

We submitted this paper earlier to IEEE S&P 2020. We attached the unaltered IEEE S&P 2020 reviews as follows.

Review #692A

Overall merit

2. Reject - This paper is well below the bar for Oakland. Will argue to reject it.

Brief paper summary (2-3 sentences)

This paper follows prior work that identified that if stored authentication credentials of Android apps are cloned into a different system, auto-login functionality might still work. Prior work assumed root-access to the victims device and the current paper shifts the threat model into something less realistic. A modification to the Cell Android container ecosystem shows that attacks against applications and mobile vendors with low security aptitude can work.

Strengths

- + Seemingly significant investment into updating Cell (will you open source your modifications?)
- + Another study demonstrating that application developers should use appropriate APIs (e.g., AccountManager API) instead of storing access credentials in local files

Weaknesses

- No novelty over existing work -- login credential cloning has been demonstrated by [18]-[20] and the MitM attack on an unprotected public WiFi has been known and used for decades
- Modifications to the threat model make it less likely for the adversary to succeed (i.e., temporal dependence on phone update and physical proximity for WiFi MitM attack)

Detailed comments for the author(s)

This paper is a great reminder for Android application developers that storing/caching access credentials in plain-old file-system objects is a terrible design decision. It also serves as a reminder that sending data over unencrypted connections can be a dangerous enterprise, as confirmed by the vulnerability acknowledgement of XioMi. While these are valid insights, they are all well known and nothing new.

The paper tries to distinguish itself from prior work by shifting the threat model. Unfortunately, the paper is dishonest in presenting the effects of assuming no-root but interception of device cloning traffic. Two fundamental constraints that the chosen threat model implies are (1) temporal, and (2) location. In earlier attacks, root access (even if remote) was sufficient. In the current paper, the attacker must be within WiFi proximity (paper states 8m). So remote attacks are fundamentally no longer possible. More importantly, the chosen threat model also bounds the temporal dimension of feasible attacks, and the paper completely ignores this very important aspect. That is, the attack can only work if the attacker is present when the user performs a device clone operation. Given that phone upgrade cycles are in the order of years, this means that exposure to the presented attack is miniscule at best. There is no doubt that the security ignorance of OEMs opened a possible

vulnerability, however, these additional restrictions over prior work, makes this attack strictly less powerful than prior work.

Furthermore, although the engineering effort in updating Cell is appreciated, this aspect does not contribute to the research aspects of the manuscript. Moreover, the description of the Cell updates in the paper is insufficient to convey to the reader what non-engineering challenges had to be addressed in that area. The VPDroid aspect only exists to enable 40 more apps to run after the authentication data has been cloned. Given the attack nature of this work, this addition to the overall apps that are vulnerable to the problem at hand does not warrant the substantial paper real estate devoted to VPDroid.

In summary, this paper is severely short on novelty over prior work. The text tries to create novelty by shifting the threat model. However, this shift has the effect that the attack becomes severely limited in practice and is less of a threat than existing attacks against the exact same vulnerabilities. As such, this paper should not be accepted for publication at the Symposium.

Review #692B

Overall merit

3. Weak reject - The paper has flaws, but I will not argue against it.

Brief paper summary (2-3 sentences)

The authors propose virtualization-assisted data-clone attacks that abuse the auto-login functionality used by many mobile apps to automatically log the attacker on the victim's account. The attack leverages vulnerabilities (e.g., weak default passwords) in phone clone apps by device vendors to copy the authentication data from the victim phone to the attacker's phone. To defeat device fingerprinting checks done by the apps, the authors present VPDroid, a virtualization framework build on top of the previously proposed Cells (SOSP 2011), which enables faking the environment of the target app to make it look as it is still running on the victim's device (instead of the attacker's device). The authors evaluate their framework on 236 popular and manage to attack 216 using VPDroid compared to 196 using the Xposed hooking framework.

Strengths

- + Attack does not require physical access, malware, or rooting of the victim's device
- + The authors enable Cells support on recent Android versions
- + Evaluation shows a large number of popular apps affected

Weaknesses

- Low novelty, attack is known and virtualization is built on top of existing framework
- Attacker needs to be nearby when user is cloning its phone. How often does cloning happen in public spaces?
- Attacker needs to know victim's phone parameters. The authors do not discuss how easy/difficult that is.

Detailed comments for the author(s)

I feel that the novelty is low since the credential cloning attack is known and the virtualization framework is built on top of an existing one. Also, it is not clear what is learned from this attack paper in terms of new techniques or defenses. The authors highlight that prior attacks did not work due to two limitations. First, that not only the credential needs to be copied, but also other files under `/data/data/app` directory. But, here it is not clear why that is a challenge. If the attacker has root access to the device then copying more files is no big a deal. Also, wouldn't the credentials be stored under `/data/data/app`? In that case, if the attacker has access to the locally stored credentials, then it also has access to the rest of files. The second challenge is bypassing device fingerprinting. That part is stronger, but the authors do not make a good case of highlighting their contribution over Cell and do not provide a security analysis on the difficulty of detecting VPDroid. You seem to claim that it is not possible to detect VPDroid for an app, but I have seen such transparency claims in the past being violated.

Your attack decreases some barriers for the attacker such as no need to have root access (e.g., malware, physical access) to the victim's device. However, your threat model requires the attacker to be nearby to the user when the user clones their phone. It seems unlikely that the victim clones its device in a public space as you suggest. I would expect most users to clone their phone at home or at the office instead. And the attack window is short. Your attack also assumes that the attacker knows the target device's attributes, so its fingerprint can be faked. While some attributes like brand and model may be guessable, low level attributes may not be. You really need to justify why you think that assumption is realistic. Overall, it does not feel like the attack is much easier in your scenario than in prior works.

Your attack leverages vulnerabilities in vendor phone clone apps, but these are not detailed. You are only describing one vulnerability due to weak default credentials. Instead, you should list exactly which phone clone apps you have found vulnerabilities in, and describe them. Are all the vulnerabilities due to weak credentials? Have they all been reported and fixed? What was the approach used to find them? Did you try to vulnerability discovery for different apps?

The main contribution is VPDroid that updates Cells with support for newer Android and extends it with some new virtualized functionality (GPS, Bluetooth, ADB). That is good, but feels a bit incremental. Specially so since you are only supporting one VP on top of the host, which is a significant limitation for a virtualization framework (although enough for your attack purposes). One important question here is whether there is additional functionality that would need to be virtualized. You added support for GPS, Bluetooth, ADB presumably because you observe those being used by the apps that do device fingerprinting. But, one wonders if other apps may look at other functionality whose virtualization is not currently supported. In general you need to better argue why VPDroid is a significant contribution over Cell.

There are already 20/236 (8.5%) apps where the proposed attack does not work because they either keep the stored credentials in memory use the Android Account Manager. Is this the way to defend against these attacks? Can the Account Manager cache be attacked in a similar way? Should apps with subscriber limit simply check the limit before performing an action (e.g., start streaming a show) instead of on initialization? Attack papers benefit from a discussion on how the issue should be resolved.

Smaller comments:

Did you observe any apps deleting the stored authorization token on phone reboot?

Are you planning to contribute your improvement to Cell? That would be a useful contribution.

You mention that large families of potentially harmful apps use privilege escalation exploits to root devices. Any app that uses an exploit can safely be considered malware, rather than potentially harmful.

The paper reads good, but feels quite repetitive. Also, it is weird to have a paper without related work section. In addition to the previous attacks there seems to be related works on Android authentication, hardcoded secrets in mobile apps, ...

Typos: ", where have"

Review #692C

Overall merit

4. Weak accept - While flawed, the paper has merit and we should consider accepting it.

Brief paper summary (2-3 sentences)

The paper proposed a practical approach to stealing the Android app user's login data and reuse this data to access the user's account in other mobile phones. The authors successfully mount their attacks on 216 apps and reported 14 zero-day vulnerabilities.

Strengths

- + A practical approach to mounting the data-clone attack
- + A lightweight virtualization platform to efficiently provide a virtualized environment for Android.
- + 14 zero-day vulnerabilities reported to and confirmed by app vendors.

Weaknesses

- The writing on VPDroid details are quite dense and hard to follow
- No performance evaluation

Detailed comments for the author(s)

This paper introduces a new attack: an attacker steals the locally-stored auto-login data of an Android user and uses this data to access the user's account. To steal the auto-login data, the authors propose to use the OEM-made phone clone apps which contain zero-day vulnerabilities. In the login phase, the authors also design a lightweight Android-based virtualization system called VPDroid to simulate the victim's phone environment without being noticed by the applications.

The proposed attack is not completely novel. There are some existing researches [18-20] that also use the auto-login data to access user accounts. However, the authors introduce a more practical attack solution that can deploy the data-clone attack on a non-rooted smartphone.

The following are some questions and concerns about this paper.

Writing: I found this paper quite well-written before Section IV (VPDroid System Design). However, Section IV is quite dense and hard for me to understand. This section describes how the authors modify each part of Cells and adds some new functions to Cells. But it assumes that the readers are familiar with Cells and thus does not explain the design of Cells well. Furthermore, this section mixes the mechanism of Cells and the updates made by VPDroid. Sometimes, it is not easy to differentiate these two parts.

- After reading the Filesystem paragraph, I cannot understand why "Cells's SD card partition virtualization does not work now".

- Some design choices are not well explained. For example, why do you modify the binder-driver data structure? Can't Cells's binder support multiple VPs?

2. I am curious about possible defense solutions. First, it seems that it is easy to detect how many devices are concurrently connecting to the server from the server side and thus it can place a limit on the number of concurrent connections. Second, why does the Android AccountManager APIs can defeat the data-clone attack? Where is the AccountManager cache? Third, VPDroid needs more Binder IPCs to virtualize a VP. Therefore, the application in VP may deduce that it is in a virtualized environment if the measured API latency is larger than in a native environment.

3. The VPDroid adds many new functionalities (11676 LOCs) to Cells. However, there are no performance experiments and data to show the performance overheads of VPDroid in the evaluation section. It is better to use evaluation data to make the readers understand VPDroid. For example, the binder service sharing technique makes it simpler to virtualize wireless and GPU in VPDroid. How does it improve the performance compared with Cells?

4. All the experiments in this paper are based on old Android versions (Android 6 and 8.1) while the newest version is 10.0. More experiments on the newer Android versions would make this paper stronger.

Minor issues:

1. Figures are not readable when printed in grayscale.

2. Celll->Cells, Cell->Cells

3. Kernal->Kernel (Figure 10)

=====

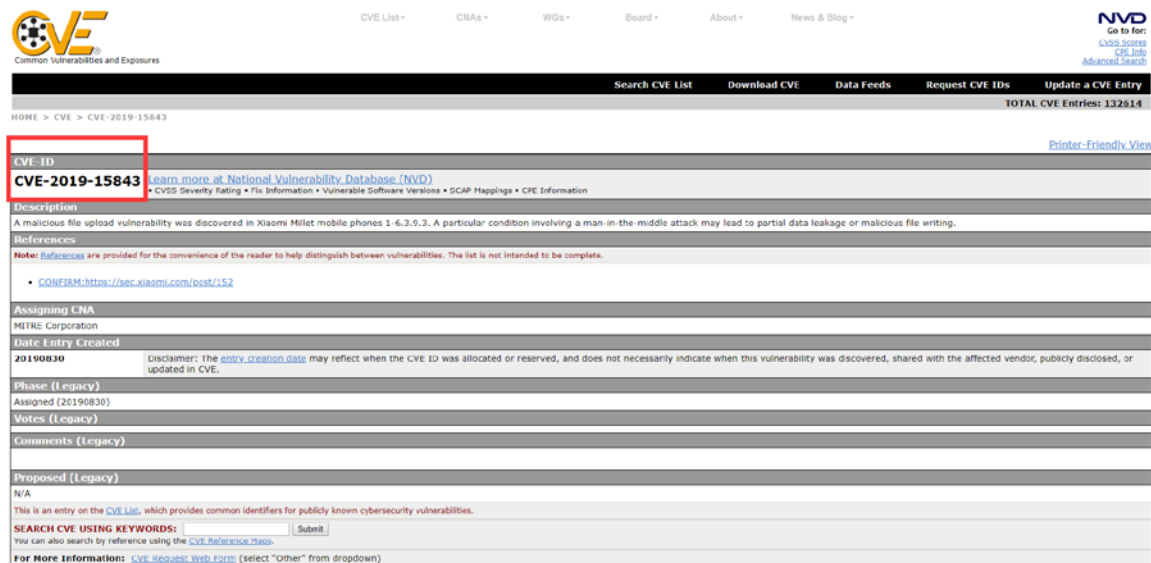
Authors' Response to Previous Reviews

1. **We highlight our intellectual contributions on Android OS virtualization: new user-level device virtualization & virtual phone's attribute customization.** Among our updates and new additions to Cells's device virtualization solutions, we improve Cells in two significant ways. **First**, we design a new user-level device virtualization solution, which has better portability and transparency than Cells. **Second**, we are able to customize the VP's device attributes, but this function is not offered by Cells. Cells is not designed to edit device attributes, and its user-level device virtualization modifies the VP's application framework layer, which can be detected by cloned apps.
2. **The assumption of a rooted device.** In the first attack model of data-clone attacks (victim identify theft), we assume that the victim's device is rooted, because we need to copy user-private credentials. However, if authentication schemes *entirely* rely on device-specific attributes, we do not require this aggressive assumption. As suggested by Reviewer A and B, we have mostly removed the part that exploiting the phone clone app to steal user credentials and made a weak claim on the first attack model.
3. **Attackers can intercept user password on a rooted device and then log in to the user's account.** Our argument is our attack via the auto-login function is much stealthier. Most apps do not allow duplicate access from different devices. For example, the messaging app KakaoTalk restricts that a single user can only log in from one device at a time. If the legal user is online, the attacker cannot log in to the same account by typing the victim's ID and password. *However, counting the number of login devices is typically not affected by the auto-login function from the same device, which leaves a backdoor for VPDroid to break through the login-device number limit.* Our demo video demonstrates our approach is better than intercepting user password (<https://youtu.be/cs6LxbDGPXU>).
4. **Session management.** Reviewer C says one possible defense solution is to detect how many devices are concurrently connecting to the server from the server side and thus it can place a limit on the number of concurrent connections. Unfortunately, such simple session management is not adopted by most mobile devices. The variable nature of mobile devices (e.g., the frequent connection switch between Wi-Fi hotspot and cell site) makes it difficult to determine an adequate number of connections, and complicates the session management on the app server side. *Although most apps do not allow duplicate access from different devices, they do permit multiple session requests that have the same device ID. This results in maintaining two or more connections per user.* We add session management discussion in Sec. 6
5. **As suggested by Reviewer C, we add performance evaluation (Sec. 5.1).** Figure 8 shows that no user-noticeable performance difference between running in VPDroid and running natively on the phone. VPDroid is better than Cells in Quadrant I/O, SunSpider, and Network results from 5% to 9%.
6. **Like Cells, VPDroid also supports running multiple virtual phones, but we did not claim this in our paper.** Because only the VP running in the foreground can always have the direct access to hardware devices. This is the requirement that we must meet to evade the detection of Android emulators. Before completing the data-clone attack, we cannot switch between multiple VPs. So, one VP is enough for data-clone attacks.
7. **VPDroid on Android 10.0: the leading authors are in the city that is under complete lockdown due to the coronavirus outbreak.** This also affects the development progress of VPDroid on Android 10.0, but we are confident that it will be ready in this summer!

8. **Additional functionality that would need to be virtualized:** Mobile device virtualization is susceptible to the new update and replacement of hardware devices in future Android versions. Especially, many devices are physically not designed for multiplexing, which will always be the key challenge of Android OS virtualization. To support additional functionalities, VPDroid's user-level virtualization solution offers a flexible and portable alternative without compromising transparency.
9. **Data-clone attack experiment result updates:**
 - a. The number of confirmed zero-day vulnerabilities increases from 14 to 21.
 - b. We figure out the failure reason when attacking the apps that rely on Android *AccountManager* APIs. The reason is *AccountManager* stores auto-login depended data under the directory of `"/data/system_xx/"` rather than `"/data/data/[app_name]/"`. After we copy the `"/data/system_xx/"` folder to the virtual phone, our data-clone attacks succeeded.
 - c. Before CCS submission, we redo our data-clone attack experiments (as shown in Table 1) on the latest versions of popular apps. The reason is that we observe many apps have already embedded device-consistency checks to secure their auto-login functions. On real devices, the successful number of data-clone attacks drops from 169 to 131, while on Xposed-based sandbox, this number drops from 196 to 148. In contrast, in VPDroid, data-clone attacks succeed in 234 out of 237 apps.
 - d. Google SafetyNet is an advanced anti-abuse/anti-fraud APIs. It helps developers determine whether their apps are running on a genuine Android device. We add a new experiment to show that SafetyNet is able to recognize all of the common virtualization environments and Android sandboxes, but it fails to detect VPDroid.

10. Screenshots of CVE and CNVD IDs discovered by us:

CVE-2019-15843



The screenshot shows the CVE List website interface. The CVE ID **CVE-2019-15843** is highlighted with a red box. The entry details include:

- CVE ID:** CVE-2019-15843 (Learn more at National Vulnerability Database (NVD))
- Description:** A malicious file upload vulnerability was discovered in Xiaomi Mi9T mobile phones 1-6.3.9.3. A particular condition involving a man-in-the-middle attack may lead to partial data leakage or malicious file writing.
- References:**
 - CONFIRM: <https://sec.xiaomi.com/post/152>
- Assigning CNA:** MITRE Corporation
- Date Entry Created:** 20190830
- Phase (Legacy):** Assigned (20190830)
- Votes (Legacy):** 0
- Comments (Legacy):** 0
- Proposed (Legacy):** N/A

The entry is an entry on the [CVE List](#), which provides common identifiers for publicly known cybersecurity vulnerabilities.

SEARCH CVE USING KEYWORDS:

You can also search by reference using the [CVE Reference Map](#).

For More Information: [CVE Request Web Form](#) (Select "Other" from dropdown)

账号设置

个人信息

我的资产

漏洞上报

漏洞上报

批量上报漏洞

我的任务

我的任务

我的关注

资产关联漏洞

漏洞收藏

我的荣誉

我的证书

原创漏洞积分

我的消息

评论管理

系统消息(1)

我的拓展

我的扫描

验证环境

批量数据接口

您好, 尊敬的, 欢迎回来!

手机: | 邮箱: 1

荣誉值: 0.925 | 用户积分: 70

立即上报漏洞

我上报的漏洞

您有(1)条消息未读, 请前往系统消息查看!

| 编号 | 漏洞标题 | 状态 | 上报时间 | 评论/关注 | 操作 |
|-----------------|--------------------|-----|------------|-------|----|
| CNVD-2019-42028 | 知乎APP存在逻辑缺陷漏洞 | 已归档 | 2019-10-16 | 0/0 | 跟踪 |
| CNVD-2019-42029 | 一直播APP存在逻辑缺陷漏洞 | 已归档 | 2019-10-16 | 0/0 | 跟踪 |
| CNVD-2019-42030 | 外研随身学APP存在逻辑缺陷漏洞 | 已归档 | 2019-10-16 | 0/0 | 跟踪 |
| CNVD-2019-42031 | 蜻蜓FM App存在逻辑缺陷漏洞 | 已归档 | 2019-10-16 | 0/0 | 跟踪 |
| CNVD-2019-42032 | 流利说-阅读APP存在逻辑缺陷漏洞 | 已归档 | 2019-10-16 | 0/0 | 跟踪 |
| CNVD-2019-42033 | 小影APP存在逻辑缺陷漏洞 | 已归档 | 2019-10-16 | 0/0 | 跟踪 |
| CNVD-2019-42034 | CAD快速看图APP存在逻辑缺陷漏洞 | 已归档 | 2019-10-16 | 0/0 | 跟踪 |

共 7 条