

警惕“NFC欺诈幽灵”：针对中国区用户的NFC中继攻击及恶意软件深度分析 | CTF导航

 ctfiot.com/292972.html



中际·2.3

安全便捷的支付体验



用户名



登录密码



记住密码



发送端



接收端

更多安全资讯和分析文章请关注启明星辰ADLab微信公众号及官方网站
(adlab.venustech.com.cn)

第一章

概述

2026年1月21日，启明星辰ADLab在威胁狩猎中捕获到一款针对中国用户定制的NFC中继攻击样本。通过溯源分析，我们发现该样本是一款基于臭名昭著的NFU Pay（恶意软件即服务MaaS）生态而定制的恶意代码，其攻击目标聚焦于中国境内用户群体。与传统银行木马依赖钓鱼页面窃取账户凭证或拦截短信验证码的攻击模式不同，NFC中继攻击利用了近场通信技术的物理特性，使攻击者能够即时获取被盗资金，有效绕过了银行转账延迟到账等传统风控措施。

该攻击样本通过“读取端”设备（受害者设备）近场窃取受害者银行卡APDU指令，经C2服务器实时路由至“接收端”设备；接收端利用HCE功能模拟银行卡与POS机交互，完成毫秒级双向中继。全程POS机无法察觉交易响应实际来自远程真实卡片，导致非接触式盗刷。为了便于后续威胁追踪和情报共享，我们将新发现的恶意软件命名为“NFC欺诈幽灵(NFC-Ghost)”。该恶意软件除了被黑客用于隐秘盗取受害者钱财外，还常被电信诈骗分子用于骗刷银行卡、信用卡等近场交易环节，并且还常常被用于国际间的洗钱活动。

“NFC欺诈幽灵”的开发目标具有明显针对性和目的性，比如其应用界面、提示信息、日志输出均采用简体中文；客户端模式标识从英文“POS_terminal/Card_reader”改为中文“接收端/发送端”；API路径使用品牌拼音缩写“/zj/”（中际）；样本品牌名称“中际”及版本标识“Card-2.0/Card-2.3”等都表明其开始了针对中国市场的侵入。

在深度溯源过程中，我们发现NFU Pay平台已衍生出多个定制品牌，除本次分析的“NFC欺诈幽灵”样本外，还包括PhantomCard、Lightning NFC等变种。当前NFC中继攻击领域已形成多个相互竞争的MaaS生态：SuperCard X是另一个由黑客运营的独立平台，其代码架构与NGate恶意软件存在相似性，两者均基于德国达姆施塔特工业大学发布的NFCGate开源项目演化而来。值得关注的是，SuperCard X同样被国内电信诈骗团伙用于实施盗刷、洗钱等非法活动。

“NFC欺诈幽灵”通过安卓系统的NFC HCE（Host Card Emulation，主机卡模拟）功能，实现银行卡APDU指令的实时中继转发。该恶意软件将发送端和接收端功能集成于同一应用中，攻击者在两台手机上安装同一应用后，通过登录界面分别选择“发送端”或“接收端”模式进行配合作业：发送端负责诱导受害者将银行卡贴近手机以读取NFC数据，接收端则在远程POS机或ATM机前模拟该银行卡完成支付或取现操作。整个攻击过程中，受害者的银行卡无需离开其视线范围，攻击行为具有极高的隐蔽性。

为深入了解该威胁的技术细节和攻击手法，启明星辰ADLab对捕获的样本进行了全面的逆向分析工作，涵盖追踪溯源和代码深度剖析等多个维度。本报告旨在为安全研究人员提供详尽的技术参考，同时为普通用户提升安全防范意识提供指导。后续章节将详细阐述我们的分析过程、技术发现以及相应的安全建议。

第二章

追踪溯源

在本次分析过程中，我们对捕获的恶意软件样本进行了全面的逆向分析和关联追踪。该恶意软件将发送端和接收端功能集成于同一应用中，用户通过登录界面选择运行模式，软件启动的主界面如图2-1所示。



中际·2.3

安全便捷的支付体验

 用户名

 登录密码

☐ 记住密码

☒ 发送端

☐ 接收端



图2-1 恶意软件登录界面及模式选择

在分析过程中，我们发现该恶意软件的后台管理系统仍处于活跃状态。后台登录界面采用中文设计，支持用户名密码、手机短信验证以及企业微信等多种登录方式，如图2-2所示。

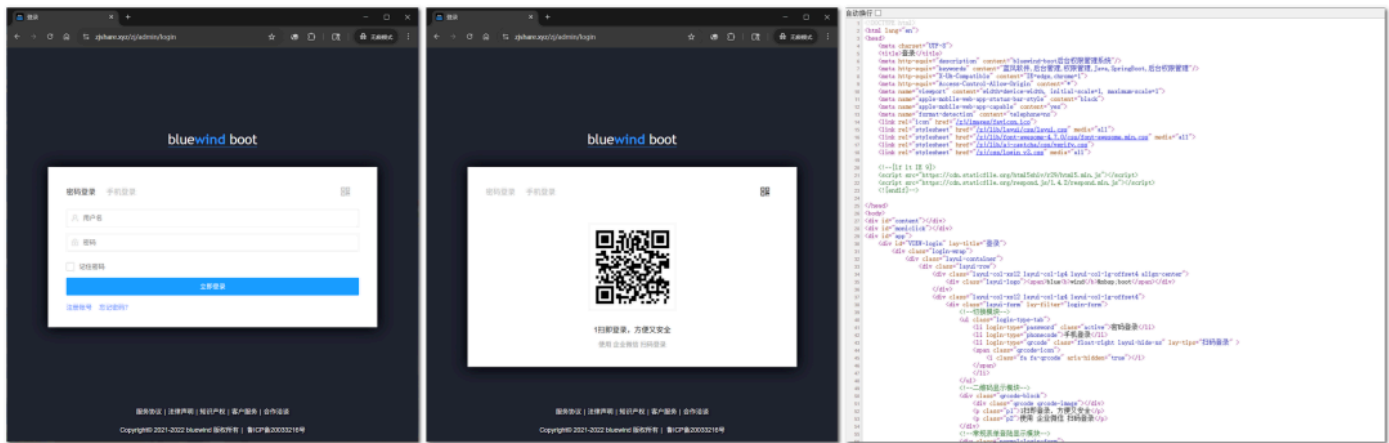


图2-2 后台登录界面

通过审查网站源码，我们注意到页面描述显示为“bluewind-boot后台权限管理系统”。经GitHub检索确认，攻击者直接复用了开源项目bluewind-boot (<https://github.com/lllillyxy/bluewind-boot>) 作为后端框架，因此后台界面本身未能提供有效的攻击者溯源线索。

为进一步扩展样本集，我们以主Activity特征字符串“nfc.share.nfcshare.”作为关联查询条件，在多个样本库中进行检索，最终收集到该恶意软件家族的76个样本。通过对这些样本的证书签名信息、包名特征、应用名称等多个维度进行系统性分析，我们得以勾勒出该恶意软件家族的技术演进脉络和攻击者的运营特征。

2.1 样本证书与签名分析

数字证书是Android应用的重要身份标识，通过对收集到的76个样本进行证书信息提取和统计分析，我们发现该恶意软件家族在证书使用上呈现出明显的规律性特征，这些特征为追踪攻击者活动轨迹和评估威胁规模提供了重要依据。

图2-3 签名证书时间线

证书签名时间能够反映样本的生成时间节点，是追踪恶意软件家族活动周期的重要指标。如图2-3所示，我们对76个样本的证书签名时间进行了统计分析，发现样本的时间分布呈现出明显的分层特征。

在早期样本方面，有8个样本使用了2008年签发的Android SDK默认调试证书，另有2个样本使用了2021年4月签发的证书。这些样本可能为早期开发测试版本，或是攻击者为追求快速分发而跳过正式签名流程的临时版本。

从正式运营时间来看，最早的正式样本签名时间可追溯至2024年11月，共有6个样本在该月被签名，这一时间节点与该家族在野外被首次发现的时间基本吻合。进入2025年后，样本签名活动呈现明显的增长态势：3月有5个样本、6月有3个样本、8月有11个样本。值得关注的是，2025年9月成为样本产出的绝对高峰期，单月签名样本数量达到22个，占总样本量的28.9%。随后的10月份仍保持较高活跃度，有12个样本被签名；11月和12月分别有5个和1个样本。进入2026年1月，我们又捕获到1个最新样本，表明该家族仍在持续活跃运营。

这一时间分布特征表明，该恶意软件家族背后的威胁行为者具备持续的开发和运营能力，且在2025年下半年明显加大了攻击投放力度——仅8月至10月三个月内就产出45个样本，占总量的59.2%。

图2-4 证书颁发者类型分布

除签名时间外，证书颁发者信息同样是追踪恶意软件家族的重要维度。如图2-4所示，我们对76个样本的证书颁发者信息进行了分类统计，识别出8种典型的证书使用模式。

“xinjiang”自签名证书是该家族使用最广泛的证书类型，共有37个样本采用，占总量的48.7%。这类证书的显著特征是在地理位置字段中统一使用“xinjiang”作为标识，证书主题中的国家（C）、省份（ST）、城市（L）字段均设置为“xinjiang”。更值得关注的是，其组织名称（O）和组织单位（OU）字段采用“两位随机字母+Unix时间戳”的命名格式，如“O=yf1755531900072”。通过对时间戳的解析，我们发现这些数值与样本实际签名时间高度吻合，强烈暗示攻击者使用了自动化工具批量生成签名证书。

简单数字证书是第二大类，共15个样本使用，占比19.7%。这类证书的特点是主题极为简化，通常仅包含一个简单数字作为CN字段值，如“CN=1”、“CN=8789789”等。这种配置通常出现在早期开发版本或需要快速分发的定制版本中，反映出攻击者在某些场景下优先考虑分发效率。

Android调试证书共8个样本使用，占比10.5%。这类样本使用Android SDK自带的默认调试证书，签名起始时间为2008年。在正常开发流程中，调试证书仅用于测试阶段，该家族中存在此类样本表明这些可能是开发测试版本，或攻击者为快速分发而跳过正式签名步骤。

其他自签名证书共5个样本，占比6.6%。这类证书格式各异，包括使用中文字符（如“CN=的撒娇好”）、地区标识（如“C=中国, ST=江苏”）或随机字符串等，反映了样本来源的分散性，可能涉及多个分发渠道或定制客户。

RC/852证书共4个样本，占比5.3%。这类证书在国家代码字段使用“852”（香港国际电话区号），可能是攻击者试图伪装成香港开发者。使用此类证书的样本主要是早期版本，签名时间集中在2025年3月。

“admin”自定义证书共4个样本，占比5.3%。这类证书所有字段均使用“admin”加随机字符串格式，与xinjiang证书类似显示出批量自动化生成特征，但采用不同命名模板。使用此类证书的样本签名时间集中在2025年11月至12月，可能代表攻击者后期采用的新签名策略。

“NFC/Dubai”证书共2个样本，占比2.6%。这类证书使用阿联酋（ARE）作为国家代码，迪拜作为城市标识，暗示可能是针对中东地区定制的版本。

伪造企业证书仅1个样本，占比1.3%。该样本使用“Innovation Hub”作为组织名称，混合使用新加坡和巴西的地理标识，意图通过正规企业名称提升可信度，规避安全检测。

综合来看，证书颁发者的多样性反映了该恶意软件家族的MaaS运营特点：核心开发团队使用自动化工具批量生成xinjiang系列证书用于主要分发，同时为不同地区的定制客户提供差异化的证书配置方案。

2.2 家族归属确认

通过对样本证书的统计分析，我们已初步勾勒出该恶意软件家族的运营规模和活动周期。然而，证书信息仅能反映样本的签名特征，要准确确认家族归属，还需要从代码层面寻找直接证据。为此，我们采用代码比对分析方法，将“NFC欺诈幽灵”与已确认归属于NFU Pay平台的样本进行对比，从代码结构、核心类实现、通信协议等多个维度验证其同源性。

2.2.1 参照样本选取

在收集到的76个样本中，我们注意到一个包名为“nfc.share.nfcshare”的样本，其包名与我们用于关联检索的主Activity特征字符串完全一致。该样本的应用名称为“NFU”，应用图标中同样包含NFU字样。通过对该样本的逆向分析，我们在其登录失败处理逻辑的Toast提示信息中发现了明确的归属证据——“遇到问题联系：@nfupay666”，如图2-5所示。

图2-5 NFU Pay的联系方式

@nfupay666是NFU Pay在Telegram平台的官方客服账号，如图2-6所示，这一发现直接证实该样本为NFU Pay平台官方分发的版本。

图2-6 NFU Pay的联系方式

因此我们选取该样本（MD5: 07a8dcccfc3c5496423923a0033dbe11）作为参照样本，与“NFC欺诈幽灵”样本进行代码比对。

2.2.2 代码对比分析

通过GDA反编译工具对两个样本进行代码目录结构对比分析，我们发现两者的核心代码均位于“nfc.share.nfcshare”包路径下，具有高度一致的模块化组织结构，如图2-7所示。

两个样本均包含相同的service和model子包结构。在service子包中，核心服务类EmulationService.java（NFC HCE卡模拟服务）和MqttService.java（WebSocket通信服务）的类名和包路径完全一致。在model子包中，数据模型类MqttChannel.java（通道类型枚举）、NfcInfo.java（NFC数据封装）、WSMessage.java（WebSocket消息格式）、CardInfo.java（卡片信息）同样保持一致。

两者的主要差异在于混淆程度不同。“NFC欺诈幽灵”经过了更强的代码混淆处理，部分辅助类被重命名为单字母类名（如a.java、b.java、c.java等），而参照样本（“NFU”）保留了原始的类名（如ApiService.java、NetworkUtils.java等）。但值得注意的是，两个样本的核心功能类命名均保持不变，这表明它们共享相同的代码基础框架。

图2-7 样本代码目录结构

在代码结构一致的基础上，我们进一步对比了两个样本的核心功能实现代码。

EmulationService是NFC中继攻击的核心服务类，负责接收POS机的APDU指令并转发至C2服务器。如图2-8所示，两个样本的EmulationService实现逻辑完全一致：均继承自Android系统的HostApuService类。onDeactivated函数内的打印日志的TAG都相同，唯一的差异是分析样本对日志字符串进行了NPStringFog加密处理，而参照样本保留了明文日志。

图2-8 EmulationService核心代码对比

MqttChannel枚举类定义了WebSocket通信的通道类型。如图2-9所示，两个样本的MqttChannel定义完全相同，均包含FETCH_CHANNEL、SEND_CHANNEL、LOG_CHANNEL、CARD_INFO_CHANNEL、CARD_REMOVED、NOTIFICATION_CHANNEL、ANSWER_CHANNEL、OFFLINE_CHANNEL共8种通道类型，枚举值的名称和顺序完全一致。

图2-9 MqttChannel枚举定义对比

综合以上代码层面的直接证据，我们可以确认本次分析的“NFC欺诈幽灵”属于NFU Pay MaaS生态系统的定制产品。这种为不同客户定制独立品牌的做法符合MaaS平台的典型商业运营模式——核心开发团队提供基础恶意软件框架，下游客户可根据需求定制品牌名称、界面风格和目标地区。在我们收集的样本中，除“中际”外还发现了“NFU”、“T4”、“鲲鹏支付”、“云联”、“EQUIPE GHOST”等多个品牌名称，这些均为NFU Pay平台为不同客户定制的产品变种。

2.2.3 基础设施关联分析

除代码层面的同源性证据外，我们还通过C2基础设施的关联分析进一步验证了家族归属关系。

在“NFC欺诈幽灵”样本的逆向分析中，我们提取到其C2服务器域名www.zjshare.xyz。通过查询该域名的WHOIS注册信息，如图2-10所示，该域名注册时间为2025年11月27日。

图2-10 域名www.zjshare.xyz的WHOIS信息

对该域名进行DNS解析，得到其指向的服务器IP地址为185.106.176.32。值得注意的是，该IP地址在另一个“中际”样本（MD5: 45902fa3f8879a18c97b12fbb186e196）中以硬编码形式直接出现，该样本的证书签名时间为2025年12月1日，仅比域名注册时间晚4天。这一时间关联印证了前文证书签名时间溯源的准确性，同时表明攻击者在2025年11月底至12月初期间完成了C2基础设施的部署和样本的签名分发。

通过威胁情报平台对该IP地址进行关联查询，我们发现它还被其他NFU Pay家族变种所使用，其中包括品牌名为“鲲鹏NFC”的样本，如图2-11所示。

图2-11 通过IP关联发现的“鲲鹏NFC”样本

这一发现表明，NFU Pay平台为旗下多个品牌变种提供统一的C2基础设施服务，“中际”、“鲲鹏NFC”等品牌共享相同的后端服务器。这种架构设计进一步印证了该恶意软件家族的MaaS运营模式。

2.3 攻击者画像

基于上述分析，我们可以勾勒出该恶意软件背后威胁行为者的基本画像。

（1）从语言特征来看，样本的用户界面文本、日志输出信息、代码注释均使用简体中文，表明开发者为中文母语使用者。例如，在LoginActivity.java中可以看到“登录请求失败”、“可能网络不稳定，请重试”等中文提示信息。

（2）从运营模式来看，该恶意软件采用MaaS（恶意软件即服务）商业模式，通过Telegram渠道（@nfupay666）进行销售和客户支持。这种模式允许攻击者将恶意软件作为服务出租给下游犯罪分子，降低了实施NFC中继攻击的技术门槛。

（3）从技术能力来看，威胁行为者具备Android逆向开发能力，熟悉NFC/EMV支付协议，并持续迭代混淆对抗技术以规避安全检测。从版本的技术演进可以看出，攻击者在字符串加密、包名随机化、应用名称混淆等方面进行了显著增强。

第三章

技术分析

本章节将从攻击原理、核心功能模块、代码混淆技术三个维度对“NFC欺诈幽灵”恶意软件进行深入剖析，并结合实际攻击场景还原完整的攻击链路。

3.1 攻击原理与架构设计

3.1.1 NFC中继攻击原理

“NFC欺诈幽灵”恶意软件的核心攻击原理是利用Android系统的NFC HCE（Host Card Emulation，主机卡模拟）功能实现银行卡数据的实时中继。HCE是Android 4.4引入的技术特性，允许应用程序在无需安全元件（SE）的情况下模拟NFC卡片，原本用于移动支付等合法场景。然而，攻击者滥用这一技术，将其改造为银行卡数据的远程中继工具。

如图3-1所示，整个攻击过程涉及五个关键角色：受害者银行卡、发送端设备（Reader）、C2服务器、接收端设备（Tapper）、POS机/ATM机。攻击流程如下：发送端设备通过NFC读取受害者银行卡的APDU数据，并通过WebSocket实时转发至C2服务器；C2服务器将数据路由至接收端设备；接收端设备利用HCE功能模拟该银行卡，响应POS机的APDU请求；POS机的请求同样沿原路返回至真实银行卡获取响应。整个中继过程在毫秒级时间内完成，POS机无法区分响应来自本地卡片还是远程中继。

图3-1 NFC中继攻击原理示意图

3.1.2 双端架构设计

“NFC欺诈幽灵”恶意软件采用双端架构设计，将发送端（Reader）和接收端（Tapper）功能集成于同一应用中，用户通过登录界面的模式选择控件切换运行模式，如图3-1所示。这种设计简化了攻击者的部署流程，只需在两台设备上安装同一应用并选择不同模式即可完成配对。

通过分析LoginActivity的源码，我们发现模式切换通过RadioGroup控件实现。当用户选择接收端模式时，应用将clientId设置为接收端，并将isPosMode标志保存为true；当用户选择“发送端”模式时，clientId设置为“发送端”，isPosMode标志保存为false。模式状态通过SharedPreferences持久化存储，确保应用重启后能够恢复之前选择的运行模式，模式选择的监听代码如图。值得注意的是，早期NFU版本使用英文标识“POS_terminal”和“Card_reader”作为clientId，而“NFC欺诈幽灵”版本改用中文标识，这一变化可能是为了适应国内用户群体。

图3-2 监听Radio的选择，设置clientId

在发送端模式下，恶意软件的主要功能是读取受害者银行卡的NFC数据。当受害者在攻击者的诱导下将银行卡贴近手机NFC感应区时，手机作为NFC读卡器与银行卡建立ISO 14443通信，恶意软件捕获所有APDU交互数据并通过WebSocket实时转发至C2服务器。

在接收端模式下，恶意软件的主要功能是模拟银行卡。EmulationService作为HCE服务运行，当攻击者将手机贴近POS机时，恶意软件接收POS机发送的APDU指令，通过WebSocket转发至C2服务器获取真实银行卡的响应，再将响应返回给POS机完成交易，接收端工作界面如图3-3所示。

图3-3 接收端界面

3.2 核心功能模块分析

本节将深入分析“NFC欺诈幽灵”恶意软件的核心功能模块实现，包括NFC卡模拟服务、实时通信模块、C2通信基础设施等关键组件。

3.2.1 NFC HCE卡模拟服务

在AndroidManifest.xml中，如图3-4所示，EmulationService的服务声明体现了HCE服务的标准配置模式。服务被注册为android.nfc.cardemulation.category.PAYMENT类别，使其能够响应支付类NFC请求。permission属性设置为android.permission.BIND_NFC_SERVICE，确保只有系统进程能够绑定该服务，防止第三方应用恶意调用。foregroundServiceType属性值为0x10（对应connectedDevice类型），这是Android 10引入的前台服务类型声明，确保服务在后台运行时具有较高的进程优先级，不会被系统内存管理机制轻易终止。meta-data标签引用的aid_list.xml文件定义了服务响应的AID（Application Identifier）列表，决定了哪些类型的NFC卡片请求会被路由至该服务。

图3-4 EmulationService核心代码实现

EmulationService是实现NFC中继攻击的核心服务类，继承自Android系统的HostApduService基类。HostApduService是Android 4.4（API Level 19）引入的系统服务，专门用于实现基于主机的NFC卡模拟功能。当POS机或ATM机向手机发送APDU（Application Protocol Data Unit）指令时，Android系统会自动将指令路由至已注册的HCE服务，并回调processCommandApdu方法进行处理。

该服务的核心实现如图3-5所示。processCommandApdu方法是APDU指令处理的入口点，其实现逻辑简洁而关键：首先调用Utils.encodeHexStr方法将接收到的原始字节数组转换为十六进制字符串格式，然后通过Utils.mqttService.pushMessageToMqtt方法将数据发送至FETCH_CHANNEL通道，最终经由WebSocket转发至C2服务器。该方法返回而非实际响应数据，表明恶意软件采用异步响应机制——服务端收到APDU指令后，将其转发至发送端获取真实银行卡的响应，再通过SEND_CHANNEL通道回传，最后由EmulationService调用sendResponseApdu方法将响应返回给POS机。当处理过程中发生异常时，方法返回“6F00”状态字，这是ISO 7816-4标准定义的通用错误状态码，表示“无精确诊断信息”。

图3-5 EmulationService核心代码实现

在服务生命周期管理方面，onCreate方法将服务实例保存至Utils.emulationService静态变量以便全局访问，同时更新界面状态指示器并向NOTIFICATION_CHANNEL通道发送“HCE已启动”通知。onDeactivated方法在NFC链路断开时被回调，根据断开原因（链路丢失或被其他应用抢占）记录相应日志。onDestroy方法负责清理资源，包括置空静态引用、发送服务停止通知、通过Handler在主线程更新界面状态。

3.2.2 WebSocket实时通信模块

MqttService负责建立与C2服务器的WebSocket长连接，实现APDU数据的双向实时传输。值得注意的是，尽管类名包含“Mqtt”字样，但通过代码分析确认该模块实际使用WebSocket协议而非MQTT协议进行通信，这种命名方式可能是开发者为混淆安全分析而故意设置的误导性命名，MqttService内部的成员函数如图3-6所示。

图3-6 伪装成Mqtt模块的WebSocket代码

MqttService内部维护一个WebSocket客户端管理器实例wsClientManager，负责管理与C2服务器的连接生命周期。connect方法接收服务器URL作为参数，在建立连接前会检查当前连接状态以避免重复连接。disconnect方法通过发送关闭帧（close frame，状态码1000表示正常关闭）断开连接并清理相关状态。pushMessageToMqtt方法是消息发送的核心接口，该方法在新线程中执行实际的发送逻辑，避免阻塞主线程影响用户界面响应。

图3-7 MqttService的connect和disconnect函数

通信协议采用JSON格式封装消息。WSMessage对象作为消息载体，其字段定义如表3-1所示。

表3-1 WSMessage消息结构

NfcInfo对象封装APDU数据的具体内容，其字段定义如表3-2所示。

表3-2 NfcInfo数据结构

MqttChannel枚举类定义了8种通道类型，实现了在单一WebSocket连接上的多路复用通信，各通道功能如表3-3所示。

表3-3 MqttChannel通道类型定义

这种多通道设计使得不同类型的数据能够在同一连接上并行传输，同时保持逻辑上的隔离。此外，恶意软件还实现了交易金额解析功能，能够从APDU指令流中提取交易金额和货币代码（遵循ISO 4217标准），使攻击者可以实时监控每笔盗刷交易的金额。

3.2.3 C2通信基础设施

通过对样本的逆向分析，我们提取到该恶意软件的C2通信架构。恶意软件启动后需要先完成服务器登录认证，登录流程如图3-8所示。

图3-8 用户登录函数

登录请求采用JSON格式封装，包含以下参数：account（账号）、password（密码）、rcid（设备Android ID，用于唯一标识设备）、mode（运行模式标识）。服务器验证通过后返回WebSocket连接地址和端口信息，客户端据此建立实时通信通道用于APDU数据中继。

API服务器地址在代码中经过NPStringFog加密存储，解密后得到基础地址为www.zjshare.xyz，采用HTTPS协议进行加密通信。值得注意的是，API请求路径统一使用“/zj/api/”前缀，其中“zj”为该恶意软件品牌名称“中际”的拼音首字母缩写，这一命名特征与样本的品牌标识保持一致。主要API端点如表3-4所示。

表3-4 C2服务器API端点

3.3 代码保护与对抗技术

通过对不同时期样本的对比分析，我们观察到该恶意软件家族在代码保护技术上呈现明显的迭代升级趋势，反映出攻击者持续提升对抗安全检测能力的意图。本节将从字符串加密、标识符混淆、加壳策略、权限配置四个维度，详细分析“NFC欺诈幽灵”样本所采用的代码保护技术及其演进特征。

3.3.1 字符串加密技术

早期NFU版本的敏感字符串以明文形式存储在代码中，包括C2服务器地址、API端点、日志标签、提示信息等，这为安全研究人员的静态分析提供了便利。我们本次分析的版本则引入了NPStringFog自定义字符串加密方案，有效阻止了基于字符串特征的静态检测。通过对NPStringFog.java源码的逆向分析，我们完整还原了其加解密算法，如图3-9所示。该方案采用HEX编码结合XOR异或运算的两阶段处理流程：加密时首先将原始字符串转换为HEX编码的字节序列，然后使用硬编码密钥“npmanager”对字节数组进行逐字节循环XOR运算。解密时执行逆向操作，decode方法首先将HEX字符串还原为字节数组，使用相同密钥进行XOR运算恢复原始数据，最后将特殊标记“\$-sxg-\$”替换为反斜杠字符。这种加密方案虽然算法简单、密钥固定，但足以将敏感字符串从静态分析视野中隐藏，增加了自动化检测的难度。

图3-9 NPStringFog字符串加解密算法

3.3.2 包名与应用名称混淆

在包名策略方面，早期NFU版本使用相对固定的包名格式，如nfc.share.nfcshare、com.nfupay666.rc等，这些命名虽然不符合常见合法应用的包名规范，但仍具有一定规律性，容易被安全厂商加入检测黑名单。我们本次分析的版本采用完全随机化的包名策略，如sckfkndjdrzays.cezqfefjwxkqrgd.sioivedygyuhq，由三段随机字母组成，每段长度在10-15个字符之间，有效规避了基于包名特征的检测规则。

在应用名称方面，早期版本使用简单的中文名称如“NFU”、“中际”，未采用任何混淆技术。本次分析的版本则在应用名称中嵌入了大量Unicode零宽字符，如图3-10所示。通过分析AndroidManifest.xml中的android:label属性，我们发现显示为“中际·2·3”的名称实际包含大量不可见字符，包括零宽连接符（U+200D，ZWJ）、零宽非连接符（U+200C，ZWNJ）、零宽空格（U+200B，ZWSP）、词连接符（U+2060，WJ）等。这些字符在视觉上完全不可见，但会显著改变字符串的二进制表示和哈希值，从而干扰基于应用名称的人工识别和自动化检测。

图3-10 超长的应用名称造成自动化分析系统异常

3.3.3 加壳策略演进

该恶意软件家族的代码保护策略呈现出明显的演进特征。早期NFU版本未采用任何代码保护措施，敏感字符串以明文形式存储。随着安全厂商检测能力的提升，攻击者开始引入代码保护机制。以MD5为8abdc38030d6686588f9a491e2a93957的样本为例，该样本采用了自定义加固方案，通过混淆处理避免核心代码直接暴露，加固后的代码目录结构如图3-11所示。

图3-11 加固后的代码目录结构

然而，“NFC欺诈幽灵”并未采用加壳保护，仅使用NPStringFog字符串加密方案。这一策略调整可能出于以下考量：加壳工具本身会引入可被识别的特征码，安全厂商可通过检测壳特征标记可疑样本；同时加壳会增加应用启动时间和运行时内存开销。采用轻量级的字符串加密方案，在保持一定保护效果的同时降低了静态特征暴露风险，体现了攻击者在隐蔽性与防护强度之间的权衡。

3.3.4 权限最小化策略

通过分析AndroidManifest.xml的权限声明，我们发现该恶意软件采用了“最小权限”策略，仅请求实现核心功能所必需的5项权限，如表格3-5所示。

表3-5 申请权限名称和功能

同时通过uses-feature标签声明android.hardware.nfc和android.hardware.nfc.hce两项硬件特性要求，并将required属性设置为true，这意味着该应用只能安装在支持NFC HCE功能的设备上。

与传统银行木马相比，该恶意软件的权限请求极为精简。传统银行木马通常会请求数十项敏感权限，包括READ_SMS和RECEIVE_SMS用于拦截短信验证码、READ_CONTACTS用于窃取通讯录、READ_EXTERNAL_STORAGE和WRITE_EXTERNAL_STORAGE用于访问存储空间、BIND_ACCESSIBILITY_SERVICE用于获取无障碍服务权限以实现界面劫持等。这些权限请求往往会引起用户警觉，也容易被安全软件标记为可疑行为。

该恶意软件采用最小权限策略的优势在于：首先，精简的权限列表降低了用户在安装时的警觉性，用户更容易接受一个只请求NFC和网络权限的应用；其次，较少的敏感权限请求使得样本更难被基于权限特征的安全检测规则识别；最后，这种策略也体现了攻击者对Android安全机制的深入理解和成熟的规避检测经验。

第四章

总结及建议

经过全面深入的技术分析，我们对“NFC欺诈幽灵”恶意软件及其归属的NFU Pay家族形成了系统性认识。该恶意软件瞄准中国用户，其利用Android系统的NFC HCE功能实现银行卡数据的实时中继，使攻击者能够在受害者毫不知情的情况下完成盗刷或取现操作。由于攻击过程中受害者的银行卡始终在其视线范围内，加之MaaS商业模式降低了攻击实施门槛，我们将该恶意软件家族的威胁

等级评估为高危。从样本签名时间分布来看，该家族在2025年下半年明显加大了攻击投放力度，仅8月至10月就产出45个样本，占总量的59.2%，显示出持续活跃的运营态势。据公开报道，我国境内已有多名受害者因感染该家族样本遭受经济损失，公安机关也已破获多起利用此类技术实施的电信网络诈骗案件。

对于普通用户而言，防范NFC中继攻击的关键在于提高安全意识。用户应仅从官方应用商店下载应用程序，对通过短信、社交媒体或陌生网站提供的下载链接保持高度警惕。需要特别注意的是，任何要求将银行卡贴近手机进行“验证”或“安全检测”的操作都应果断拒绝，正规金融机构不会采用此类方式进行身份核验。建议用户开启银行交易短信通知功能，一旦发现异常交易应立即联系银行冻结账户并向公安机关报案。

对于银行和支付机构而言，建议加强NFC支付交易的异常检测能力，重点关注交易地点与持卡人常用地地点不符、短时间内在不同地点发生多笔交易等异常特征。可考虑引入设备指纹技术识别交易发起设备，并对高风险交易增加短信确认或语音电话确认等二次验证机制。此外，金融机构应通过官方渠道向用户普及NFC中继攻击的风险特征，帮助用户识别和防范社会工程诱导。

IOC

注：以下为我们研究期间收集到的样本的IOC信息

[1]ws://185.106[.]176.32:8091/

[2]http://185.106[.]176.32:8080/zj/api/user_logout

[3]http://185.106[.]176.32:8080/zj/api/user_config

[4]http://185.106[.]176.32:8080/zj/api/user_login

[5]https://www.zjshare[.]xyz/zj/api/

[6]45902fa3f8879a18c97b12fbb186e196

[7]49961202edb37c093201b71907f742d4

[8]8017741d7840cb9d6a322de44771a1d3

[9]8abdc38030d6686588f9a491e2a93957

[10]ff54db962a351d853c551b258dbcc30e

启明星辰积极防御实验室（ADLab）

ADLab成立于1999年，是中国安全行业最早成立的攻防技术研究实验室之一，微软MAPP计划核心成员，“黑雀攻击”概念首推者。截至目前，ADLab已通过 CNVD/CNNVD/NVDB/CVE累计发布安全漏洞7000余个，持续保持国际网络安全领域一流水准。实验室研究方向涵盖基础安全研究、运营商基础网络设施安全研究、移动终端安全研究、云安全研究、信创安全研究、物联网安全研究、车联网安全研究、工控安全研究、无线安全研究、数据安全研究、AI安全研究、低空安全研究、高级威胁研究、攻防体系建设。研究成果应用于产品核心技术研究、国家重点科技项目攻关、专业安全服务等。



| 本篇文章来源于微信公众号: ADLab