# LaChouTi: Kernel Vulnerability Responding Framework for the Fragmented Android Devices

Jingzheng Wu, Mutian Yang
General Department, Institute of Software, The Chinese Academy of Sciences, China
State Key Laboratory of Computer Sciences, Institute of Software, The Chinese Academy of Sciences, China
jingzheng08@iscas.ac.cn

## ABSTRACT

The most criticized problem in the Android ecosystem is *fragmentation*, i.e., 24,093 Android devices in the wild are made by 1,294 manufacturers and installed with extremely customized operating systems [16]. The existence of so many different active versions of Android makes security updates and vulnerability responses across the whole range of Android devices difficult. In this paper, we seek to respond to the unpatched kernel vulnerabilities for the fragmented Android devices. Specifically, we propose and implement LaChouTi, which is an automated kernel security update framework consisting of cloud service and end application update. LaChouTi first tracks and identifies the exposed vulnerabilities according to the CVE-Patch map for the target Android kernels. Then, it generates differential binary patches for the identified results. Finally, it pushes and applies the patches to the kernels. We evaluate LaChouTi using 12 Nexus Android devices that have different Android versions, different kernel versions, different series and different manufacturers, and find 1922 unpatched kernel vulnerabilities in these devices. The results show that: (1) the security risk of unpatched vulnerabilities caused by fragmentation is serious; and (2) the proposed LaChouTi is effective in responding to such security risk. Finally, we implement LaChouTi on new commercial devices by collaborating with four internationally renowned manufacturers. The results demonstrate that LaChouTi is effective for the manufacturers' security updates.

## CCS CONCEPTS

• **Security and privacy → Mobile platform security**;

## KEYWORDS

Android Fragmentation, Vulnerability, Patching, Identification, Security

**ACM Reference format:**
Jingzheng Wu, Mutian Yang. 2017. LaChouTi: Kernel Vulnerability Responding Framework for the Fragmented Android Devices. In *Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17),* 6 pages.


*ESEC/FSE'17, September 4–8, 2017, Paderborn, Germany*

ACM ISBN 978-1-4503-5105-8/17/09…$15.00
https://doi.org/10.1145/3106237.3117768

https://doi.org/10.1145/3106237.3117768

## 1 INTRODUCTION

Android dominates the global smartphone market with a share of 86.8% in 2016Q3 according to the report from the International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker [11]. Google also said that there are over 1.4 billion active users every month on the Android platform worldwide. With more than 1.9 million applications or Apps, Android has touched almost every aspect of our lives nowadays, and thus has changed the way we think, study, communicate, travel and even entertain ourselves.

The most criticized problem in the Android ecosystem is *fragmentation*. By customizing the official versions of the Android system, 1,294 manufacturers make 24,093 unique Android devices by the year of 2015 [16]. In order to adapt to existing countless hardware platforms, carriers, countries/regions, ecosystem services and user requirements, the Linux kernel stack and the Android framework are extremely customized and tailored [1]. Furthermore, Android itself is also continuously evolving, making the active versions across from Froyo 2.2 to Nougat 7.0 simultaneously.

Fragmentation is both a strength and a weakness of the Android ecosystem [16]. The upside is that there are thousands of smartphone options on the market to choose from for the users while the downside is that the existence of so many different versions of Android makes security updates and vulnerability responses across the whole range of Android devices difficult. Due to the fragmentation, it leaves the vulnerabilities unpatched and the users permanently vulnerable.

The unpatched vulnerabilities, especially the kernel level vulnerabilities, in the above situation expose Android devices to the attackers for long attack windows [5, 12, 14, 23]. To defend against this kind of vulnerable threats, researchers have proposed various security policies and security enhancements [4, 6, 7, 13]. For instance, Security Enhancement (SE) Android developed by NAS, is available on Android version 4.2 and up, aiming to defend against various root exploits and application vulnerabilities [19, 20]. Many other systems, e.g., TrustDroid [7], XManDroid [6], FlaskDroid [8], and AppPolicyModules [4] are developed, which aim to protect Android devices.

As a vendor, Google releases security updates through OTA (On The Air) updates monthly. However, on one hand, a month is a sufficient time window that allows the attacks. More and more attacks even happen on the same day that the vulnerability becomes generally known, namely *zero-day exploitation*. For instance, CVE-2016-0728 is described as a zero-day "it affects all Android phones with versions KitKat and higher" by Israeli security firm Perception

Point. Because an exploit was released once the patch was well on its way [24]. On the other hand, Google's security OTA updates are only for its own Nexus devices. Therefore, not all the manufacturers are able to catch up with Google.

Although the prior work has been proposed to enhance the security, vulnerability exploitations such as root privilege escalation are still possible and unavoidable. In this paper, we improve the security update process by responding to the unpatched kernel vulnerabilities for the fragmented Android devices. We propose LaChouTi, an automated kernel security updates framework consisting of cloud service and end application. LaChouTi first tracks and identifies the exposed vulnerabilities for the target Android kernels. Then, it generates differential binary patches for the identified results, and pushes and applies the patches to the kernels. We evaluate LaChouTi leveraging 12 different Nexus devices with fragmented features, and find it is effective in reducing the unpatched vulnerability security risks. Specifically, we make the following contributions in this paper.

**New Techniques.** We develop a set of new techniques for automatically responding to the exposed vulnerabilities in the target kernels for the extremely fragmented Android devices. These techniques can be utilized by any of the device manufacturers to identify which vulnerabilities are related to their devices, and to release the patch updates timely.

**A New Framework.** We propose LaChouTi, an automated kernel security update framework that can patch all the vulnerabilities once known. It tracks and patches the exposed vulnerabilities in the fragmented Android devices and improves the security capacity for Android manufacturers.

**Implementation, Evaluation, and Application.** We implement LaChouTi and evaluate it using different versions of Android on Nexus and other devices. Furthermore, by collaborating with four manufacturers, we work toward deploying LaChouTi in real applications. With the help of LaChouTi, those four manufacturers offer immediate security updates for some models of their devices.

## 2 BACKGROUND AND MOTIVATION

Android is open source [2] and is released publicly when a new version of Android is developed. Google releases the Nexus phones and tablets to act as their flagship Android devices, demonstrating Android's latest software and hardware features. Android manufacturers in Open Handset Alliance (OHA) follow up the new releases and modify the OS to support and enrich their own devices [1]. All those customizations from the manufacturers, carriers, and end sales make *fragmentation*. Therefore, the fragmentation makes secure updates and vulnerability responses across the whole range of Android devices difficult [18, 26].

Researchers found that on average 87.7% of Android devices are exposed to at least one of 11 known critical vulnerabilities [21]. And they increase every day, leaving billions of Android devices vulnerable [3, 9, 10, 21, 22, 25, 27]. Google does move relatively quickly and the patches have been rolling out. However, it is a big problem for the manufacturers to respond to all of their devices, especially for those who have many devices and many versions of Android.

New Android security vulnerabilities may be published at any time. However, as a vendor, Google only releases security updates through OTA updates monthly. To make things worse, Google stops to provide patches for Android 4.3 or prior, even if those may affect nearly a billion devices. For instance, as CVE-2014-8609, a typical SYSTEM-level information disclosure vulnerability, Google has decided not to provide patches for "legacy" Android WebView while only notifying the manufacturers. In addition, Google is committing new update policies, including keeping the now-monthly security updates and guaranteeing the major OS updates for two years from release.

The matter is more complicated for the device manufacturers. Because of the fast device update, the manufacturers are difficult to provide security updates to all devices they sold with limited manpower. Meanwhile, long upgrade cycles imply that plenty of users are going to be stuck on broken devices with known exploits. Thus, it is believed that a better update framework is needed.

To respond the unpatched kernel vulnerabilities, we have the following challenges.

**How to identify the vulnerabilities for each of the deeply fragmented Android kernels?** Each of the manufacturers maintains and customizes its own versions of kernel for their devices. Once a kernel vulnerability is published, manufacturers need to know whether it affects their devices immediately. Therefore, a solution should be able to automatically and accurately identify the vulnerabilities in the large source code repositories.

**How to generate the vulnerability patches for each version of the fragmented Android?** It is a key step to pull and apply a corresponding patch to the specific kernel source code repository for each identified vulnerability. The solution should support to generate differential binary patches between the original and patched kernels, store the binary patch sets in the manufacturers' private cloud, and provide index for the user queries.

**How to apply the binary patches for the fragmented Android devices in the wild?** Google's security OTA updates are only for their own Nexus devices, while the other manufacturers catch up according to their capabilities. A flexible solution should notify the users that differential binary patches are ready to apply, and allow the users to query the states of their fragmented devices. All the actions in the updating that the users need to do are button click and waiting for rebooting. After rebooting, the vulnerabilities should be eliminated from the devices.

## 3 DESIGN AND IMPLEMENTATION

To address the challenges and improve the Android security update process, we propose and implement an automated security update framework, namely LaChouTi. With LaChouTi, manufacturers can respond to the unpatched kernel vulnerabilities by tracking and identifying the vulnerabilities for fragmented Android devices.

Figure 1 shows the architecture of LaChouTi, which consists *manufacturer side* and *user side*. At a high level, LaChouTi offers private cloud service to Android users, mainly including four steps ① - ④. The first three steps are on the manufacturer side. In step ①, LaChouTi tracks and identifies the exposed vulnerabilities for the target Android kernels. Then, it generates differential binary patches, tests the patched kernels in step ② and stores the patches

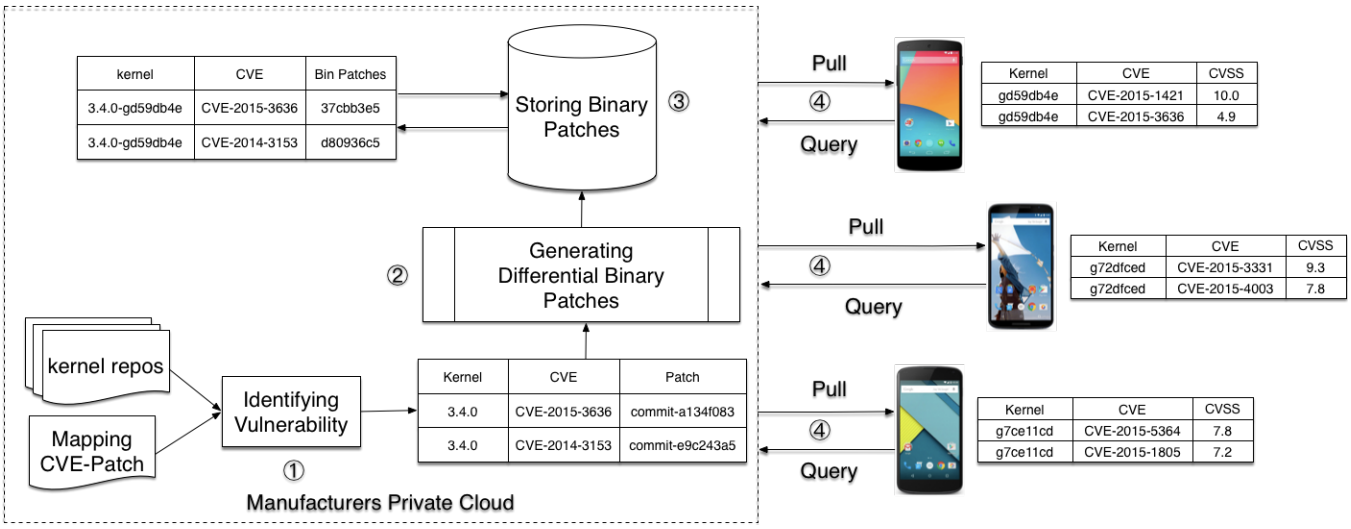



**Figure 1: Architecture of LaChouTi. LaChouTi consists the manufacturer service side and the end application side, and it offers timely and flexibly kernel vulnerability patching updates for the fragmented Android devices by the private cloud service.**

in the private cloud in step ③. Step ④ is on the user side, where users query or pull the patch update and apply it to kernel with an Android application. By operating in the application, the kernel can be patched and the vulnerabilities will be eliminated after the devices rebooting. The key index that connects the cloud and the end users is the CVE-Patch map between the kernel version and the corresponding patches. LaChouTi repeats these ① - ④ steps automatically, and updates the devices as soon as the vulnerabilities are publicly known.

## 3.1 Identifying Vulnerabilities

The first key step of LaChouTi is *vulnerability identification*, named *idPatch*. idPatch is mainly designed (1) to track the exposed vulnerabilities and build CVE-Patch map; (2) to identify the vulnerabilities from the large and deeply customized source code bases; and (3) to output the patches for the identified vulnerabilities [15].

Firstly, idPatch tracks the vulnerabilities published on CVE and build CVE-Patch map according the kernel source code repository. CVE-Patch map is a set of key-value pairs, whose key is CVE id and the value is patch. Then, idPatch extracts information from the patches, which includes the original/patched code segments and the position identifiers. Thirdly, idPatch divides the source files and patches into groups. Each group consists of a patch set and the related source file. Fourthly, idPatch scans the vulnerabilities by comparing the source code using a *sliding window algorithm*, where the length of the window is decided by the normalized patch file. Finally, after identification and refining, idPatch outputs the patch sets for the target kernels.

## 3.2 Binary Patch Generation

With CVE-Patch maps, the patches are applied to the manufacturers' kernel repositories. Then, the patched kernel is cross-compiled and a new kernel is obtained. Finally, the differential binary patches are generated by using *bsdiff*, which is a tool for building small patches for executable files [17].

Manufacturers store the binary patches in private cloud which is only accessible to specified clients. LaChouTi provides the differential binary patches as security updates, and the update service is only accessible for the users with the right devices.

## 3.3 Patching Vulnerabilities

We develop an application, named *KernelSafe*, to get the push notices or to actively pull the updates from the manufacturers' private cloud. If there is a new differential binary patch notice, KernelSafe will query the CVE-Patch map, pull the corresponding patch into the local device, and then apply the binary patch to the kernel using *bspatch*. *bspatch* is a tool that used to applied binary patch to another binary file.

KernelSafe requires the appropriate permission of the device, which is easy to authorize from the manufacturers. It is recommended that the manufacturer signs the KernelSafe application with the system signature, and installs it as a built-in application. Comparing with the monthly OTA security updates provided by Google, LaChouTi provides binary patches almost as soon as the vulnerabilities are published.

# 4 EVALUATION

We implement LaChouTi and evaluate it using 12 Android devices as shown in Table 1. From Table 1, *Vers* indicates Android version, *Kernel* means Linux kernel version, *Vuln* presents the number of vulnerabilities and *Manu* indicates manufactory name. These devices all belong to the Nexus series but with different Android versions, different kernel versions and different manufacturers. We also evaluate four devices produced by four internationally renowned manufacturers. However, according to the agreements signed with them, we do not list them here.

**Table 1: Vulnerabilities identified from the devices.**

| Model | Vers | Kernel | Vuln | Manu |
|---|---|---|---|---|
| Nexus6P | 6.0 | 3.10.73-gcf36678 | 42 | Huawei |
| Nexus5X | 6.0 | 3.10.73-g60cf314 | 42 | LG |
| Nexus6 | 5.0 | 3.10.40-g72dfced | 105 | Motorola |
| Nexus5 | 6.0 | 3.4.0-g2aa165e | 198 | LG |
| Nexus5 | 5.0 | 3.4.0-g323bbd9 | 198 | LG |
| Nexus5 | 4.4.4 | 3.4.0-gd59db4e | 200 | LG |
| Nexus10 | 4.4.2 | 3.4.39-g5b5c8df | 156 | Samsung |
| Nexus10 | 4.2.2 | 3.4.5-gaf9c307 | 200 | Samsung |
| Nexus4 | 4.2.2 | 3.4.0-g7ce11cd | 205 | LG |
| Nexus7 | 4.4.4 | 3.1.10-g1e42d16 | 190 | ASUS |
| Nexus7 | 4.3 | 3.1.10-g1e8b3d8 | 192 | ASUS |
| Nexus7 | 4.2.2 | 3.1.10-g05b777c | 194 | ASUS |

We evaluate LaChouTi from the following perspectives: (1) We implement the vulnerability identification module idPatch, and create a CVE-Patch map with 782 vulnerability and patch items. Then, we evaluate idPatch by identifying vulnerabilities from the kernels of the 12 devices. (2) We implement the cloud service and develop KernelSafe to get the push notices and to actively pull the updates from the cloud. We evaluate the differential binary patches generation and patching effects of LaChouTi using the devices.

## 4.1 Vulnerabilities Identification

**CVE-Patch Map.** idPatch uses a crawler to collect the detailed information of all the Linux kernel related vulnerabilities and creates a CVE-Patch map including 1028 pairs of the kernel vulnerabilities and their patches till to Dec., 2016. The time range of the CVEs is from 2005 to 2016, which covers all the vulnerabilities for the targeting device kernels v3.1.x-3.10.x (where ".x" indicates the minor revision of the kernel). In addition, the patch source code is provided along with the reported vulnerabilities and thus they are also crawled by idPatch. With the code and CVE-Patch map, binary patches can be generated for the target kernels using the technique discussed in Section 3.

**Vulnerabilities in the Employed 12 Devices.** Table 1 shows the identified vulnerabilities for each device using idPatch. Clearly, all the devices in the table belong to the Google Nexus series, which are released by Google as the flagship Android devices to demonstrate Android's latest software and hardware features.

From Table 1, we can see that except for the newly released Nexus6P and Nexus5X with less than 100 vulnerabilities, most of the devices have nearly 200 vulnerabilities. Although Google releases security updates monthly, not all of the devices update timely, and some devices even cannot access the update service. For the devices produced by other manufacturers, the results may be much worse.

**Vulnerabilities in Fragmentation.** Each participant of the Android ecosystem may customize Android and the kernel for their own purposes, which makes the Android world extremely fragmented. For the 12 devices in Table 1, fragmentation and vulnerabilities are easily observed.

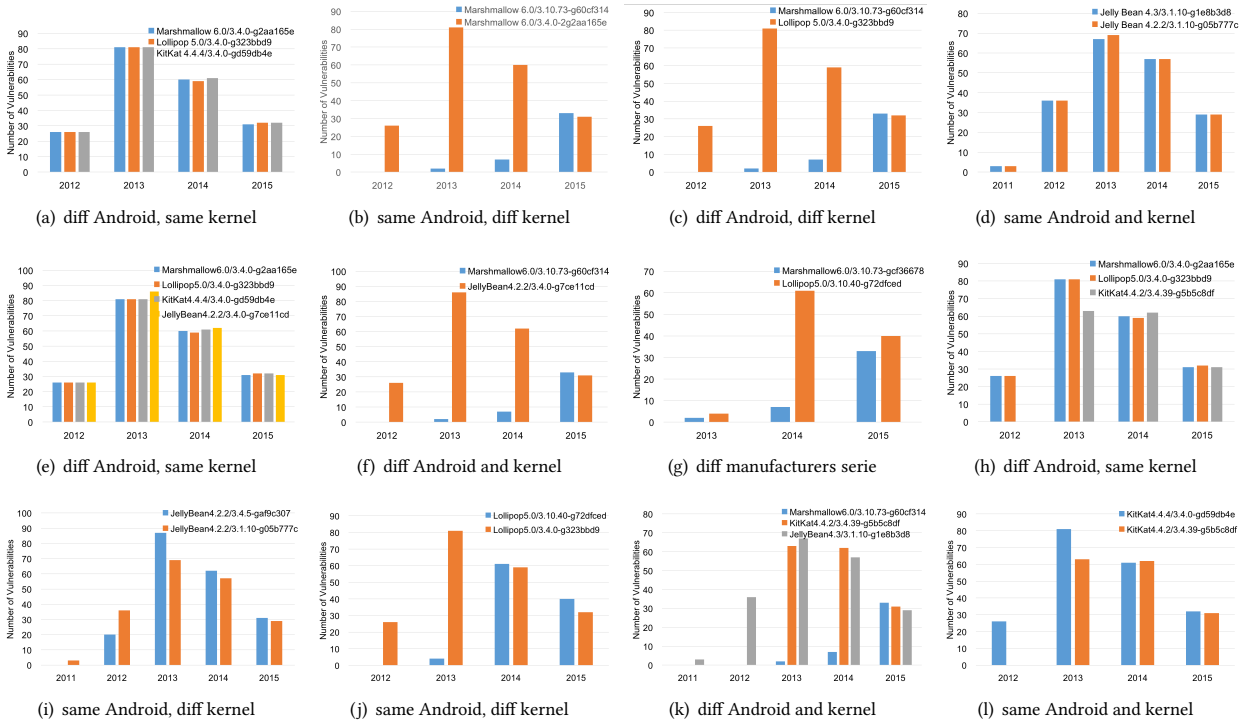
We classify the fragmentation situation into four top-level categories by manufacturers and the series in Figure 2. (1) Category 1: the devices are made by the same manufacturer and in the same series; (2) Category 2: the devices are made by the same manufacturer while in different series; (3) Category 3: the devices are made by different manufacturers while in the same series; and (4) category 4: the devices are made by different manufacturers and in different series. We examine the vulnerabilities of each fragmentation, and we have the following observations.

*Same manufacturer with same series.* According to Table 1, two series belong to this category, which are Nexus5 and Nexus7 produced by LG and ASUS respectively. This category can be further classified into four subcategories by the versions of Android and kernel. *Case 1: different Androids with same kernel.* From Figure. 2(a), three devices have almost the same number of vulnerabilities because the same kernel is affected mostly by the vulnerabilities. *Case 2: same Android with different kernels.* From Figure. 2(b), Nexus5X has much less vulnerabilities than Nexus5 mainly because v3.10.73 (Nexus5X) is a relatively newer kernel than v3.4.0 and thus some vulnerabilities have been patched. *Case 3: different Androids with different kernels.* From Figure 2(c), Nexus 5 with 6.0 has much less vulnerabilities than Nexus5 with 5.0 because v3.10.72 is a relatively newer kernel than v3.4.0. *Case 4: same Android with same kernel.* From Figure 2(d), two devices have almost the same number of vulnerabilities because they have the same kernel.

*Same manufacturer with different series.* According to Table 1, two series belong to this category, which are Nexus4 and Nexus5 produced by LG. Since only four devices locate in this category, this category can only be further classified into two subcategories by the versions of Android and kernel. *Case 1: different Androids with same kernel.* From Figure 2(e), four devices have almost the same number of vulnerabilities because they have the same kernel. *Case 2: different Androids with different kernels.* From Figure 2(f), Nexus5X has less vulnerabilities than Nexus 4 since the kernel version of Nexus 5X (v3.10.73 ) is relatively newer kernel.

*Different manufacturer with same series.* According to Table 1, Nexus 6 and Nexus 6P produced by Motorola and Huawei respectively belong to this category. Since only two devices locate in this category, this category is shown in Figure 2(g). From Figure 2(g), Nexus 6P has much less vulnerabilities than Nexus 6. This is mainly because v3.10.73 (Nexus 6P) is a relatively newer kernel.

*Different manufacturer with different series.* According to Table 1, all the devices belong to this category. This category can be further classified into four subcategories by the versions of Android and kernel. *Case 1: different Androids with same kernel.* From Figure 2(h), three devices have almost the same number of vulnerabilities since they have the same kernel. *Case 2: same Android with different kernels.* From Figure 2(j), Nexus 10 has more vulnerabilities than Nexus 7 while the kernel of Nexus 10 is released newer than Nexus 7. We manually audit the results, and found that some usual vulnerabilities were not patched as the other manufacturers in Samsung Nexus 10. This is mainly because the manufacturers might patch the vulnerabilities in their own ways. Figure 2(j) is also described in this case, Nexus 6 has less vulnerabilities than Nexus 5. *Case 3: different Androids with different kernels.* From Figure 2(k), Nexus 5X has much lesser vulnerabilities. *Case 4: same Android with same kernels.* From Figure 2(l), the two devices have almost the same number of vulnerabilities.




(a) diff Android, same kernel (b) same Android, diff kernel (c) diff Android, diff kernel (d) same Android and kernel

(e) diff Android, same kernel (f) diff Android and kernel (g) diff manufacturers serie (h) diff Android, same kernel

(i) same Android, diff kernel (j) same Android, diff kernel (k) diff Android and kernel (l) same Android and kernel

**Figure 2: Detailed Analysis of the Fragmented Devices, which is classify into four top- level categories (1) Category 1 shown in Figure 2(a) - 2(d), (2) Category 2 shown in Figure 2(e) - 2(f), (3) Category 3 shown in Figure 2(g), and (4) category 4 shown in Figure 2(h) - 2(l).**

**Summary.** From the above four top-level categories shown in Figure 2(a) - 2(l), we can see that (1) the existence of fragmentation is universal; (2) the security risk caused by fragmentation is serious; (3) idPatch is effective and practical in identifying the kernel vulnerabilities from source code; and (4) kernel version and release time are the key factors related to the vulnerabilities.

## 4.2 Vulnerabilities Patching

After identifying a vulnerability and its patch, LaChouTi applies patches for the targeting kernel source code, generates differential binary patches and offers those patches in private cloud. Then, application KernelSafe is used to query and pull the updates from manufacturer's private cloud. Finally, the kernels in devices are patched, and after patching the vulnerabilities are eliminated.

Take Nexus 5 as an example, it has 200 reported vulnerabilities ranging from the year 2012 to 2016. As maintainers of the devices, manufacturers should patch all of those vulnerabilities and especially patch the *High* severity ones immediately. Figure 3 shows binary patching process for Nexus5. If there is an update patch offered in manufacturer's cloud, the detailed vulnerabilities queried by KernelSafe are shown in Figure 3(a). Then, KernelSafe pulls the binary patch, applies the patch to the kernel, and reboot the device as shown in Figure 3(b) and Figure 3(c). After rebooting, the vulnerabilities are eliminated. At this time, if the user query again, there is no available patches as shown in Figure 3(d).


(a) Querying (b) Pulling (c) Applying (d) Querying

**Figure 3: Vulnerabilities Patching for Nexus5**

## 5 DISCUSSION

**LaChouTi: Novelty versus Limitation.** As shown in the experiments (Section 4), LaChouTi is effective in responding to the unpatched kernel vulnerabilities for the extremely fragmented Android devices by using the identifying and patching processes. However, it is important to note that the biggest difficulty encountered in LaChouTi is binary patching permission. Some devices do not allow the applications to access to the boot section or authorization is required. This is the limitation only to us, because manufacturers who has the Android source code can authorize KernelSafe proper

permissions. With those permissions, KernelSafe can patch to the vulnerable kernel or even directly flash a new kernel.

Although, Google and some manufacturers update their devices monthly or periodically, there are still attack windows. LaChouTi is aiming to narrow the attack window and support timely update without introducing additional works for the manufacturers. LaChouTi is flexible because either the whole or the identification process idPatch can be adopted by the manufacturers to advance their security update process.

**LaChouTi: Present versus Extension.** In this paper, we mainly focus on kernel vulnerabilities. idPatch is also effective to identify the known Android vulnerabilities. After identification, the manufacturers can use LaChouTi or their original approach to update the Android framework.

We just present the known vulnerabilities in evaluation, while LaChouTi also supports the unpublished ones. The manufacturers can write patches for the unpublished ones, generate binary patches and offer them as usual. By this way, the capable manufacturers can support more security updates.

In those evaluations, we found that the Android devices are extremely fragmented. The manufacturers, the series, the Android and the kernel versions may combine randomly, making vulnerabilities diversity. It is difficult for the third parties to provide a unified solution. Therefore, an updates framework easy to take and without additional efforts like LaChouTi is the best choice for the manufacturers.

## 6 CONCLUSION

In this paper, we propose LaChouTi to respond to the security risk induced by *fragmentation* in Android ecosystem. With LaChouTi, the unpatched vulnerabilities can be identified and patched immediately. We implement LaChouTi and evaluate 12 Nexus devices, finding 1,922 unpatched kernel vulnerabilities, and collaborate with four manufacturers on new commercial devices. The results show that: (1) the security risk caused by fragmentation is extremely serious; and (2) LaChouTi is effective in reducing such security risk. Our work has meaningful implications to manufacturers in helping them improve security update.

## ACKNOWLEDGMENTS

We thank the great help and suggestions of Professor Shouling ji, Tianyue Luo, Mei Liu, Chen Ni in this work. This work was partly supported by the National Natural Science Foundation of China No. 61303057, the project of Core Electronic Devices, High-end Generic Chips and Basic Software No. 2012ZX01039-004, the Provincial Key Research and Development Program of Zhejiang, China under No. 2016C01G2010916, the Fundamental Research Funds for the Central Universities, the Alibaba-Zhejiang University Joint Research Institute for Frontier Technologies (A.Z.F.T.) under Program No. XT622017000118, and the CCF-Tencent Open Research Fund under No. AGR20160109.

## REFERENCES

[1] Yousra Aafer, Nan Zhang, Zhongwen Zhang, Xiao Zhang, Kai Chen, Xiaofeng Wang, Xiaoyong Zhou, Wenliang Du, and Michael Grace. [n. d.]. Hare Hunting in the Wild Android: A Study on the Threat of Hanging Attribute References. In *CCS'15*. 1248–1259.

[2] Android. 2017. Welcome to the Android Open Source Project! (2017). http://source.android.com

[3] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. [n. d.]. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In *NDSS'14*.

[4] Enrico Bacis, Simone Mutti, and Stefano Paraboschi. [n. d.]. AppPolicyModules: Mandatory Access Control for Third-Party Apps. In *ASIACCS'15*. 309–320.

[5] Andrea Bittau, Adam Belay, Ali José Mashtizadeh, David Mazières, and Dan Boneh. [n. d.]. Hacking Blind. In *SP'14*. 227–242.

[6] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Thomas Fischer, Ahmad-Reza Sadeghi, and Bhargava Shastry. [n. d.]. Towards Taming Privilege-Escalation Attacks on Android. In *NDSS'12*.

[7] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Stephan Heuser, Ahmad-Reza Sadeghi, and Bhargava Shastry. [n. d.]. Practical and lightweight domain isolation on Android. In *SPSM'11*. 51–62.

[8] Sven Bugiel, Stephan Heuser, and Ahmad-Reza Sadeghi. [n. d.]. Flexible and Fine-grained Mandatory Access Control on Android for Diverse Security and Privacy Policies. In *SEC'13*. 131–146.

[9] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri. [n. d.]. A Study of Android Application Security. In *SEC'11*. 21–21.

[10] Heqing Huang, Sencun Zhu, Kai Chen, and Peng Liu. [n. d.]. From System Services Freezing to System Server Shutdown in Android: All You Need Is a Loop in an App. In *CCS'15*. 1236–1247.

[11] IDC. 2017. Smartphone Vendor Market Share, 2016 Q3. (2017). http://www.idc.com/promo/smartphone-market-share/vendor

[12] Vasileios P. Kemerlis, Michalis Polychronakis, and Angelos D. Keromytis. [n. d.]. ret2dir: Rethinking Kernel Isolation. In *SEC'14*. 957–972.

[13] Vasileios P. Kemerlis, Georgios Portokalidis, and Angelos D. Keromytis. [n. d.]. kGuard: Lightweight Kernel Protection against Return-to-User Attacks. In *SEC'12*. 459–474.

[14] Anil Kurmus and Robby Zippel. [n. d.]. A Tale of Two Kernels: Towards Ending Kernel Hardening Wars with Split Kernel. In *CCS'14*. 1366–1377.

[15] Tianyue Luo, Chen Ni, Qing Han, Mutian Yang, JingZheng Wu, and Yanjun Wu. [n. d.]. POSTER: PatchGen: Towards Automated Patch Detection and Generation for 1-Day Vulnerabilities. In *CCS'15*. 1656–1658.

[16] OpenSignal. 2016. Android Fragmentation Visualized (August 2015). (2016). http://opensignal.com/reports/2015/08/android-fragmentation/

[17] Colin Percival. 2017. Naive differences of executable code. (2017). http://www.daemonology.net/bsdiff/

[18] Chuangang Ren, Yulong Zhang, Hui Xue, Tao Wei, and Peng Liu. [n. d.]. Towards Discovering and Understanding Task Hijacking in Android. In *SEC'15*. 945–959.

[19] Stephen Smalley and Robert Craig. [n. d.]. Security Enhanced (SE) Android: Bringing Flexible MAC to Android. In *NDSS'13*.

[20] Ray Spencer, Stephen Smalley, Peter Loscocco, Mike Hibler, Dave Andersen, and Jay Lepreau. [n. d.]. The Flask Security Architecture: System Support for Diverse Security Policies. In *SEC'99*.

[21] Daniel R. Thomas, Alastair R. Beresford, and Andrew Rice. [n. d.]. Security Metrics for the Android Ecosystem *(SPSM '15)*. 87–98.

[22] Jingzheng Wu, Shen Liu, Shouling Ji, Mutian Yang, Tianyue Luo, Yanjun Wu, and Yongji Wang. 2017. Exception Beyond Exception: Crashing Android System by Trapping in "uncaughtException". In *ICSE'17*. 283–292.

[23] Wen Xu, Juanru Li, Junliang Shu, Wenbo Yang, Tianyi Xie, Yuanyuan Zhang, and Dawu Gu. [n. d.]. From Collision To Exploitation: Unleashing Use-After-Free Vulnerabilities in Linux Kernel. In *CCS'15*. 414–425.

[24] ZDNet. 2017. How to fix the latest Linux and Android zero day flaw. (2017). http://www.zdnet.com/article/how-to-fix-the-latest-linux-and-android-zero-day-flaw/

[25] Hang Zhang, Dongdong She, and Zhiyun Qian. [n. d.]. Android Root and its Providers: A Double-Edged Sword. In *CCS'15*. 1093–1104.

[26] Yajin Zhou and Xuxian Jiang. [n. d.]. Dissecting Android Malware: Characterization and Evolution. In *SP'12*. 95–109.

[27] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. [n. d.]. Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets. In *NDSS'12*.