0.25)	0.699	35.2	0.892	0.172
	Our(0.4,0.1,0.25,0.25)	0.688	36.4	0.903	0.165
	Our(0.1,0.4,0.25,0.25)	0.721	33.9	0.884	0.176
	Our(0.4,0.3,0.15,0.15)	0.669	37.1	0.900	0.169
Transformer	Before	0.932	22.9	/	/
	WordHanding	0.781	29.8	0.873	0.144
	CWordAttack	0.762	36.6	0.852	0.175
	Our(0.25,0.25,0.25,0.25)	0.672	36.2	0.904	0.169
	Our(0.4,0.1,0.25,0.25)	0.653	37.6	0.913	0.161
	Our(0.1,0.4,0.25,0.25)	0.698	35.8	0.898	0.173
	Our(0.4,0.3,0.15,0.15)	0.642	38.6	0.909	0.165
TextCNN	Before	0.923	22.9	/	/
	WordHanding	0.725	29.1	0.901	0.135
	CWordAttack	0.634	39.4	0.891	0.175
	Our(0.25,0.25,0.25,0.25)	0.615	34.2	0.905	0.155
	Our(0.4,0.1,0.25,0.25)	0.588	35.6	0.912	0.148
	Our(0.1,0.4,0.25,0.25)	0.629	33.2	0.901	0.165
	Our(0.4,0.3,0.15,0.15)	0.581	36.3	0.910	0.152

评估

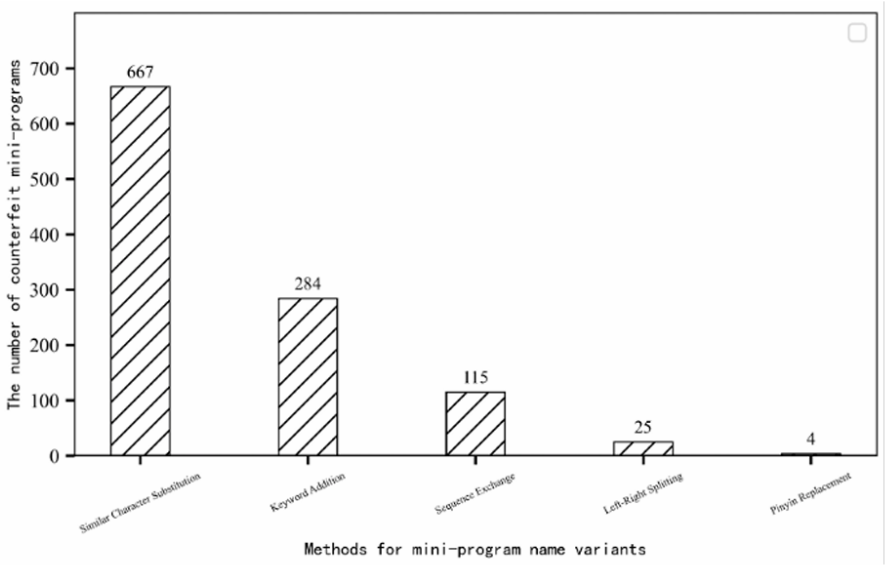
问题1：假冒小程序的分布情况如何？生成假冒小程序名称的最常见技术是什么？

问题2：假冒小程序是否更有可能以流行小程序为目标？

评估结果



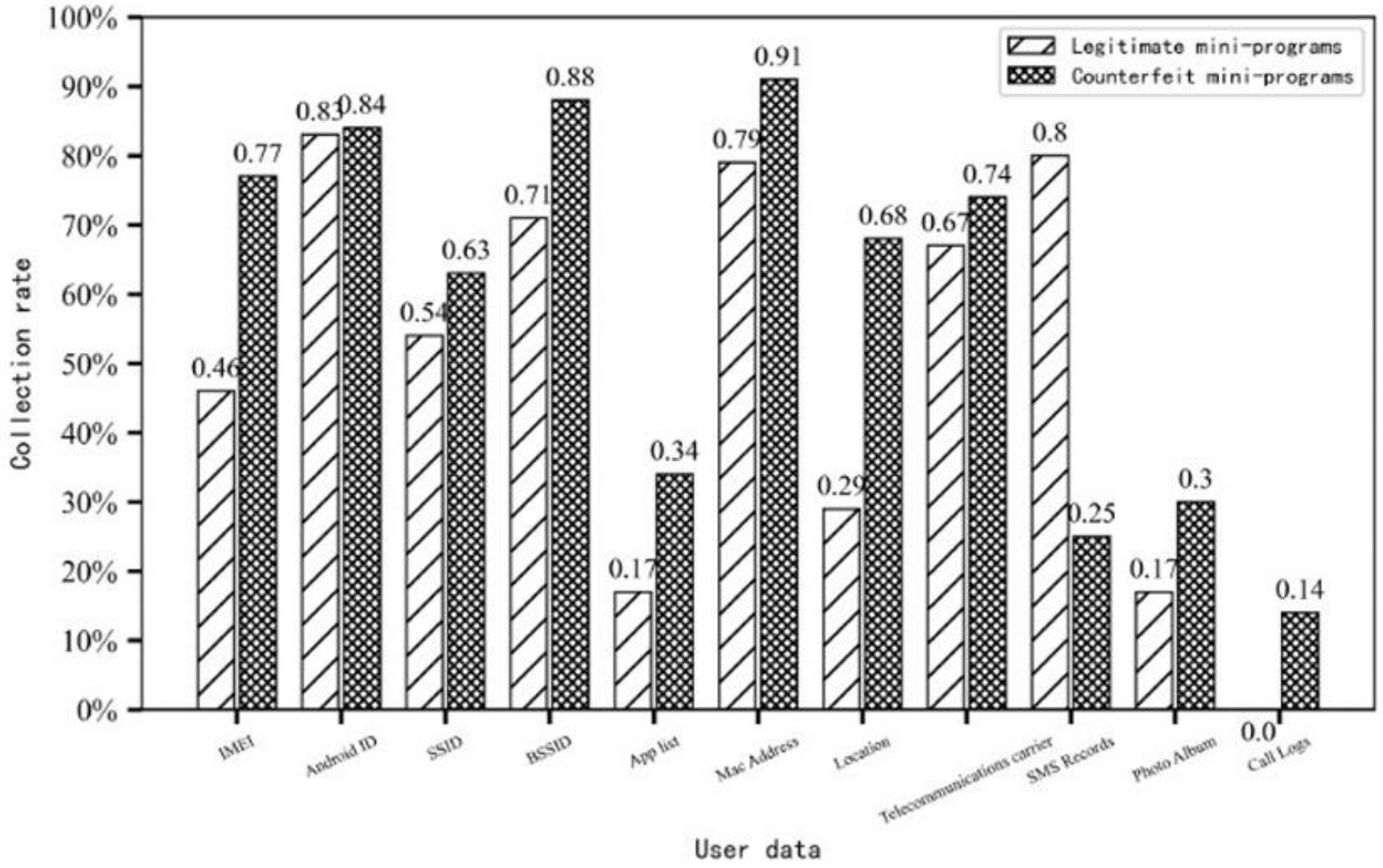
假冒小程序数量最多的小程序前五



生成假冒小程序技术前五

假冒小程序的影响

窃取用户数据



假冒小程序的影响

假冒特许经营

销售假冒产品

研究的局限性

首先，基于研究动机，采用的同音字生成技术存在假阳性和假阴性问题。它错误地生成了同音字符，并遗漏了一些相关字符，对随后生成小程序名称变体产生了负面影响

其次，在生成小程序名称变体时，采用了五种方法。在实际研究过程中存在一些伪造的小程序名称并非由这五种方法中的任何一种生成。为了解决这个问题，可将小程序变体生成模块化，以便于添加新的变体生成方法

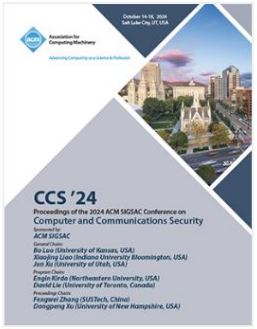
最后，研究重点只放在在食品配送、生活服务、电子商务和公用事业等类别中，只选取了排名前50位的迷你应用。在今后的研究中，计划将研究扩展到更多类别，如游戏等

MiniCAT：解释并检测小程序中的跨页面请求伪造漏洞

MiniCAT: Understanding and Detecting Cross-Page Request Forgery Vulnerabilities in Mini-Programs

Zidong Zhang, Qinsheng Hou, Lingyun Ying, Wenrui Diao, Yacong Gu, Rui Li, Shanqing Guo, and Haixin Duan. 2024. In Proceedings of the 2024 ACM SIGSAC Conference on Computer and Com munications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA. ACM, NewYork, NY, USA, 17 pages. https://doi.org/10.1145/365864 4.3670294

Published In



CCS '24: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security

December 2024 | 5188 pages

ISBN: 9798400706363

DOI: 10.1145/3658644

General Chairs: Bo Luo, Xiaojing Liao, Jun Xu,

Program Chairs: Engin Kirda, David Lie (Less)

路由实施

Username Alice

Password •••••••

登录

```
<!-- page/index/index.wxml -->
<button bindtap="formSubmit" class="
    loginBtn" type="primary" form-type='
    submit'>Login</button>
```

② 触发事件

```
/* page/index/index.js */
formSubmit: function(e) {
var username = e.detail.value.username;
var password = e.detail.value.password;
wx.request({
    url:"http://../login",
    ...,
    success: function (res) {
        wx.navigateTo(
            {url:'/pages/user/index/index?
  username='+res.data.username})}})
    }
```

①

③

page/index/index

page/user/index/index

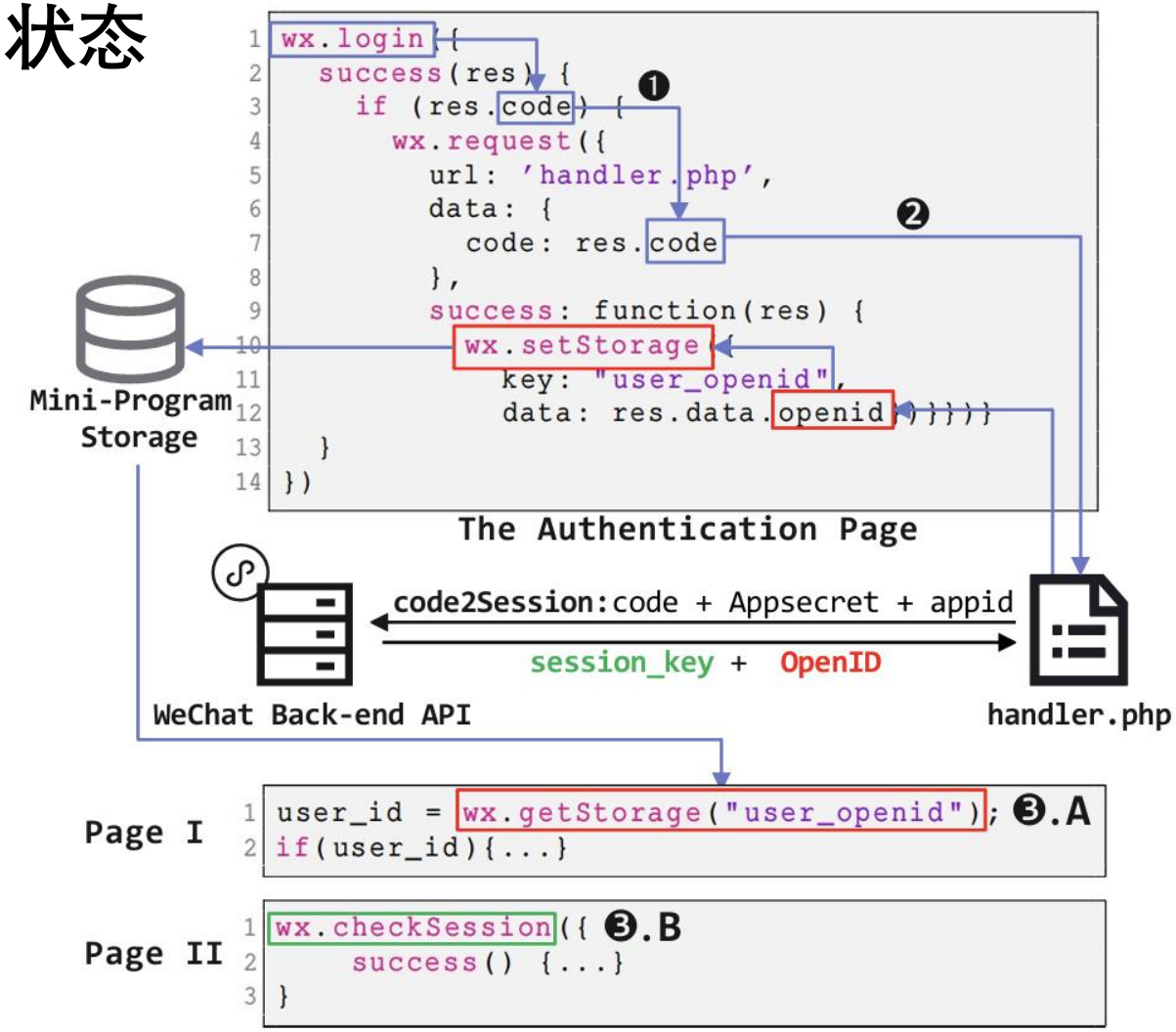
wx.navigateTo

/page/user/index/index?username=Alice

wx.navigateTo
wx.redirectTo
wx.reLaunch

事件通信和路由示例

用户状态



身份验证和用户状态管理示例

- 步骤1：小程序启动 wx.login，，提示微信客户端生成一个临时凭证
- 步骤2：开发人员需要在自己的服务器中设置一个处理程序页面 handler.php。小程序的前端会发送然后页面将通过后端 API code2Session与微信服务器通信，获取用户唯一的 OpenID 和 session_key。此时，session_key-OpenID 就构成了用户状态的一对凭证

有两个选项可以检查特定页面上的用户状态：

- A：使用 OpenID设计自定义用户状态，并通过 wx.setStorage 存储在本地。需要验证时，开发人员可使用 wX. getstorage 获取 OpenLD
- B：使用 wx.checkSession 验证由wx.login 生成用户状态

分享与转发

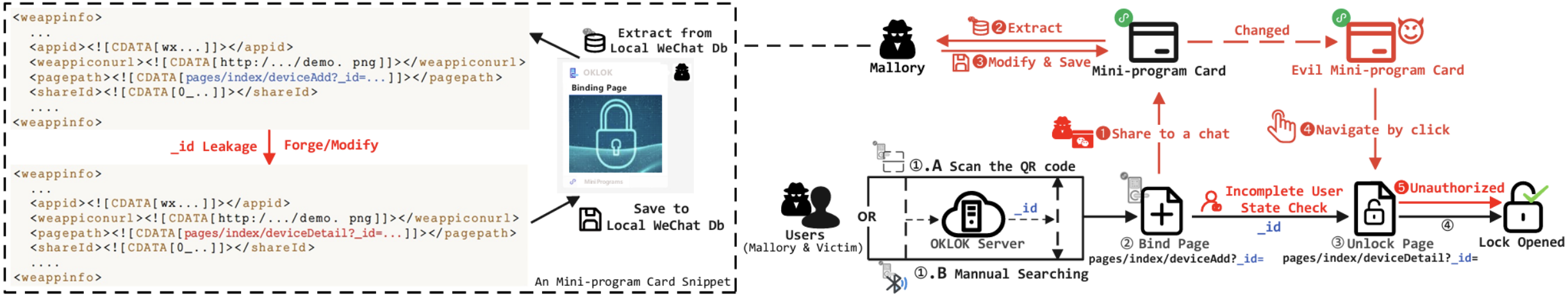
javascript

```
Page({
    onShareAppMessage: function () {
        return {
            title: '分享的标题',
            path: '分享的页面路径',
            imageUrl: '分享所用的图片路径'
        };
    }
});
```

MiniCPRF

本文发现了一个新漏洞，该漏洞源于迷你程序中页面路由和用户状态管理设计的缺陷，包括导航和参数传输对 URL 模式的依赖、未加密的参数通信以及用户状态保存的不一致性。特别地，利用小程序的共享和转发功能以及不安全的本地存储，攻击者可以在小程序内操纵指定用于页面路由的URL。这种操纵可以控制目标路由页面和传输参数。这一特定漏洞被称为 MiniCPRF（小程序中的跨页面请求伪造）

攻击案例



OKLOK的工作流程和攻击路径

攻击步骤

- **步骤一**：攻击者识别可能存在漏洞的网页，并通过转发从网页的路由 URL 获取参数

- **步骤二**：攻击者通过修改页面路由 URL来修改小程序卡或生成新的小程序卡

- **步骤三**：当受害者（或在某些漏洞利用案例中的攻击者）：一点击制作好的小程序卡，攻击者就可以进行敏感操作

根本原因

•微信小程序的路由只允许开发者在 URL 模式中传达参数。如果开发者缺乏安全意识，例如在页面路由 URL 中以明文传输敏感参数，这可能会导致信息泄漏和相关的安全风险

• 微信小程序缺乏统一、整体的用户状态实施，现有的用户状态安全取决于开发人员的意识。在上述案例中，敏感页面没有实施用户状态验证攻击成功的关键因素

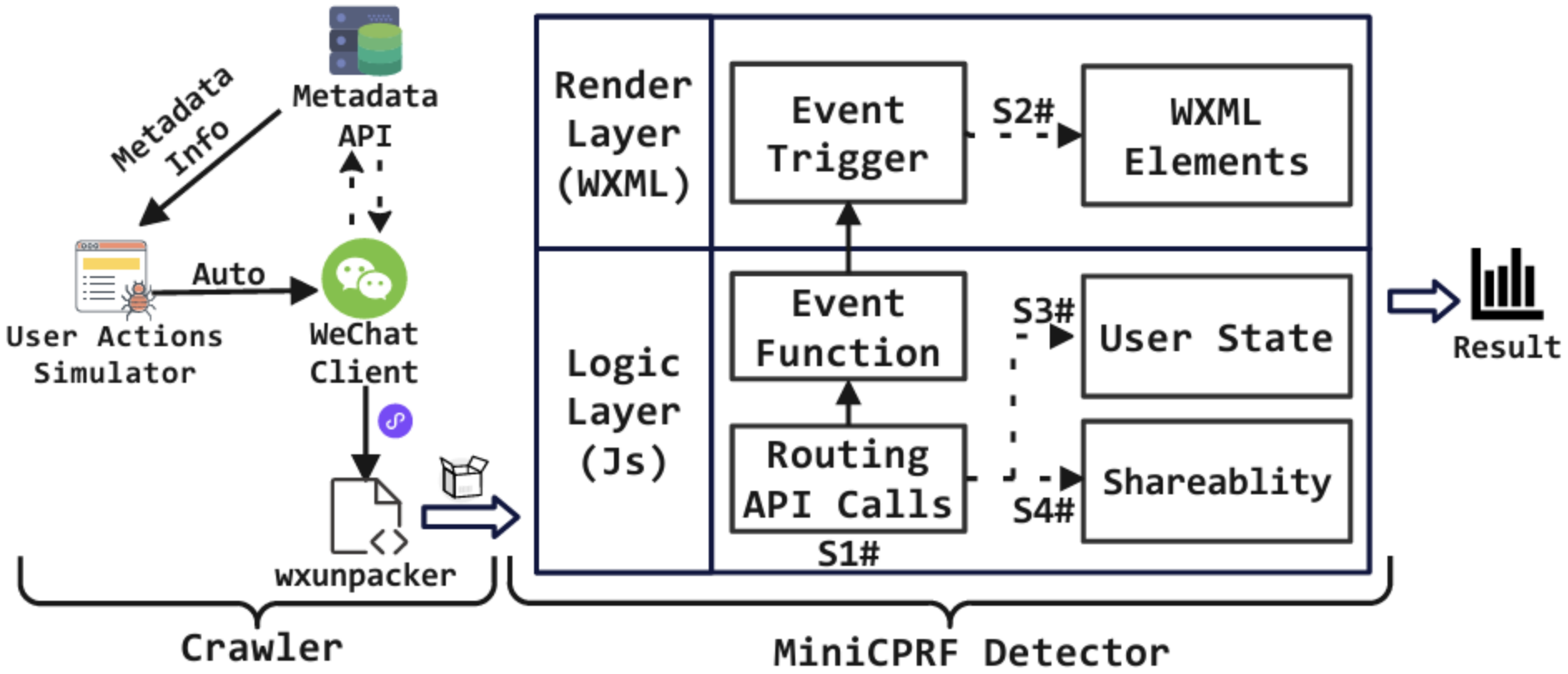
• 微信缺乏完整性检查，因此可以转发修改和伪造恶意微型程序卡，而这些这些卡以 XML 纯文本形式存储在本地存储器中，攻击者轻松查看和修改这些卡中的页面路由参数

后果

- **未经授权的操作**：如果开发人员将敏感操作小程序页面路由 URL 中的相关参数，在恶意软件可以篡改或伪造这些 URL，以执行未经授权的操作。此外，通过共享或生成小程序代码，这些恶意小程序卡还可被他人使用，从而造成重大安全威胁

- **敏感数据泄露**：攻击过程中可能会泄露敏感数据。在上述情况下，锁的_id 应该是不可见的。小程序用户。然而，攻击者可以通过 MiniCPRF 漏洞获取_jd。如果_id 被用于小程序的其他页面或涉及其他敏感操作，它就会增加攻击面，容易被进一步利用

MiniCAT


Metadata Info
Metadata API
Auto
User Actions Simulator
WeChat Client
wxunpacker
Crawler
Render Layer (WXML)
Logic Layer (Js)
Event Trigger
S2#
WXML Elements
Event Function
S3#
User State
Routing API Calls
S4#
Shareablity
S1#
Result
MiniCPRF Detector

小程序爬虫

利用自然语言处理（NLP）技术来构建关键字词典。与MiniCrawler 中提到的方法类似，小程序爬虫也是利用微信Windows 客户端中的元数据来搜索小程序。

https://doi.org/10.1145/3410220.3460106

user_file/AppLet/AppID

此外，小程序爬虫利用小程序元数据生成关键词字典。元数据可通过元数据API使用用户配置文件目录中的小程序 AppID 获取。此外，爬虫还从用户配置文件目录中检索小程序包，并使用wxapp Unpacker 将其解压缩为源代码，以便进一步分析，作为MiniCPRF 检测器的输入。

https://github.com/system-cpu/wxappUnpacker

CPRF检测器检测步骤

- 步骤一：识别调用页面路由API
- 步骤二：构建攻击路径
- 步骤三：检查用户状态
- 步骤四：共享性检查

```
/* deviceAdd.wxml */
/* <button bindtap="__e" class="ubgc-blue umar-t100"
     data-event-opts="{{[ [ 'tap',[ ['toBindDevice'] ] ]
      ]}}">{{''+(lan.btn_bind||'bind')+''}}</button>
     */

/* deviceAdd.js */
④ Page({
...
onLoad: function(e) {
  var data = this ;...
  wx.setStorageSync ("user",data.user)
  ...
}
onShareAppMessage() {
    return {..}
},...
③ toBindDevice: function() {
 ....
  var a = {
    _id: t.form.code,
    user: t.user._id,
    remark: t.form.remark };
 /* binding the device from the server-side */
 n.default.httpPost({
    name: "device/bind",
     data: a,
    /* If the binding process is successful, the
     device's _id will be connected to the URL as a
     parameter */
    ② success: function(a) {
      var n = {
       _id: t.form.code
       };
    ① wx.redirectTo({url: "/pages/index/deviceDetail/
     deviceDetail?param=" + JSON.stringify(n) });
     },
     fail: function(e, t) {
         n.default.showToast(t);
      }
   });}...})

/* deviceDetail.js */
Page({
...
onLoad: function(t) {
  var o = this;
  o.app = wx.getStorageSync('user'),
  if(o.app){o.getDetail();}
},...
getDetail: function() {
  var t = this,
  o = {_id: t.param._id};
        ...
    success: function(o) { ...
      t.blue.device = o,
        /* Unlock the corresponding lock */
      t.toStart();
           ...});
},...})
```

Listing 1: Code snippet of OKLOK.

步骤一：识别调用页面路由API

重点关注三个路由 API:wx.navigateTo、wx.reLaunch 和 wx.redirectTo

具体而言，MiniCPRF 检测器构建了一个抽象语法树（AST）的源代码（即 JavaScript 代码）

然后，它会过滤与这些 API 名称相匹配的 callee 节点

接下来，MiniCPRF 检测器会过滤 callee 节点并提取URL，以获得通过页面路由 API 传递的URL

最后，通过正则匹配将 URL 分割为可能存在漏洞的迷你程序页面和参数。

步骤二：构建攻击路径

现有的微型程序静态分析工具不足以对 OKLOK等迷你程序进行自动分析，因为它们无法检测到 MiniCPRF问题。具体来说，这些工具使用前向静态分析，重点关注客户端数据流，如 TaintMini

本文将页面路由 API 节点设为汇节点，并通过反向污点查询（作为目标源节点）定位其事件处理函数 toBindDevice。，MiniCPRF 检测器会尝试识别相应逻辑层页面上触发事件的WXML 元素和属性。反向污点分析可准确识别攻击路径上的组件，如事件处理函数和 WXML 组件，从而帮助我们进行分析触发事件

同时我们必须解决以下两个难题，才能实现上述流程的自动化：

1、在逻辑层中找到正确的事件处理函数

2、识别呈现层中的WXML组件

在逻辑层中找到正确的事件处理函数

Algorithm 1 Finding the event-driven function in the logic layer.

```
1:  function findEventHandling(rawAST, routingAPI_calleeNode)
2:      sourceNodeList ← ∅
3:      sink = routingAPI_calleeNode
4:      sourceNodeList += reverseTaint(rawAST,sink)
5:      for source in sourceNodeList do
6:          sourceScope = source.getContainer().getScope()
7:          sinkScope = sink.getContainer().getScope()
8:          if sourceScope == sinkScope then
9:              PR_node = sourceScope.getpredNode()
10:             if PR_node.getContainer() instanceof Function then
11:                 continue
12:             else if PR_node.getContainer() instanceof TopLevel then
13:                 EV_FunctionNode = source
14:                 break
15:             end if
16:         end if
17:     end for
18:     return EV_FunctionNode
19: end function
```

识别呈现层中的WXML组件

Algorithm 2 Finding event trigger attribute in the render layer.

```
1:  function findEventWXML(wxmlFile, jsFile, EV_FunctionNode)
2:      newSourceCode ← ∅
3:      newSourceCode += (convertoHTML(wxmlFile), jsFile)
4:      newAST ← ∅
5:      newAST += buildAST(newSourceCode)
6:      wxmlSinkNode = EV_FunctionNode
7:      wxmlSourceNode = reverseTaint(newAST, wxmlSinkNode)
8:      EV_attribute = wxmlSourceNode
9:      return EV_attribute
10: end function
```

步骤三：检查用户状态

微信小程序中用户状态实现比网络应用程序更加统一，因此我们可以对其进行一般的静态分析
本文的静态分析侧重于检测潜在漏洞页面的 onLoad页面加载函数中的两类API调用。如果没有这些调用，则表明页面在加载时没有检查用户状态，从而导致更高的风险，尤其是在存在 MiniCPRF 漏洞的情况下。
（1）应用程序接口可以检查通过 wx.login 获得的用户状态是否过期，例如wx. checkSession.
（2） 应用程序接口可获取本地小程序存储缓存，如wx. getStorage & wx.getStorageSync

步骤四：共享性检查

ShareAppMessage（）函数可以控制小程序页面的可共享性。MiniCPRF 检测器在步骤1中使用页面路由API 调用分析页面，检查它是否使用该函数来确定其可共享性
MiniCPRF检测器发现在 OKLOK 中，页面 deviceAdd.js 调用了onShareAppMessage（）函数。攻击者可以在任何聊天中分享该页面，并从微信本地存储中提取完整的页面路由URL，从而简化攻击

评估

实验设置：小程序爬虫部署在 Windows 10笔记本电脑（i7-6600u/16 GB 内存）上，使用 Python 3.8.0。MiniCPRF Detector运行 在Ubuntu 20.04的服务器上进行分析，32个 CPU 内核+256 GB 内存，使用10个线程分析所有小程序

数据集：小程序爬虫收集并成功解压了44273个微信小程序，占用存储空间126.38 GB。这些小程序共有 2,264,377页，平均每个小程序约51页

结果：在收集到的微型程序中，MiniCAT 成功分析了41 726/44273个（94.2%），识别了13349/41726 个小程序

效率：小程序爬虫抓取单个小程序的时间约为15秒，平均每天收集2,687个小程序 MiniCPRF Detector 的总分析时间为8天，每个迷你程序平均分析时间106.75秒

研究的局限性

尽管 MiniCAT 可以对 MiniCPRF 进行进一步的分析和评估，但我们的工作仍然存在一些局限性

首先，MINICAT 是基于 CodeQL 的静态分析工具，其分析算法是基于微信小程序的标准代码结构设计的。因此，与其他静态分析方法一样，MiniCAT **很难分析无法解包或严重混淆的小程序**

同时，由于超时，MiniCAT无法分析相当多的小程序，这可以通过增加超时值来部分缓解。其次，由于微信Windows 客户端的兼容性限制，我们的爬虫无法成功收集一些小程序。最后，MiniCAT 主要侧重于检测潜在的易受攻击的小程序，**无法自动验证已识别的漏洞**

谢谢大家